
SMART NETWORKS

IFIP - The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- open conferences;
- working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

SMART NETWORKS

*IFIP TC6 / WG6.7 Seventh International Conference on
Intelligence in Networks (SmartNet 2002)
April 8–10, 2002, Saariselkä, Lapland, Finland*

Edited by

Olli Martikainen

*Lappeenranta University of Technology
Finland*

Kimmo Raatikainen

*University of Helsinki
Finland*

Jenni Hyvärinen

*Lappeenranta University of Technology
Finland*



KLUWER ACADEMIC PUBLISHERS
BOSTON / DORDRECHT / LONDON

Distributors for North, Central and South America:

Kluwer Academic Publishers

101 Philip Drive

Assinippi Park

Norwell, Massachusetts 02061 USA

Telephone (781) 871-6600

Fax (781) 681-9045

E-Mail <kluwer@wkap.com>

Distributors for all other countries:

Kluwer Academic Publishers Group

Post Office Box 322

3300 AH Dordrecht, THE NETHERLANDS

Telephone 31 786 576 000

Fax 31 786 576 474

E-Mail <services@wkap.nl>



Electronic Services <<http://www.wkap.nl>>

Library of Congress Cataloging-in-Publication Data

IFIP TC6 WG6.7 International Conference on Intelligence in Networks (7th : 2002 : Saariselkä, Finland)

Smart networks : IFIP TC6/WG6.7 seventh International Conference on Intelligence in Networks (SmartNet 2002), April 8-10, 2002, Saariselkä, Lapland, Finland) / edited by Olli Martikainen, Kimmo Raatikainen, Jenni Hyvärinen.

p. cm. — (International Federation for Information Processing ; 84)

Includes bibliographical references.

ISBN 978-1-4757-1024-3 ISBN 978-0-387-35584-9 (eBook)

DOI 10.1007/978-0-387-35584-9

1. Telecommunication—Data processing—Congresses. 2. Computer networks—Congresses. I. Martikainen, Olli. II. Raatikainen, Kimmo. III. Hyvärinen, Jenni. IV. Title. V. International Federation for Information Processing (Series) ; 84.

TK5105 .I33 2002

621.382—dc21

2002020616

Copyright © 2002 by International Federation for Information Processing.

Softcover reprint of the hardcover 1st edition 2002

All rights reserved. No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording, or otherwise, without written permission from the Publisher (Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061), with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

Printed on acid-free paper.

Contents

Preface	ix

INVITED PAPERS

Middleware for Mobile Applications Beyond 3G KIMMO RAATIKAINEN	3
Active Network Modelling and Simulation: A Behavioural Approach DOMINIQUE GAÏTI AND LEÏLA MERGHEM	19
A Service Deployment Architecture for Heterogenous Active Networks Nodes MATTHIAS BOSSARDT, LUKAS RUF, BERNHARD PLATTNER, AND ROLF STADLER	37
Driving Trends in Information and Communication Technology – Markets and Technologies REDA REDA AND HARMEN R. VAN AS	51

PART I - PLATFORMS

Investigating Service Discovery, Management and Network Support for Next Generation Object Oriented Services BILHANAN SILVERAJAN, JOONA HARTMAN, AND JANI LAAKSONEN	79
Virtual Private Components ALEK OPITZ, TORSTEN MIELKE, AND HARTMUT KOENIG	95
A New Organisational Framework for Network Modelling Using a Multi-Agent System MARC LEMERCIER AND DOMINIQUE GAÏTI	113
I3- A New Intelligent Generation for Internet Networks NADIA BOUKHATEM, BENOIT CAMPEDEL, HAKIMA CHAOUCHI, VINCENT GUYOT, FRANCINE KRIEF, THI MAI TRANG NGUYEN, AND GUY PUJOLLE	129
Accessing CORBA Services from Bluetooth Mobile Terminals SAMI MEROVUO, ARI VALTAOJA, AND ANDREI KOTCHANOV	149
Comparision of Softswitch-Architectures Using Parlay as Application Interface J. STADLER, I. MILADINOVIC, R. PAILER, AND M. VUCIC	161
Mobility Management in a Plug and Play Architecture MAZEN MALEK AND FINN ARVE AAGESEN	177

PART II - SERVICES

Inside iTV- Interactive Commerce Services for Broadcast Digital Television R. CHERNOCK, P. DETTORI, F. SCHAFFA, AND D. SEIDMAN	195
GCP MARIO ZUEHLKE AND HARTMUT KOENIG	211
Developing MLS Location-Based Service Pilot System JOUNI MARKKULA, ARTEM KATASONOV, AND ARTEM GARMASH	229

PART III - PROTOCOLS

Design of a Minimal Signaling Based Bandwidth Brokering Protocol	247
MOHIT AGARWAL AND ABHIK MAJUMDAR	
A Client Puzzle Based Public-key Authentication Protocol	263
CHUN-KAN FUNG, M.C. LEE, AND M.Y. WONG	
NAAP: A User-To-Network Authentication Protocol	279
HENRY HAVERINEN	
Test Bed for Plain C/C++ Protocol Implementations	293
HENRY HAVERINEN AND LASSI LEHTINEN	

Preface

Smartnet'2002 is the seventh conference on Intelligence in Networks arranged by the IFIP TC6 Working Group 6.7. The development in the contents of the conference series closely reflects the developments in networking. In the beginning, the focus was on Intelligent Networks (IN) and how to find solutions that would allow the quick creation and deployment of value-added programmable services in telecommunication networks. In the IN approach, intelligence was mostly centralized in the networks. Later on, research primarily focused on distributing intelligence over different parts of the network. The challenge was to apply the new developments of modern information technology in telecom service creation, deployment and management.

We are currently facing two conflicting trends: The increasing popularity of the Internet and the growth of mobile communications, which poses challenges for developing and distributing the intelligence in networks. The traditional Internet approach is to regard a network as a dummy pipe of bits and place all the intelligence in the end-devices. The traditional telecom approach is to regard the end-devices as dummy senders or receivers of bits and to put all the intelligence in the network. For fixed line access to Internet using a powerful PC, the Internet approach is plausible. However, for a nomadic user, who sometimes accesses the services through a fixed-line Internet connection and, at other times, through a wireless connection using a restricted-capability access device, the situation becomes intolerable. Intelligence needs to be distributed and partially replicated in end-user devices, access networks and the core network. The papers in this proceedings reflect these new trends.

The present research papers fall into three major topics: platform software, services and protocols. The sessions on platform software address various aspects of software infrastructure in networks including object-orientation, components, agents and plug-and-play. The session on services includes interactive TV, group communication and location awareness. The session on protocols discusses bandwidth brokers, authentication and protocol testing.

The editors express their sincere thanks to the reviewers for the evaluation of the manuscripts and to all the authors who submitted papers to the conference and contributed to the success of the event.

December, 2001

Kimmo Raatikainen, Olli Martikainen, and Jari Porras.

Steering Committee:

Olli Martikainen (chair), Finn Aagesen, Harmen R.van-As, Villy Baek Iversen, and Guy Pujolle.

Program Committee IFIP WG 6.7:

Olli Martikainen (chair), Finn Arve Aagesen, Scott Alexander, Harmen van As, Nadia Boukhatem, Georg Carle, Dominique Gaiti, Villy Baek Iversen, Terje Jensen, Jorma Jormakka, Masanori Kataoka, Hartmut König, Koos Koen, Eckhard Möller, Ahmed Al-Naamany, Elie Najm, Valeri A. Naoumov, Scott Nettles, Karl W. Neunast, Bernhard Plattner, Guy Pujolle, Kimmo Raatikainen, Reda Reda, Konstantin E. Samouylov, Manfred Schneps-Schneppe, Jonathan Smith, Otto Spaniol, Asser N. Tantawi, Osamu Takada, Vilas Wuwongse, Radu Popescu Zeletin, Yechiam Yemini, and Kenichi Yoshida.

Organizing Committee:

Kimmo Raatikainen (chair), Kirsimarja Blomqvist, Jarmo Harju, Jenni Hyvärinen, Jorma Jormakka, Ansa Laakkonen, Olli Martikainen, Martin Mäklin, Valeri Naoumov, Jari Porras, and Petri Pulli.

Reviewers:

Finn Arve Aagesen, Scott Alexander, Nadia Boukhatem, Dominique Gaiti, Terje Jensen, Jorma Jormakka, Hartmut König, Olli Martikainen, Karl W. Neunast, Jari Porras, Guy Pujolle, Kimmo Raatikainen, Reda Reda, Jonathan Smith, Otto Spaniol, and Asser Tantawi.

INVITED PAPERS

Middleware for Mobile Applications Beyond 3G

Kimmo Raatikainen

Nokia Research Center and University of Helsinki

Abstract Context-awareness and adaptability to changes in the execution and communication environment will be the key enablers for future applications that try to realise the “always-on” vision. This paper addresses fundamental research challenges and issues in middleware for mobile computing in systems beyond 3G. The key areas covered include adaptability, wireless communication, distribution of functionality, dynamic end-user systems, proximity, and open standards. In each area we identify crucial research issues. It should be noted that the division is not orthogonal: The same or similar issues are met in different key areas. We also outline a roadmap to construct a middleware solution to support future applications that will enable seamless service provisioning in heterogeneous, dynamically varying computing and communication environments.

Keywords: CORBA, Java, Internet protocols, adaptation, wireless communications, reconfiguration, interoperability, open standards

“Any technology distinguishable from magic is insufficiently advanced,” Gregory Benford.

1. INTRODUCTION

Context-aware applications will be of fundamental importance in mobile Internet that will be much more than Internet access from mobile devices. In fact, the Internet will be almost invisible since people will use mobile services and their favourite applications.

The transition to the Mobile Internet will be much more demanding than the transition to mobile phones in voice services. The primary reason is heterogeneous demands of various services and applications on the

underlying computing and communications infrastructure. Direct use of existing Internet applications in a mobile environment has usually been unsatisfactory. Web browsing, for example, over GSM Data Service using the standard HTTP/TCP/IP/PPP stack typically can utilize only about one third of the nominal bandwidth. Therefore, services and applications need to take into account the specific characteristics of mobile environments.

In principle, one could rewrite each service and application for Mobile Internet. This, however, would be both cumbersome and expensive hence slowing down development and deployment of services and applications. Therefore, essential generic functionality for mobile applications should be implemented in the infrastructure.

Middleware is a widely used term to denote a set of generic services above the operating system. Although the term is popular, there is no consensus of a definition. Nevertheless, middleware is used to denote a set of generic services above the operating system. A good summary can be found in the IETF RFC2768 [1]. Typical middleware services include directory, trading and brokerage services for discovery, transactions, persistent repositories, and different transparencies such as location transparency and failure transparency. Examples of middleware include Common Object Request Broker Architecture (CORBA) [2], Java 2 Enterprise and Micro Editions (J2EE and J2ME) [3, 4], Distributed Common Object Model (DCOM) [5], and Wireless Application Environment (WAE) [6]. Characteristically, the competing middleware specifications provide many similar but slightly different services. In order to overcome the problems due to different specifications, the Parlay Group [7] has specified a set of UML models and corresponding APIs that can be implemented in CORBA, Java and DCOM environments.

Context-awareness is considered as a fundamental property of future mobile applications. In essence, context-awareness means that the behaviour of an application depends on the current context. The context includes user preferences, device characteristics, properties of connectivity, geospatial location (time and space), state of service (session), usage history. Context-awareness also means that changes essential enough in the context trigger changes in the application behaviour. This kind of functionality is often called as adaptability.

In this paper we examine functional requirements of middleware supporting context-aware applications. We start by briefly discussing future mobile applications, their communication characteristics and their composition. The functional requirements are presented as research challenges in Section 3. The research space is divided into six research areas, each containing a set of key functional requirements that enable seamless service provisioning in heterogeneous, dynamically varying computing and

communication environments. In Section 4, we outline a roadmap, which tries to show a way from the current state-of-practice to the envisaged state-of-future.

2. FUTURE MOBILE APPLICATIONS

2.1 Communication Characteristics

The most significant feature in communication needs of applications in the Mobile Internet Era will be diversity. All kinds of applications will be in use. Their Quality-of-Service requirements (bandwidth, latency, reliability) as well as communication patterns will be numerous. Some applications will also adjust their behaviour according to the properties of connectivity. Future mobile terminals will have a few, say 3-6, applications simultaneously active. Some terminals will also be able to use different access technologies either simultaneously or one at a time. Therefore, the most important property of any communication system is its ability to handle mixtures of flows and traffic characteristics in a reasonable way.

When the application interacts with a human end-user, interaction mode is a good classifier of applications. Most of the applications fall into one of the following three categories:

- **Messaging.** These applications are non real-time. The underlying network can use store-and-forward, store-and-retrieve, or store-and-push mechanisms. The applications are not delay sensitive. The delays can be in the scale of seconds, minutes, or even hours. However, the delivery must usually be error free. The message size varies from a few bytes to several Megabytes. This category calls for efficient use of network resources.
- **Interactive content retrieval.** These applications are nearly real-time. The users do not tolerate long delays. Some content formats, such as audio and video, are delay sensitive but tolerate losses. In these applications playback buffers can be used to balance delay variation. Some other content formats, text in particular, are not delay sensitive but require error free transmission.
- **Rich call.** These applications require real-time communication. An example of today is voice call and that of the future is a multiparty video conference. These “calls” will be annotated by drawings, text files, etc.

When machine-to-machine applications, that is applications without human interactions, are also considered, then two main categories of applications can be identified: control and command applications and management applications:

- **Control and command.** In these applications short messages (payload from 10 to 10,000 bytes) are typical. The message delivery must be timely, reliable and error free. The delays can seldom be more than one second. The usual interaction pattern is either a simple request-reply or a single notification.
- **Management.** In management applications the interaction patterns can be quite complicated involving a dialog of several messages. Management applications require reliable and error free transmission but the time scale of delay bounds is in tens of seconds.

2.2 Application Composition

Figure 1 depicts a highly abstracted vision of how a service application is distributed among various application servers, network elements and terminal or end-user systems. It should be noted that, for simplicity, the figure only shows a single terminal device although multi-party applications will be much more important and challenging than one-party applications such as information browsing. In addition, we must also be ready to cope with end-user systems based on body area networks and home communication systems.

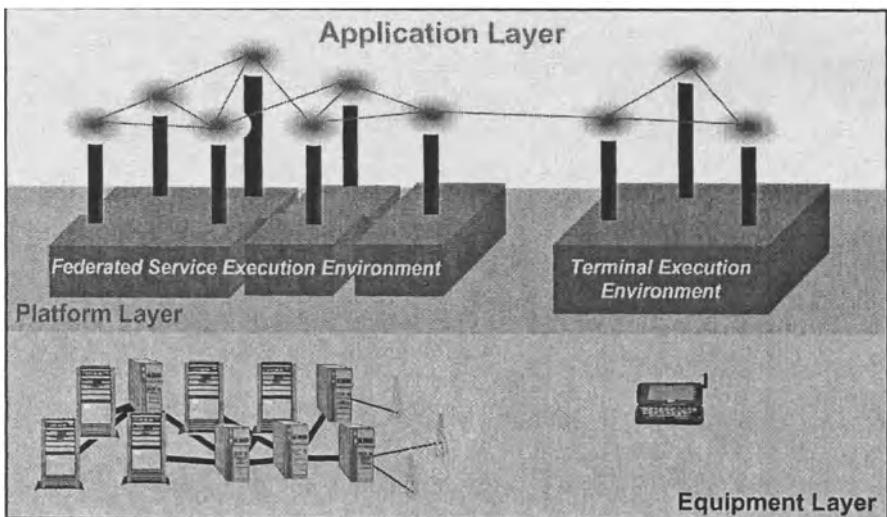


Figure 1. Partitioning and Distribution of Application Logic

The execution environments or the platform layer consist of middleware, operating systems and protocol stacks that should support fast service

development and deployment. The platforms should make it easy to divide the application logic into co-operating parts—someone may call them components, to distribute and configure these components as well as to redistribute and reconfigure them. Additional requirements for future mobile applications include adaptability to changes in the execution and communication capabilities, efficient use of available communication resources, dynamic configuration of end-user systems as well as ultimate robustness, high availability and stringent fault-tolerance.

The requirements for data accessed by these applications are quite similar. The execution environment should provide consistent, efficiently accessible, reliable and highly available information base. This implies a distributed and replicated worldwide “file system” that also supports intelligent synchronization of data after disconnections.

3. RESEARCH CHALLENGES IN MOBILE COMPUTING

In this section we identify the key research challenges in the future software systems enabling seamless service provisioning in heterogeneous, dynamically varying computing and communication environments. The division is not orthogonal; same or similar research topics and issues appear in more than one research challenges. We have divided the research space into the areas of adaptability, wireless communication, distribution of functionality, dynamic end-user systems, proximity, and open standards. Other divisions can be found in [8-16].

3.1 Adaptability

Adaptability is one of the key research areas in nomadic computing. The basic principle of adaptability is simple. When the circumstances change, then the behaviour of an application changes according to the desires of a user—or more precisely according to principles ascribed to her.

Let us take an example: 1) Important messages or their shortened versions (at most 200 characters) should be sent me at any price. 2) Voice messages should be sent me if the cost is 10 cents, at most. 3) Emails are to be sent if the cost does not exceed 5 cents. Attachments may be dropped but the message cannot be shortened. Email traffic is not allowed to disturb reception of important messages or voice messages. 4) Only the reception of important messages can slow down the delivery of voice messages. 5) Only the reception of very important messages can delay the delivery of important messages.

Although the example above concerns only messaging, it is quite long and far from a precise definition. In that sense it is typical and sets an important research topic: How to translate the wishes of users, which are almost always inaccurate, incomplete and sometimes even contradictory, into a set of rules precise enough for processing to be automated with sufficient reliability?

Learning the wishes and desires of a user is a crucial part of adaptability. Fully automated learning may be a utopia. Starting from an empty set of knowledge would take too much time. The learning path would necessarily contain too many malfunctions that upset the user who will be ready to discard the gadget.

Instead, starting from teaching is much more fruitful. In the same way as a child in a family is taught the habits of the family in a matter of years, the personal agents are taught the habits of the user in a few days. In this process the user interface is crucial: How and in which forms are users willing to give feedback? Most probably the feedback system cannot be unified but must be personalized for each user. The common attitude of a layman user—"I give feedback as I will"—must be the starting point of the user interface design. In particular, obtaining positive feedback is problematic. In some cases, missing feedback can be interpreted as a correct selection of action. However, in other cases that interpretation is questionable or even misleading.

Another crucial problem is the size and computational complexity of the knowledge base. Most probably the knowledge needs to be partitioned so that each subset is small enough. The partitioning, however, is not alone sufficient. The models presenting the subsets of the knowledge must be combined in different ways for different purposes. It should be noted that this is not the only application of partitioning in nomadic computing.

Adaptability cannot only be reactive. When the battery dies or the connectivity breaks, many actions are impossible. However, something could have been done beforehand. Therefore, adaptation must also be proactive, which, in turn, requires predictability of the near future. An important question in predictions is to distinguish between the situations in which the user behaviour seems to be predictable and those being unpredictable.

The objective should not be a perfect system since that would take forever. Instead, the goal must be a system that often (or at least sometimes) behaves correctly but almost never behaves incorrectly. Doing nothing is quite often a lesser evil than taking the wrong action.

3.2 Wireless Communication

As a communication channel, air is problematic. In the last ten years the progress in coding has significantly increased the capacity of wired channels. Unfortunately these fruits cannot fully be utilized on wireless channels. The applied coding is always a compromise between information density and redundancy providing robustness against interference. The basic problem with wireless links is instability in the sense that the level of interference varies in time and place, and according to environmental conditions.

It is very often said that the speed of 2 Megabits per second, which is assumed to be available in the 3G, will be fast enough. However, in the history of computing, the spare capacity has never been left unused. In addition, one should notice that the capacity and speed of wired connections has increased much faster. For each magnitude of improvement in wireless communications there has been an improvement of 2-3 magnitudes in wired communications.

The problems of wireless links are not uniform. We have wireless LANs, satellite links, cellular networks, and short-range radio links. Each poses specific problems of its own. Therefore, wireless communications must be regarded as a polymorbid patient.

Each of the access technologies mentioned above differs, at least in some aspect, from the others. Hence, the support system of a nomadic user must be able to support communication links of different kinds. It must enforce the higher layers of communications to adapt to the situation at hand. However, the adaptation of communication is not sufficient, the behaviour of applications also needs to be adapted.

3.3 Distribution of Functionality

Situations, in which a user moves with her end-device and uses information services, are challenging. Moreover, the nomadic user of tomorrow will not appreciate a static binding between her and an access device; not even in the case of multi-mode access devices that can handle several access technologies including wireless LAN, short-range radio, and packet radio. It must be possible to move a service session (or one end-point of a service session) from one device to another.

In these situations the partitioning of applications and the placement of different co-operating parts is a research challenge. The support system of a nomadic user must distribute, in an appropriate way, the parts among the end-user system, network elements and application servers. In addition, when the execution environment changes in an essential and persistent way, it may be beneficial to redistribute the co-operating parts. The redistribution

or relocation as such is technically quite straightforward but not trivial. On the contrary, the set of rules that the detection of essential and persistent changes is based on is a challenging research issue.

Another research issue of fundamental importance in distribution is fault-tolerance. Replication, which is a commonly used method to achieve fault-tolerance in traditional distributed systems, is not sufficient alone. The baseline applications must remain operational, at least in a tolerable manner, even if some services of the underlying execution environment cannot be utilized. Here we have a nice additional requirement for those who research modularisation of application and adaptability.

3.4 Dynamic End-User Systems

The end-user devices of today are primarily integrated units like PDAs, laptops, and mobile phones. However, the situation will change in the future. The successor of the current mobile phone, or at least the successor of its successor, will be quite different. It will not disappear or lose its importance but its role will be very different. This “*FuturePhone*” will be the core of the personal computing and communication system. The *FuturePhone* probes its surroundings looking for suitable peripheral devices such as displays, input devices, processors, fast access memories and access points to communication channels. It dynamically builds up the most appropriate end-user system that can be auto-configured. The set of rules how the appropriateness of different configurations is evaluated is far from trivial. In addition, we must remember that the *FuturePhone* must be able to remain operational even if it cannot find any suitable peripherals. Moreover, the *FuturePhone* must be able to establish different kinds of ad-hoc networks that are simultaneously operational.

In the dynamic configuration we have a huge space of research items. On the conceptual level there are research issues related to profiles, various kinds of context also including the social context, roles and trust. On the technical level we must solve the problems related to authentication, authorization, and delegation.

3.5 Proximity

Location was identified already in the NII white paper [12] as one of the key issues in nomadicity. The baseline location-based services are not anymore a research issue. Simple location-based queries and notifications/advertisements are ready to be deployed. Perhaps, the privacy issues still need further studies, in particular the enforcement of regulatory

rules that differ from one country to another. The challenge is to design a flexible policy mechanism that allows fast changes in applicable policies.

Another challenge, not necessarily a research issue, is the way to cope with a plethora of location standards. The Location Interoperability Forum (LIF) [17] was established to an industry-wide initiative for promoting a secure, simple, ubiquitous, and interoperable location services solution to improve technology and maximize business. One of the underlying observations was a need for a global forum to:

- address the complexity and multiplicity of current solutions and market situation;
- define, develop, and promote an interoperable location services solution that is open, simple, and secure; and
- allow user appliances and Internet-based applications to obtain location information from the wireless networks independent of their air interfaces and positioning methods.

A research issue, however, is the prediction of end-users' future locations. The predictions may, for example, be used to schedule an appointment. An important aspect of location predictions is to combine information from different sources—from models of user movements and behavior, calendar, and to-do-lists, for example.

To include time and location into queries may not be straightforward. As an end-user I would like to post a request to “book a table for me and my wife in a nearby restaurant today at 7 pm.” When the booking service starts to process my request, it meets several challenges:

- Where I am going to meet my wife today, sometime between now and 7 pm?
- How much we are ready to walk to a suitable restaurant? Do we prefer a moderate restaurant within ½ mile over a better one within 2 miles?
- What kind of restaurant we prefer? My preferences may be different from those of my wife? How to balance between our preferences to meal, to walk for a good meal, etc?

The example above demonstrates that the personal profile indicating personal preferences and matching rules is not trivial. Therefore, research on personal profiles and on matching rules is crucial. Furthermore, the semantics of nearby or rating value (single number) of proximity is of major importance.

3.6 Open Standards

The assumption that the services are available anytime and anywhere calls for open standards. The time when a closed proprietary solution was sufficient is history. The baseline services must be available and operational

everywhere. Therefore, open worldwide standards are necessary. However, this does not mean that closed proprietary solutions are not developed anymore. They are still produced but they must be interoperable with the rest of the world. The proprietary solutions will have an important but specific scope. They will be used as means to customize and differentiate services and products.

4. ROADMAP

When our research agenda and those of the other groups—a summary can be found in [18]—are put together, then the first conclusion must be that there is a long to-do list before ubiquitous mobile information society is real. The research groups devoted to mobile, nomadic, pervasive or ubiquitous computing have taken different paths in order to achieve their goals. However, many groups have adopted the traditional paradigm of constructive computer science: To implement prototype(s) based on the proposed solutions in order to demonstrate the feasibility of the proposal.

Prototype implementation is fun. You can quite easily hire hackers to do the programming. However, we are not quite sure how productive, in a long run, prototype implementations are in a research process. Therefore, we have downgraded the role of prototyping from the first class to the economy class. In essence, prototyping is only a tool in verification and validation but not a driving force.

4.1 All IP Networks and Middleware

The current trend in developing forthcoming telecommunication networks is to utilize Internet protocols. An immediate implication is that IP is the layer 3 protocol and that the addresses are IP addresses. However, this is not sufficient. Other solutions—both above and below the IP protocol—are also needed to meet the requirements for next generations of telecommunication networks. Issues under study in the Internet community and in various standardization bodies, forums and consortia of telecommunications include mobility, Quality-of-Service, security, management of networks and services, discovery, ad-hoc networking, dynamic configuration, and geospatial location.

Another significant trend is the requirement of ever-faster service development and deployment. An immediate implication has been the introduction of various service/application frameworks/platforms. Middleware is a widely used term to denote a set of generic services above the operating system. Although the term is popular, there is no consensus of

a definition [1]. However, typical middleware services include directory, trading and brokerage services for discovery, transactions, persistent repositories, different transparencies such as location transparency and failure transparency.

The benefits of middleware software, such as CORBA, J2EE, J2ME , and DCOM, are obvious; see e.g. [19-21]. The most significant advantage—when compared to a pure IP-based socket programming approach—is in the improved programming model. The middleware solutions are usually based on object-oriented programming and method invocations. The invocations are based on strongly typed interfaces that provide both compile and run time error checking. They also hide many implementation details. Therefore, middleware-based application development is much faster than the Internet based one.

Although the middleware solutions provide superior programming environments over the Internet solution, they also pose serious shortcomings. The fundamental problem of the current middleware specifications is that they only take advantage of a narrow subset of useful Internet protocols. The current middleware specifications were born in a time when the Internet protocols were a synonym of the TCP/IP transport. Later they have developed solutions of their own for Quality-of-Service, directory, discovery, and so on, independently from each other and from the IETF specifications. The fundamental research challenge is the question of how the developments in Internet protocols and in different middleware solutions could be harmonized.

By harmonization we mean two things. Firstly, we need to solve the problem of incorporating evolving Internet solutions of Quality-of-Service, mobility, discovery, and security into the existing middleware specifications without breaking those specifications. Secondly, we need to find solutions to how different middleware solutions can become interoperable in the sense that components of an application can be executed on different middleware platforms.

Figure 2 outlines a framework for our roadmap. It decomposes the abstract notion of execution environment in Figure 1 into a layered architecture. One should notice that the operating system is not shown in the figure because it would introduce another dimension. The roadmap is the way how we assign the research issues identified in our research agenda into various functional blocks in Figure 2 and further into various standardization activities in relevant forums including IETF [22], W3C [23], 3GPP [24], OMG [25], Parlay [26].

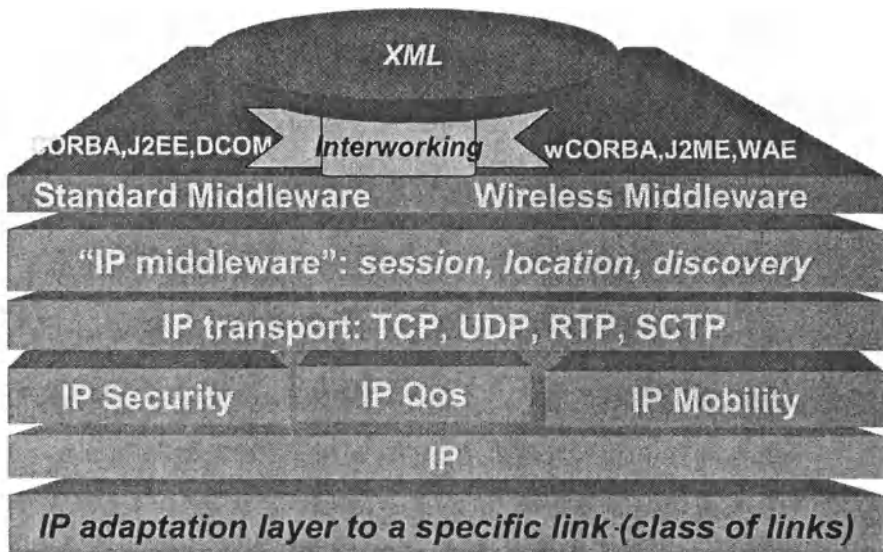


Figure 2. Layered Architecture of Middleware and Internet protocols

We need to work on details in various working groups of standardization bodies. We also need to have a vision where to go. Here we need both practitioners—ones who know all the details—and visionaries—ones who know almost nothing in details but almost anything in general.

4.2 Interoperability

The increasing diversity of devices—terminals, network elements, and application servers—imply that different middleware solutions will be in use. It is highly improbable that there will be, in a near future, a single dominant middleware platform which would be good enough for different devices and purposes. This heterogeneity requires interoperability on two levels: between middleware platforms and between parts of an application running on different middleware platforms.

The interoperability between different platforms is quite mature. In particular, the OMG has been the leading forum in specifying interoperability bridges between CORBA and other middleware platforms. In contrast, the interoperability between parts of an application running on different middleware platforms is still immature. There are practically no tools available to support this kind of interoperability. The burden of interoperability is totally left to application developers.

The OMG, however, has started a comprehensive development of a new architecture denoted as Model Driven Architecture (MDA) [27, 28]. The objective is to interrelate IDL specifications, UML modeling [29], Meta-Object Facility [30], and XML Metadata Interchange (XMI) [31]. The forthcoming MDA might provide a useful starting point for tools supporting interoperability between parts of an application running on different middleware platforms. A sketch of an interoperability framework based on UML is outlined in Figure 3.

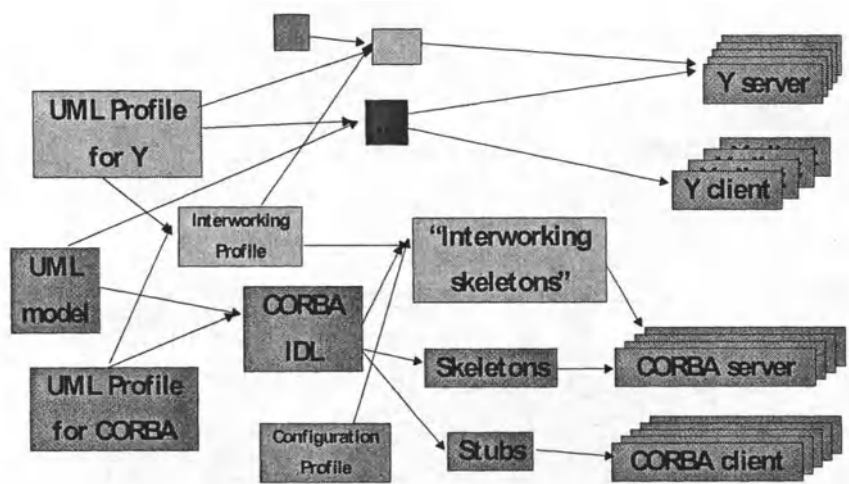


Figure 3. A UML-based Interoperability Framework

5. CONCLUSIONS

Almost ten years ago Mark Weiser [8] identified the target in future computing to be invisible computing, to hide computers and communications. The research agenda of Weiser [9] was later essentially extended by Alan Demers [10] and by Leonard Kleinrock [11, 12]. Today we mainly use in our research agenda the terms and concepts originating in the pioneering work in Xerox PARC [8-10]. This does, however, not mean that the research has done almost any progress in the last ten years. On the contrary, accumulated shared knowledge over the problem space as well as pros and cons of different proposed solutions have been utilized. We have been able to progress both in details and in two or three party interactions.

The worldwide research community has provided some insight into ubiquitous mobile information society. There is, however, a huge lot to be done. We believe that we can decompose the research and problem space

into manageable R&D initiatives. Long-term interactions between identified subsystems are, perhaps or most probably, manageable.

Our primary concern is the overall picture. It is easy to draw a few PowerPoint slides. However, the road to a product or to a prototype is long and rocky. The divide-and-conquer principle is a necessity. However, very many can divide a whole into pieces, almost anybody can manage a well-defined piece but to conquer the whole is the challenge.

REFERENCES

- [1] B. Aitken et al., "Network Policy and Services: A Report of a Workshop on Middleware," RFC 2768, February 2000.
- [2] OMG, "CORBA 5. Specification," <http://www.omg.org/>.
- [3] JavaSoft, "Java 2 Enterprise Edition," <http://java.sun.com/j2ee/>.
- [4] JavaSoft, "Java 2 Micro Edition," <http://java.sun.com/j2me/>.
- [5] Eddon, G. and Eddon, H. "Inside Distributed Com," Microsoft Press, 1998
- [6] WAP Forum, "Wireless Application Environment," WAP Specification WAE-
- [7] Parlay Group, "Parlay 2.1 Specification," <http://www.parlay.org/>.
- [8] M. Weiser, "The Computer for the Twenty-First Century," Scientific American, September 1991, 94-104.
- [9] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Communications of the ACM, July 1993, 74-84.
- [10] A.J. Demers, "Research Issues in Ubiquitous Computing," Proc. ACM PODC'94, August 1994, 2-8.
- [11] R. Bagrodia, W.W.Chu and L. Kleinrock, "Vision, Issues, and Architecture for Nomadic Computing," IEEE Personal Communications, December 1995, 14-27.
- [12] Cross-Industry Working Team, "Nomadicity in the NII," available from <http://www.lk.cs.ucla.edu/LK/lkxiwt/>.
- [13] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Proc. ACM SigMobile, 1, 1, pp. 1-7, April 1997.
- [14] G. Banavar et al., "Challenges: An Application Model for Pervasive Computing," in Proc. MobiCom'2000, August 2000, 266-274.
- [15] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, 8, 4, pp. 10-17, August 2001.
- [16] L. Kleinrock, "Breaking Loose," Commun. of the ACM, 44, 9, pp. 41-45, September 2001.
- [17] Location Interoperability Forum, <http://www.locationsforum.org/>.

- [18] K. Raatikainen, "Functionality Needed in Middleware for Future Mobile Computing Platforms," Proc. ACM Advanced Topic Workshop: Middleware for Mobile Computing November 16, 2001, Heidelberg, Germany
- [19] K. Raatikainen, "Middleware Solution for All IP Networks," Proc. 3Gwireless workshop, March 23-26, 2001, London, UK, pp. 335-340.
- [20] K. Raatikainen, "Middleware for Future Mobile Networks," Proceedings of IEEE International Conference on 3G Wireless and Beyond, May 30 – June 1, 2001, San Francisco, Calif., pp. 722-727.
- [21] K. Raatikainen, "Middleware," Chapter 3.2.7 in MITA Handbook, IT Press, November 2001.
- [22] IETF Home Page, <http://www.ietf.org/>
- [23] W3C Home Page, <http://www.w3.org/>
- [24] 3GPP Home Page, <http://www.3gpp.org/>
- [25] OMG Home Page, <http://www.omg.org/>
- [26] Parlay Home Page, <http://www.parlay.org/>
- [27] R. Soley, ed. An OMG discussion paper on Model Driven Architecture, version 3.2. OMG document omg/2000-11-05, November 2000.
- [28] OMG MDA Home Page, <http://www.omg.org/mda/>
- [29] OMG, UML Specification. OMG document formal/2000-03-01, March 2000.
- [30] OMG, MOF Specification. OMG document formal/2000-04-03, April 2000.
- [31] OMG, XMI Specification. OMG document formal/2000-11-02, November 2000.

Active Network Modelling and Simulation: A Behavioural Approach

Dominique Gaïti and Leïla Merghem

¹ *Laboratoire de Modélisation et de Sécurité des Systèmes (LM2S)*

University of Technology - Troyes (UTT) – France

12 rue Marie Curie, BP 2060, 10010 Troyes Cedex

Email : dominique.gaiti@utt.fr

² *LIP6*

University of Paris 6

8 rue du Capitaine Scott

75015 Paris – France

Email : {Leila.Merghem, dominique.gaiti} @lip6.fr

Abstract Network environments become more and more complex in terms of new services, quality of service to handle and user demand for increasing throughput. All these facts have to be performed in real time (or almost), and consequently, network management and control are difficult to realise in this continuously changing environment. The network is now unpredictable and there is a real lack in the modelisation and simulation tools to handle this dynamicity. In this context, agents and multi-agent systems seem to be good candidates to provide a way to model and to manage the network dynamicity. This is due to the fact that agents can perform tasks in an autonomous, distributed, adaptable and cooperative manner. Our goal here is to model an active network by using a multi-agent approach and more precisely a behaviour based agent approach. Instead of representing the actions of the network elements by complex algorithms, we rather describe them by their behaviours which are usually simple and familiar words, inspired in part from human behaviour. On this basis, we have made simulations of an active network environment, and implemented different combinations of four individual behaviours by using a multi-agent platform called Swarm. Results of these simulations are significant and show that we do not loose important packets (with priority) any more when we introduce a behaviour in the nodes.

Keywords: Multi-agent system, Behavioural modelling and simulation, Adaptive systems.

1. INTRODUCTION

The complexity (in terms of services and multimedia streaming) and dynamicity of telecommunication networks are continually growing, making network management, and control more and more difficult. The domain of modern telecommunications is characterised by its continuous changes with decentralised data and control, and mobility. The agent technology allows us to manage such a dynamic world, due to its main characteristics namely the autonomy, the ability to communicate with the others in order to solve some common problems in a decentralised manner as well as learning aptitudes.

Our aim is to model an active network by using a multi-agent approach, in order to include some intelligent and dynamic control to avoid congestion and lost packets in order to get a better decentralised management task and global performance of the network.

In order to obtain an agent-based model, several steps have to be followed [5], (1) determine what we have to model, i.e. what are the entities we need to represent, (2) decide what must be active in a network: will it be nodes? (a certain percentage or all the nodes?), will it be packets? (which type of packets?) or the both of them?, (3) choose an agent model (number of agents, their nature (cognitive, reactive, hybrid, adaptive,...), the relations with the other agents (co-operation, negotiation, competition,...), (4) choose the software platform that will be used for the implementation of our model, (5) implement, verify and validate the model.

In this paper, we will not develop all the aspects of an agent-based modelling, but we would rather present the behavioural approach and benefits that are generated. We have implemented this approach with the help of a software platform called Swarm.

The remainder of this paper is organised as follows. We first present the target system of our modelling, i.e. active networks in section 2. Then, we describe some behavioural modelling in other domains than network in section 3. Some behaviours that could be applied in networks are proposed in section 4, followed by the description of the simulations realised up to now and the results obtained (section 5). In section 6 we conclude the paper.

2. ACTIVE NETWORKS – THE TARGET SYSTEM

An active network is a network in which some components are dynamically programmable by third entities (operator, service provider, applications, users) [8].

Active networks are classified by the following approaches:

- active packets: where the deployment of services in nodes is conceptually integrated to the user flow, and in which life span of a service depends conceptually on the traffic in question;
- active nodes: where services are dynamically deployed in nodes, but not in the same flow as the data processed by these services.

Several problems have occurred in current networks like the difficulty to integrate new standards and technologies into the shared network infrastructure, the poor performance due to redundant operations at several protocol layers, and the difficult integration of new services in the existing architectural model [13]. In order to solve these problems or at least a part of them, many projects related to active networks (Smart Packets [12], SwitchWare [1], COMPOSE [14], ANTS and Active IP (MIT) [16]...) are currently being or are already developed in different universities.

Supervision and control of networks are certainly the predilection fields of the active networks. Active networks are also used to make multicast protocols more reliable. It allows to gracefully solve problems that result of the implosion of the negative acknowledgement (NACK) and the useless retransmissions. Active networks can be also used to avoid congestion that may occur during the transmission of multimedia data, by allowing the routers to ask sources to reduce their flows in case of congestion.

However, this new approach generates new problems, due to this increment of complexity. There are new variables to be taken into account and also new resources to be managed and protected.

We are mainly interested in the dynamicity and complexity introduced by the active network concept through the active packets or active nodes. Conceptually, nothing in that kind of network will remain static. This dynamicity needs to be modelled and simulated and we propose the behaviour-oriented approach to do it, because no tool in the telecommunication area provides a real answer.

3. BEHAVIOURAL MODELLING

We propose to use behaviours to model the activities of the different network's elements. Thus, each element will have one or more behaviours, and its interactions with the others will be based on that behaviour. Moreover, instead of representing actions of network elements by complex algorithms, we rather describe them by simple and familiar words, inspired from human behaviour. We say for example that a node is faithful instead of saying that it always sends its packets to the same node, by following the same way, whatever the conditions of this route are. Thus, if a node A says to a node B that its behaviour is faithful, it must not send it the actions that it

is performing concerning the routing for example, because B can easily predict them. The profile of a node will be the association of its elementary behaviours, and interactions between the network's nodes will be the result of the interactions between their behaviours. The dynamicity of an active network will be provided by the change of behaviour in response to the local element's environment.

We are mainly interested in behavioural modelling because there are several successful behavioural simulations in other fields than networks (road traffic simulation ([3], [10], [15])), biologic and social phenomena simulation ([4], [6]). These fields are clearly as complex as the network area and contain factors that are generally out of human control. Despite these difficulties, scientists have obtained very good results, and have proven the reliability of multi-agent simulations, and specially their utility in understanding, explaining, and discovering new phenomena. For example, road traffic simulations can be a good source of inspiration for us. After all, in a telecommunication network, we face lots of problems that are also handled in a road network such as traffic management, priority management.

A good example of these systems is Trafficopter [10], a multi-agent system that provides a fast and reliable way of assisting the driver of a vehicle in deciding the most convenient route he has to take to reach his destination. The behaviour of drivers is affected by the information on the traffic conditions that lay ahead sent by the cars that are there now, or have been there recently (helpful behaviour). This information can also be collected as a result of a request sent by the driver (needy behaviour).

In the OLSIM project [3], [15], real time information on traffic is sent, in order to affect the behaviour of drivers and consequently reduce the traffic.

All these behaviours model different entities in the system whatever this system is.

4. BEHAVIOURS TO MODEL

4.1 What do we need to model?

The network entities that should be modelled are: nodes, links, and packets.

According to our point of view, an entity is active if it has a behaviour, which could change according to the state of the network (state of traffic and neighbour nodes, etc...).

For the moment, packets do not have a behaviour; they are routed from one node to another to carry data that might affect the behaviour and decisions of covered nodes. This is also valid to links that are passive

entities. Only nodes have one or several behaviours which modify the actions that they are performing on data (packets). Data can be more or less important depending on the quality the user is asking for its flows.

In our system, we distinguish three Qualities of Service [18]:

1. Premium packets have the highest priority. They have a total guarantee for the required service : they are never lost, and are not delayed ;
2. Olympic packets belong to the second class of priority. Packets must reach their destination (no lost) but with no guarantee of time;
3. Best Effort packets have the lowest priority. In fact, these packets are routed only if the bandwidth is available, otherwise they are lost.

These classes are under study in the IETF (Internet Engineering Task Force) normalisation group [18]. Each node has at least one behaviour that reflects its actions and the set of conditions to which it responds.

4.2 What kind of agents ?

Agents can be reactive, cognitive, hybrid or adaptive depending of the autonomy level, the capacity to reason or the ability to know their environment. We have opted for adaptive ones because they support, by definition, our goals. Adaptive agents are indeed entities that change their behaviour in the light of changing circumstances; they can sense their environment and act upon what they sense, hence the term adaptive [2], [5], [7]. They can be therefore a good candidate to represent active entities, that change their behaviour following the network states.

Adaptive agent simulations are often used in research to study issues that are too complex to be addressed by any other way. They routinely are used to model life and other complex, non-linear systems [9]. In this context, we will use adaptive agent simulations to evaluate and to understand complexity issues of the network environment.

4.3 Some behaviours in the network area

The notion of behaviour of a node is important because it allows the other nodes to acquire the profile of this node, and consequently be able to reason and take new decisions on the basis of what should that node do, and what should be its proper reactions.

In this paper, we will focus on the two following behaviours:

- a) **Careful**: this behaviour consists to observe the state of the buffers with the help of two thresholds (figure 1). When the buffer load is below 40%, the node has no particular action to do. But when the first threshold is reached, it starts to reject Best Effort packets coming from its own sources. From 70% of buffer load (2nd threshold), the node rejects all

Best Effort packets (even those coming from other nodes), and sends control messages to its nearest neighbours;

- b) **Careless**: the node starts to reject Best Effort packets coming from its sources when the first threshold (60%) is reached, and rejects Best Effort packets coming from its own sources and from one neighbour (router) if the second threshold (90%) is reached. In this case, it sends control messages to its nearest neighbours.

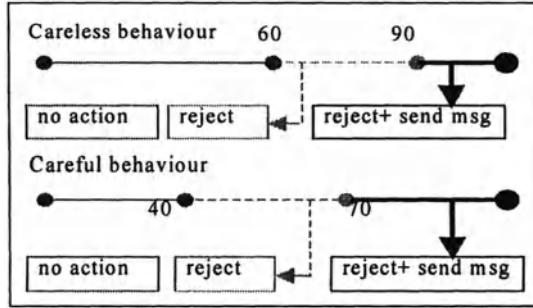


Figure 1. Careful and careless behaviours

These two behaviours have been implemented and tested in order to demonstrate the benefit of the concept of behaviour in the network management. Examples of other behaviours that have been defined but not yet implemented are given in the following:

- **Faithful**: is a node that never changes its routing tables; when an **Unfaithful** node will change it according to the current state of the environment;
- **Nationalist**: if node A's buffers are $\lambda\%$ occupied, then it rejects packets from the source which behaves in a different way than node A. The priority is given to packets coming from nodes having the same behaviour than A;
- **Self-seeking**: is the opposite of the nationalist behaviour. In fact, this behaviour can be applied if there is a hierarchy between behaviours and it will consist to reject packets coming from nodes having behaviours that belong to a less important class;
- **Thrifty**: is an important behaviour because it avoids the existence of useless packets. A is thrifty if it does not keep useless packets in its buffers. In fact, if a packet P that must be routed to a node B has λ as a lifetime, and the minimum necessary time to achieve B is $\gamma > \lambda$, then A will kill P. We can imagine also that when a Premium packet is lost the node can reject packets belonging to the same flow;
- **Equitable**: is a node that manages its buffers with equality vs *inequitable* node that starts to process the one with the highest priority.

4.4 Coexistence of behaviours

Including behavioural entities in the network is beneficial because these entities can adapt (i.e. change) their behaviour to the current situation, taking into account the profile of their neighbours, in order to make the most convenient decision. But what can be more advantageous is the use of entities that have more than one behaviour that tends to fill the gap of an incomplete behaviour. In fact, each behaviour responds to a certain number of situations and conditions but not to the whole condition. Including a multitude of behaviours in the same node will make this one more watchful to the changes in its environment, and therefore more adaptive, improving thereby system performance.

4.5 Relations between behaviours

A node can adopt more than one behaviour at the same time; it may have both careful and faithful behaviours for example. But in order to be coherent, the node must not adopt contradictory behaviours simultaneously, and in case this should happen, it will be considered as a node failure.

The relations between the different behaviours can be:

- a) Contradictory: careful and careless behaviours are a good example of this relation. Two contradictory behaviours can not exist in the same node at the same time; if this case appears it will be considered as a failure node;
- b) Complementary: two behaviours are complementary if the existence of one of them is strengthened by the existence of the other. Thrifty behaviour for example is strengthened by careful or careless behaviour;
- c) Indifference: this relation exists between two behaviours that affect different parameters and are triggered by different factors. Nationalist and faithful are indifferent behaviours.

4.6 Change of behaviours

An entity changes its behaviour as a result of the following events:

- reception of a packet from a node which asks it to adopt a certain behaviour (distributed decision);
- change of the conditions of the current behaviour; the node must decide what is the most convenient behaviour to the current network conditions (local decision).

This entity can inform the other nodes of this change and especially the nodes lying in the nearest neighbourhood but it is not mandatory. The other nodes can reason on what they really know or on what they suppose to be

true. This behaviour change can be followed by other changes at the neighbourhood level and even at the entire network level.

5. SIMULATIONS

The aim of the present simulations is to measure the benefits introduced by the behaviour concept in a node in terms of loss of Premium or Olympic packets. There are of course other parameters that should be optimised like the response time, the confidence interval, etc. According to the network environment and the chosen parameter to optimise, one of the different tested configurations fits best the current conditions.

The simulations are realised in a multi-agent platform called Swarm [17], that provides a framework to simulate a large set of complex problems. Swarm is a multi-agent, discrete event simulation, domain independent, that is used in different projects. In the Swarm system, the basic unit of simulation is the *swarm*, a collection of agents executing a schedule of actions. Swarm provides object libraries of reusable components to build models and analyse, display and control experiments on those models.

5.1 Parameters of the simulations

Our network topology (figure 2) is composed of six routers (1 to 6) and eight clients (sources: 1a, 1b, 2a, 3a, 3b, 4a, 5a, 6a). A screen catch of that topology is shown figure 3. The simulation environment is a two-dimension grid, on which we place the network's elements. Two kinds of entities are represented: agents and objects.

5.1.1 Objects

The different objects of the simulation are: links (between two nodes), packets (which move autonomously according to the decision taken by the node; a packet has no goal and does not respond to any stimulus), queues or buffers (contain packets waiting to be treated by the node).

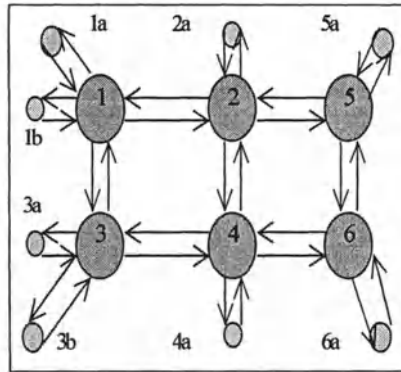


Figure 2. Topology

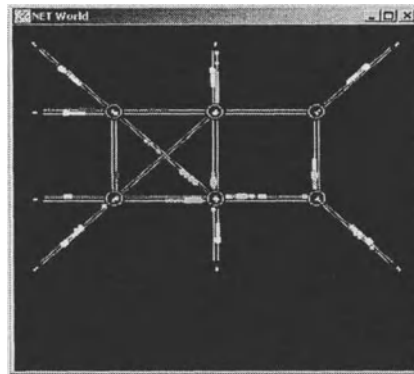


Figure 3. Network topology on screen

5.1.2 Agents

We have defined the following classes of agents:

- Service agent: processes packets lying in the node's buffer. There are two kinds of these agents: router service agent and client service agent. The first one sends packets to other nodes according to the routing table and their destination, while the second one is responsible of flow generation;
- Queue manager: processes the packet arrival and places them in the queue according to the current behaviour (taking into account the type of the packet when the behaviour is selective, careful or careless, or without caring about the priorities of packets (FIFO behaviour, C.f. 5.2);
- Master agent: monitors service agents and queue managers;
- Node agent: is the association of a service agent, a master agent and several queue managers. It represents the entity in its environment.

Each node possesses a static routing table. Packets are sent by a client to another client. The time between two flows is 200 simulation steps. We simulate the different flows in that proportion: 50% Best Effort flow, 30% Olympic flow and 20% Premium flow. In the future we will adopt a 20% Best Effort flow, 60% Olympic flow and 20% Premium flow proportion which will be more realistic. Our choice here was to prove that no behaviour will lead to an important loss of Premium and Olympic packets even if they do not constitute an important percentage of the whole packets in the system.

The size of each queue depends on the number of input links (number of input links *20), so queues of nodes 1, 2, 3 and 4 have 100 places, where queues of nodes 5 and 6 have 60 places.

5.2 Implemented behaviours

Four behaviours have been implemented in the current simulations. A node can have one of the following behaviours:

- **FIFO**: in that case, the node processes the packets in a FIFO manner; it represents the current routers;
- **Selective**: the node places packets in the queue according to their priority, but when its buffer becomes full, it will reject all packets even the Premium and Olympic ones. Client agents have this behaviour by default;
- **Careful and careless**: have been explained above (see 4.3).

The two last behaviours are different from the others by the fact that they never loose Premium packets. These two behaviours reject Best Effort packets in advance in order to gain time and buffer load. In fact, by starting to loose Best Effort packets before the buffer becomes full, we let more slots for Premium packets (and eventually Olympic ones) that should arrive later. An important gain of time can be reached when we avoid to treat Best Effort packets in the queue and to discard them (if the buffer is full) when Premium packets arrive.

5.3 Results

We have made simulations with several configurations in order to see the impact of the use of each behaviour on the node performances, and consequently the global system. In this paper, we will focus on seven configurations that will illustrate the main results that we got. These results are given in Table 1 (I: careless, P: careful, F: FIFO, S: selective)

When all the nodes have queues which proceed in a FIFO manner (we will call that nodes FIFO nodes) (figure 4), we have the worst results in term

of global loss (35,16 %, Table 1). This result is decomposed into 42,66% of Best Effort lost, 34,37% of Olympic lost and 22,97 of Premium lost.

In fact, by using selective nodes (figure 5), the global loss drops from 35,16% with the FIFO nodes to 30,55% with the selective ones. But we cannot be satisfied with these results because the percentage of Premium and Olympic lost packets remain important (Olympic packets represent 35,61% of the lost packets, and the Premium ones 25%).

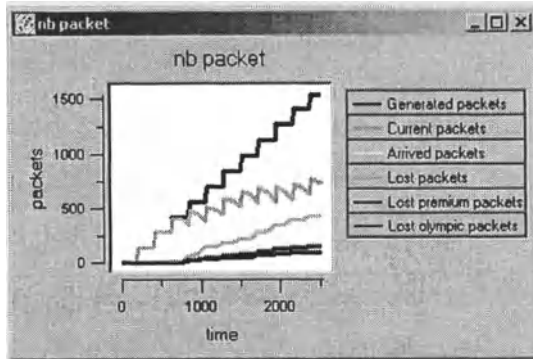


Figure 4. Configuration with only FIFO nodes

Table 1. Results of some simulations

	Global loss (GL) in %	Best Effort loss/GL	Olympic loss/GL	Premium loss/GL
FFF FFF	35,16	42,66	34,37	22,97
SSS SSS	30,55	39,39	35,61	25
III III	29,17	98,87	1,13	0
PPP PPP	31,48	100	0	0
SPI ISP	8,44	92,31	3,84	3,85
SPI SPI	8,44	38,46	40,77	20,77
ISP ISP	10,06	100	0	0

Configurations with careless nodes only (figure 6) is better than the two previous ones, because we loose only 1,13% of Olympic packets, the reminder of loss concerns Best Effort packets. This is due to the fact that when a node has this behaviour, it rejects Best Effort packets before that the buffer becomes full, giving more space to Premium or Olympic packets that should arrive. The loss of Olympic packets occur when the queue is only composed of Premium and Olympic packets, and in this case the Queue

Manager is obliged to remove one Olympic packet from the queue in order to place the arrived Premium packet. This lost remains insignificant.

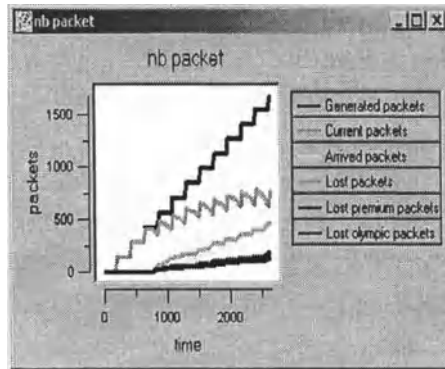


Figure 5. Selective nodes only

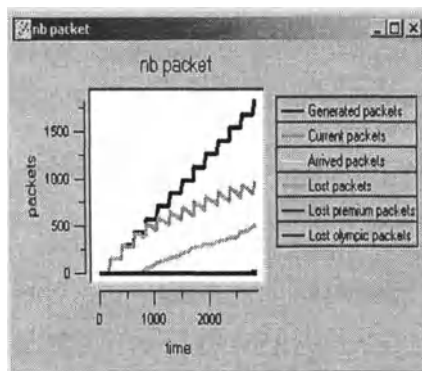


Figure 6. Careless nodes only

Better results are achieved when all the nodes are careful (figure 7). In fact, in that case, we do not loose Premium or Olympic packets, because we start to reject Best Effort packets from low thresholds of buffer load in order to let space for Premium or Olympic packets.

A configuration with the three behaviours (careless, careful and selective) confirms the results obtained above (figure 8).

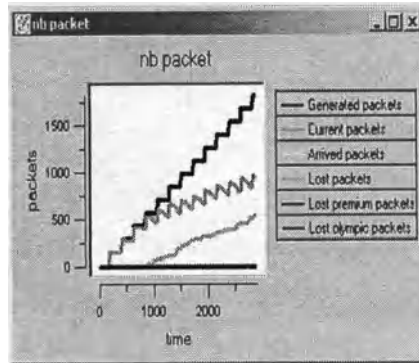


Figure 7. Careful nodes only

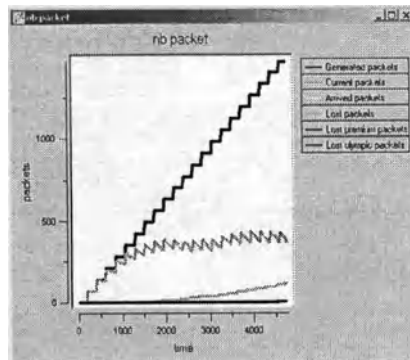


Figure 8. 2 S, 2 I and 2 P nodes

In fact, node 1 which is selective here (Table 1 row 5, figure 9) loses Premium and Olympic packets, whereas the node 3 (careless node) is responsible of the majority of Best Effort packets' loss. The number of lost packets depends on the node's behaviour and also on its position in the network. In fact, we notice that the nodes 1 and 3 are always those that cause the most loss because they are quite loaded (two sources) whereas the other nodes have only one source, and so less traffic (see figure 10 where we describe the node 4 in the same configuration than the node 1 in figure 9). This is clearly illustrated in figures 9, 10, 11, 12, 13 and 14.

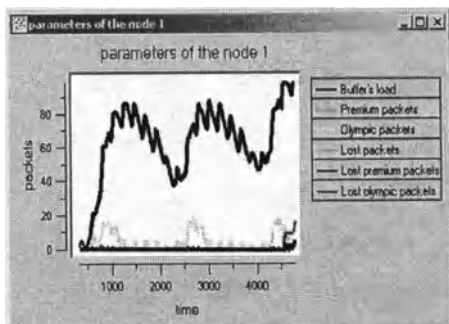


Figure 9. Node 1 in the (SPI/ISP) configuration

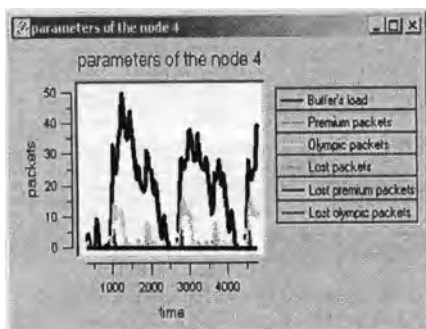


Figure 10. Node 4 in the (SPI/ISP) configuration

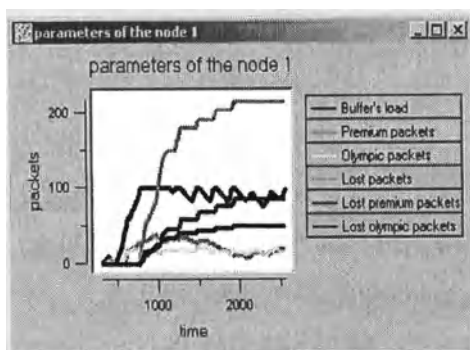


Figure 11. Node 1 in all FIFO configuration

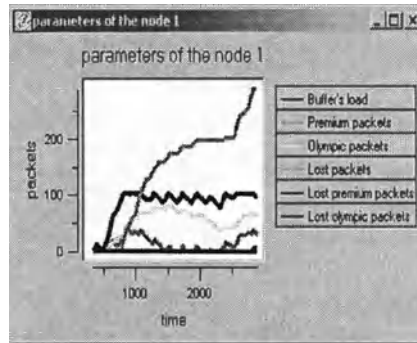


Figure 12. Node 1 in all careless configuration

The simulations show that the use of routers having behaviours gives better performance in terms of lost packets (Best Effort packets and especially Olympic and Premium ones), than when we use FIFO routers. This is due to the fact that we favour the processing of Premium and Olympic packets by putting them in the head of the queue, and moreover with careful and careless behaviours where we start to reject Best Effort packets before the buffer becomes full in order to let more space to Premium or Olympic packets that should arrive later.

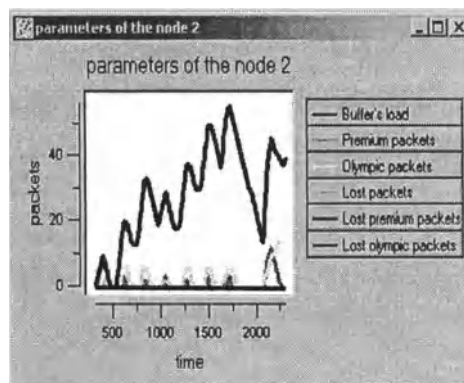


Figure 13. Node 2 in the all selective configuration

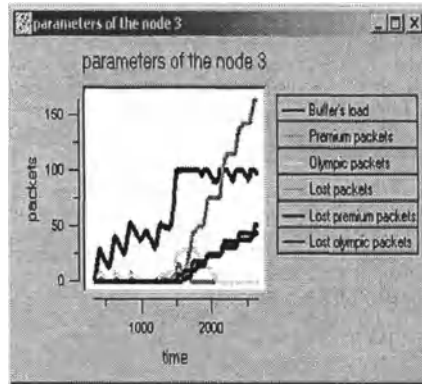


Figure 14. Node 3 in the all selective configuration

We proved through the different simulations that careful and careless behaviours represent better results than selective one concerning the loss of Premium or Olympic packets. Further simulations will be done by introducing the time parameter and by implementing and evaluating other behaviours, that should change in the light of the network change.

6. CONCLUSIONS

This paper presents a multi-agent simulation of an active network realised with the Swarm platform. Using agents in telecommunication network area is not new; many mobile agents platforms are used in many projects (for more details see [11]), but our approach is however innovative for three reasons: (1) the use of behaviour to model network entities, (2) the use of behaviour to simulate the dynamicity of the network, and (3) the use of simple concept inspired from the human model to name these entities. In this light, many behaviours have been defined, but only four of them have been implemented and tested in order to find the best combination regarding the number of lost packets (Best Effort, Olympic or Premium). The first results are really encouraging.

Adaptive agents are in a position to represent active entities, and will allow the system to get better performance, because each entity will be able to adopt the best behaviour regarding its local environment conditions, and a global equilibrium will result of the different interactions of these behaviours. This will be the next step of our research because it supposes cooperation and interaction between agents.

REFERENCES

- [1] Alexander D., et al.. The SwitchWare Active Network Architecture. IEEE Special Issue on Active and Controllable Networks, Vol. 12, N° 3, pp 29-36, 1998.
- [2] Barber S. and Martin C. Dynamic Adaptive Autonomy in Multiagent Systems: Representation and Justification. IJPR & AI, Vol. 15, N° 3, pp 405-433, 2001.
- [3] Bazzan A.L.C, Wahle J and Klügl F. Agents in Traffic Modelling - From Reactive to Social Behaviour. KI'99, Bonn, Germany, LNAI 1701, pp 303-307 September 1999.
- [4] Bensaid L. et Bouron T. Multi-agent Simulation of Consumer Behaviours in a Competitive Market. Pre-proceedings of MAAMAW'2001.
- [5] Doran J. Agent-Based Modelling of EcoSystems for Sustainable Resource Management. EASSS'01, Prague, Czech Republic, LNAI 2086, pp 383-403, July 2001.
- [6] Drogoul A., et al. MANTA : New experimental results on the emergence of (artificial) ant societies".in Artificial Societies, Nigel G. & R. Conte (Eds), UCL Press, London, 1995.
- [7] Guessoum Z., Cardon A. and Ramdani A. Toward Self- Adaptive Multi-Agent Systems. MAAMAW'99, Spain, June 1999.
- [8] Lazar, A., Campbell, A., Schulzine, H., Stadler R., Building Open Programmable Multi media Networks, Computer Communications Journal, Vol. 21 No. 8, pp. 758-770, 1998.
- [9] Loughran J. Island of Complexity Models and Adaptive Agent Simulations. http://www.sisostds.org/webletter/viso/iss_39/art_156.htm
- [10] Moukas A., Chandrinos K. and Maes P. Trafficopter: A Distributed Collection System for Traffic Information, CIA'98, Paris, France, LNAI 1435 pp 34-43, July 1998.
- [11] Perdikeas M.K., et al. Mobile Agents Standards and Available Platforms. Computer Networks Journal, Elsevier Publisher, Holland, vol. 31, n° 19, pp 1999-2016, 1999.
- [12] Schwartz B., et al.. Smart Packets for Active Networks. OpenArch'99, March 1999
- [13] Tennenhouse D.L., et al. A Survey of Active Network Research. IEEE Communications Magazine, volume 35, n°1, pp 80-86, January 1997.
- [14] Thibault S., Marant J. and Muller J. Adapting Distributed Application Using Extensible Networks. IEEE ICDCS'99, Austin, TX, June 1999.
- [15] Wahle J., Bazzan A.L.C, Klügl F. and Schreckenberg M. Anticipatory Traffic Forecast Using Multi-Agent Techniques. <http://www.traffic.uni-duisburg.de/>

[16] Wetherall D., Gutttag J. and Tennehouse D. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. IEEE OpenArch'98.

[17] <http://www.swarm.org>

[18] <http://www.ietf.org>

A Service Deployment Architecture for Heterogenous Active Networks Nodes

Matthias Bossardt¹⁾, Lukas Ruf¹⁾, Bernhard Plattner¹⁾, and Rolf Stadler²⁾

¹⁾*Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland*
{bossardt,ruf,plattner}@tik.ee.ethz.ch

²⁾*Center for Telecommunications Research, Columbia University, New York, USA,*
stadler@ctr.columbia.edu

Abstract In order to realise service deployment on high-performance active nodes, the problem of installing and configuring software components in complex, heterogeneous node environments must be addressed. The paper presents our approach to this problem, called Chameleon. The service specification is kept independent of any particular node architecture. During the service deployment phase, the service specification is resolved recursively on each node offering the service and is driven by node-specific parameters. The result of this resolution is a tree of service components, which can differ among different types of nodes. Our solution allows a service to take full advantage of specific node features, such as those related to performance or security. The design is illustrated using a video scaling service.

Keywords: Active networks, service deployment

1. INTRODUCTION

Service deployment on active network nodes includes installing and configuring software components that perform processing in the data path. Service deployment is difficult on high-performance nodes due to their complex architectures. They are often based on multiprocessors, run service

components as kernel modules [1] and provide multiple concurrent execution environments.

Many types of execution environments have been developed each of which is usually optimised for a certain type of tasks. Active nodes typically provide more than one to support the whole spectrum of services. Also, different types of active nodes usually support different sets of execution environments. All this motivates us to develop a service deployment scheme that can cope with heterogeneous active nodes, i.e., with active nodes running sets of execution environments that can vary from node to node. Our goal is to develop a scheme, where the service specification is node-independent, but service deployment and installation recognise the diversity of execution environments, thereby allowing a service to exploit the particular functionality and performance features of active network nodes.

Our approach to service deployment on heterogeneous active network nodes, called *Chameleon*, has two important aspects. First, the service model we propose is based on components with two types of interfaces - a data flow interface for programming packet flows and a control interface for controlling and managing the service components. A service is structured as an arbitrary tree of such components. Second, the service specification is independent of any particular node architecture. During the service deployment phase, the service specification is resolved on each node offering the service. The resolution and service creation process depends on the locally available set of execution environments. This way, the service can take advantage of the specific node features.

This paper is organised as follows. Section 2 outlines the service model. It is illustrated using the example of a video scaling service. Section 3 presents our approach to local service resolution and creation. The previously presented video scaling service is used to illustrate the deployment mechanism. Section 4 summarises related work. In section 5, we draw our conclusions and discuss future work.

2. SERVICE MODEL FOR ACTIVE NETWORK NODES

With *Chameleon*, we propose a service model that alleviates the development of services, and at the same time takes advantage of the flexibility and versatility offered by active networks. Services are modelled such as to support a flexible deployment on heterogeneous nodes, which is the main contribution of *Chameleon*. Typically, a service can be seen as a distributed application in the network. An active node provides the platform on which part of the service is run. In this paper, we focus on the node level.

The network level aspects exceed the scope of this paper. This section introduces the service model. We motivate our approach by presenting the requirements.

2.1 Requirements

Service models have been developed for various network and system architectures. Most models target the telecom environment [13], some of them end systems [2]. Similar to the work described in this paper, [7,9,12] feature models that focus on the node level. To a certain extent, they all allow for a recursive composition of services. However, to the best of our knowledge, they do not completely support our requirements as summarised below. For a more detailed discussion of the requirements, the reader is referred to [3].

2.1.1 Separating the Data Flow Interface from the Control Interface

The task of a router in traditional networks is to forward packets. An incoming packet traverses several processing stages before it leaves the router. An active node is the equivalent of a router in active networks. Packet forwarding remains the main task of an active node. However, the distinctive feature of an active node is that packet processing in the node is freely programmable. An active node allows programming both the interconnection and configuration of processing components, as well as the actual processing in the components. Because of the flow-based nature of packet processing in active nodes, we introduce an explicit data flow interface. Several benefits originate from a service model that supports such an interface. First, an explicit data flow interface supports representation of an active service as a directed graph, where the vertices correspond to packet processing components and where the edges represent packet flows. Data flow diagrams are an intuitive way to model a network service. Second, a data flow interface, which is separated from a control interface, facilitates the mapping of the service specification into an efficient implementation by mapping the data flow onto high performance communication facilities of the active node. An example for such a feature is described in [14], which proposes a zero-copy facility for packet processing.

The control interface allows controlling and managing a service component. It must provide a generic way to exchange control and monitoring information among processing components. The implementation of this interface depends on the specific execution environment and is also addressed by standardisation efforts [12].

2.1.2 Support of Different Active Node Environments

An active node may provide more than one execution environment. This is motivated by the following facts. First, each execution environment is usually optimised for a certain type of tasks. Some of them are rich in functionality, others are more restricted, but offer better security or better performance for certain tasks. Second, certain processing components are implemented for one specific execution environment. Finally, it is not realistic to assume that all active nodes provide the same set of execution environments [6], we require a service model to enable service specification that is independent of the node environment.

2.1.3 Support for Services Spanning Several Execution Environments

An optimum mapping of a service specification onto the node environment may result in an implementation that spans several execution environments. A packet classifier, for example, which processes packet at link speed, would be mapped to reconfigurable hardware execution environment [8], whereas a component that performs a routing algorithm, e.g. RIP, is more likely to be implemented in a CORBA execution environment. As a consequence, the service model must include abstractions for communication facilities between the service components. These communication facilities are execution environment specific. Moreover, the node may provide facilities that allow for communication between components in different execution environments.

2.2 Node Independent Service Model

This paper advocates a service model that is specialised for active networking. In order to fulfil the requirements from the preceding section, we need abstractions to model a service in a node independent way. We base the model on two basic abstractions, namely *composable containers* and *connectors*.

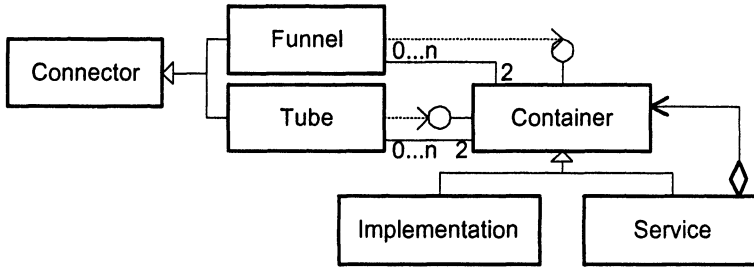


Figure 1. UML representation of the Chameleon service model

2.2.1 Composable Containers

We apply the *composition design pattern* [11], which allows a service to be structured in a hierarchical and recursive way. The UML diagram in Figure 1 shows that a *service* is composed of *containers*. We notice further that a *container* is a generalisation of a *service*, as well as of an *implementation*. A *container* defines the interfaces, which are common to both *service* and *implementations*. As a consequence, services and implementations can be handled the same way. A new service can be specified by composing existing services and implementations. An implementation, however, is not composed. It represents the primitive object of the composition.

As a result of using the composition design pattern, a service instance can be represented as a tree of dependencies, which can be resolved recursively. The leaves of the tree are implementation objects, whereas the nodes are services.

An implementation represents both code modules that must be or are already installed in a specific execution environment, and services that are installed when the node is bootstrapped.

Since a container, is a generalisation of both a service and an implementation, it is up to the local mapping process at the active nodes to determine whether a specific container representing an implementation or a service. The resulting service tree may therefore be different for each node environment. We exploit this property when mapping a service description onto different node environments (see section 3).

2.2.2 Connectors

Figure 1 shows that *funnels* and *tubes* are a specialisation of a connector. Both are an abstraction for communication facilities between containers. Tubes bind two data flow interfaces, whereas funnels allow communication between control interfaces.

A tube transports packets and is unidirectional. Both end points of a tube have the same capabilities. One end point sends packets, whereas the other receives them.

A funnel, in contrast, binds two control interfaces of, possibly, different type. As a consequence a funnel is a more complex abstraction. Attributes of an *implementation's* control interfaces specify the implemented API. Based on this information the active node may provide adapters to enable communication between different types of control interfaces. It is assumed that control APIs specified by different standardisation efforts may coexist in this way. We do not assume, however, that for each possible pair of control API standards an adapter can be provided.

In general, implementations of tubes and funnels are execution environment specific. The node, however, provides adapters and communication facilities that allow for an implementation of tubes and funnels across execution environment boundaries.

2.3 Modelling a Video Scaling Service

The goal of this section is to illustrate the applicability of our service model. We model a video scaling service as described in [4]. The service is based on a hierarchical encoding scheme, named WaveVideo [5]. The service measures the output queue length (congestion) of the WaveVideo stream. Based on this measurement, it selectively drops packets of the stream in order to adapt to the available bandwidth and to achieve a graceful degradation of the video quality.

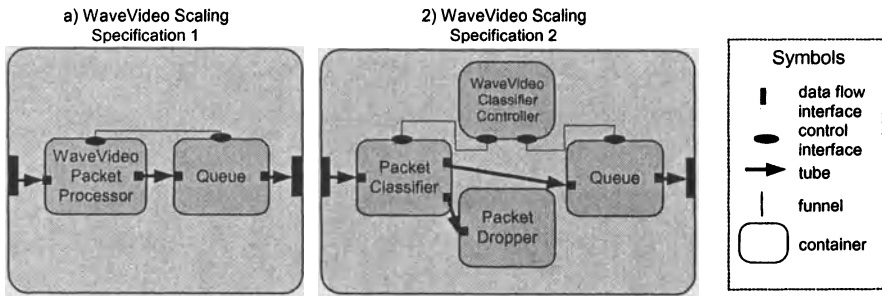


Figure 2. Visual representation of two specifications of the same service

Figure 2 shows a visual representation of two possible WaveVideo Scaling service specifications that may be registered in appropriate registries within the network. Specification b) contains more details than a), but both describe the same behaviour. Being more generic, specification a) allows for

more freedom during the mapping process on the node. We come back to this example in section 3.

3. SERVICE DEPLOYMENT ON ACTIVE NETWORK NODES

Service deployment in a network takes place on two levels: 1) on the network level, where network nodes are identified that will run the service, and 2) on the node level, where software is deployed within the node environment. In this paper, we investigate specifically service deployment on the node level. Service deployment on the network level is left as future work.

In traditional networks, a service is specified and deployed by a centralised entity, e.g. the network manager. In active networks, however, users, if permitted, or services themselves can create service specifications and request the deployment of new services. We will refer to the entity that requests a service as *service requester*.

A service request includes a specification that contains references to sub-services and a map of their interconnection, as shown in Figure 2. In our implementation, we use an XML document and a corresponding XML schema to represent such a service specification.

We consider a scenario where the service request is sent to a particular network node. This request contains a specification of the service to be installed locally. Such service specifications can be transported to a network node using a file transfer protocol, mobile agents, or encapsulating the specification in active packets.

The remainder of this section describes our approach to a local service creation architecture, and the mechanism that maps a service specification to the local node environment.

3.1 The Local Service Creation Engine

On receiving a service request, the local service creation engine (see Figure 3) maps the request onto the local node environment. In the following, we describe the components of the local service creation engine in detail.

The **specification processor** is the heart of the local service creation engine. It controls the mapping process. The input to the specification processor is the service specification. When the mapping process terminates, an implementation map is generated. The implementation map is referring to

implementation objects (see Figure 1) that must be installed on the node and shows their interconnection.

The **specification parser** receives a service specification from the *specification processor*. The service specification is read and checked on syntactical correctness. Its output is a one level tree representation of the service specification.

The **local service registry** is used to lookup service specifications. The specification processor passes (sub-)service name and, optionally, service attributes to the local service registry. The local service registry matches the request with registered services or consults an external service registry. It subsequently returns a list of matching (sub-)service specifications to the specification processor. The local service registry contains also entries for services that are preinstalled in the execution environments when bootstrapping the active node.

The **mapping policy** contains rules that influence the decision process of the *specification processor* when selecting a service specification from the list generated by the local service registry. E.g. a policy might indicate that implementations for execution environment 1 are always preferred to those for execution environment 2.

The **node information base** contains information about the configuration of the active network node. The information includes the types of available execution environments on the active node. Furthermore, the node information base contains information about the types of funnels and tubes available. The node information base is part of the local management information base (MIB).

The **deployment engine** receives an implementation map from the specification processor and generates a specific implementation map for each execution environment. The execution environment specific implementation maps contain information about the code modules to be installed and configured, as well as their interconnection. If required, tube and funnel adapters, which allow for communication across execution environment boundaries, are inserted in these maps.

The **code fetcher** receives a list of code modules that are required to be installed on the node. The list is compiled by the *deployment engine* after the service mapping process. The code fetcher contacts the appropriate code repositories to download the code modules.

The **adapter repository** contains funnel and tube adapters, which may be requested by the deployment engine.

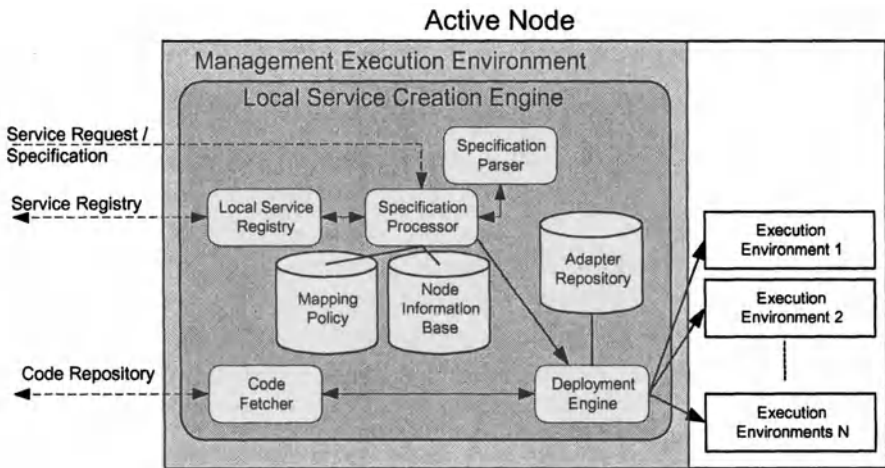


Figure 3. Active node with local service creation engine

3.2 Mapping the Service Specification onto the Node Environment

The mapping of the service specification onto the node environment involves two phases. In the first phase, a service tree is *recursively* built, the leaves of which contain implementation objects. An implementation map containing these implementation objects and the description of their interconnection is passed to the deployment engine. In the second phase, the deployment engine builds a specific implementation map for each execution environment, triggers the fetching of the required code and the installation in the execution environments. In the following sections, we describe the mapping process in more detail.

3.2.1 Building the Service Tree

The process that builds the service tree is controlled by the *specification processor*.

Step 1. The *specification processor* receives a service specification from the service requester.

Step 2. The *specification processor* calls the *specification parser*, which transforms the service specification into a tree representation. The leaves of the tree are the sub-services the service is composed of. The tree is returned to the specification processor.

Step 3. The specification processor uses the *local service registry* to lookup specifications matching the (sub-)services referred by the leaves of the tree. As a result a list of service specifications for each leaf node is

passed to the *specification processor*. If a service specification points to an implementation, information about the location(s) where the code module can be retrieved from or a reference to a preinstalled service is returned. In the second case, in which a service specification points to (sub-)services, (sub-)service names and, optionally, attributes are returned.

Step 4. For each leaf of the service tree, the *specification processor* decides, which (sub-)service or implementation to insert in the tree. The decision process is based on information from the *mapping policy* and the *node information base*. If a leaf is mapped to an implementation, it must not be considered anymore in the remaining steps of building the service tree.

Step 5. If all leaves are mapped to implementations, the tree building process terminates. Otherwise steps 2--4 are repeated for the remaining leaves, which point to (sub-)services, in order to recursively build the service tree.

Step 6. The *specification processor* generates an implementation map and passes it to the *deployment engine*.

3.2.2 Installing the Mapped Service

The *deployment engine* controls the process of installing the mapped service.

Step 1. The *deployment engine* receives the implementation map from the *specification processor*. Based on this information, it constructs execution environment specific implementation maps, inserting references to tube and funnel adapters where required.

Step 2. The *deployment engine* contacts the *code fetcher* with a list of code modules and requests their download. It further retrieves the required tube and funnel adapters from the *adapter repository* and configures them.

Step 3. The *deployment engine* passes the specific installation maps and references to the code modules and adapters to the execution environments, where they are installed by execution environment specific mechanisms.

3.3 Mapping a Video Scaling Service

Applying the service specifications from Figure 2, we illustrate the mapping process as described above. An example in which the service is mapped onto two different node architectures can be found in [3].

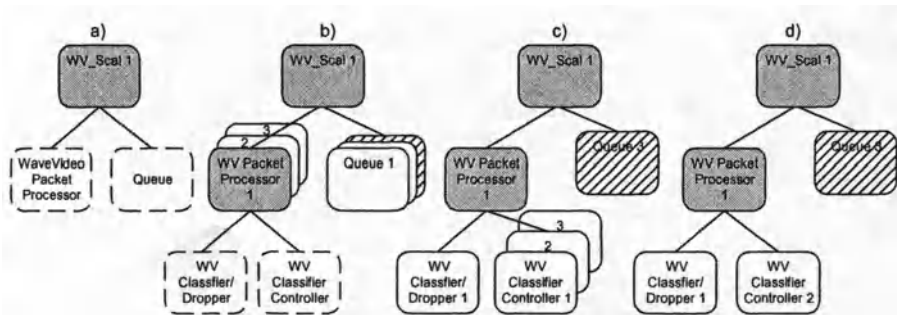


Figure 4. Four steps of resolving a WaveVideo Scaling service specification on a node

The active node gets a request to deploy a service according to the service specification from Figure 2a). Figure 4 illustrates this subsequent mapping process. The service specification is composed of two sub-services, as can be seen in Figure 4.1. The *container discovery* is used to lookup specifications of these sub-services. Three matching specifications of `WVPacketProcessor` as well as three specifications of `queue` are retrieved (cf. Figure 4.2). One `WVPacketProcessor` specification (grey box) points to two sub-services, whereas the remaining two (white boxes) point directly to implementations for specific execution environments. One `queue` instance is a preinstalled (sub-)service available on the node (hatched box), the other two are pointing to implementations as well. We notice in figure Figure 4.3 the *specification processor* decided - after consultation of the *node information base* and the *mapping policy* - to use `queue_3` and continues the recursive resolving process for the composed `WVPacketProcessor_1`. The resulting implementation consists in three implementation entities (cf. Figure 4.4). One is provided by the node (`Queue_3`), the other two instances are code modules (`WVClassifierDropper_1`, `WVClassifierController_2`) and must be downloaded by the *code fetcher*. The *deployment engine* requests the appropriate execution environments to install the code and binds the interfaces according to the implementation maps. The binding of the interfaces might require the insertion of adapters - abstracted by tubes and funnels - in order to cross execution environment boundaries.

4. RELATED WORK

Click Router [7] proposes a software architecture for configurable routers. Click Router can interpret a configuration file, and create and configure services. Services are composed of C++ elements that run in a kernel execution environment. Click Router assumes a direct mapping of the

element names listed in the configuration file to the ones instantiated on the node. All required elements are available from a node local code repository. Click Router does not support multiple, concurrent execution environments.

NetScript [9] uses a recursive mechanism for the composition of services from components. However, it does not deal with the service deployment process and does not support multiple, concurrent execution environments.

The *IEEE P1520 standards initiative* [12] aims at a standardised API to the control interface of network elements. In low level of the architecture, it uses a similar service composition mechanism. P1520, however, does not specify service deployment mechanisms.

TINA-C [13] specifies a network architecture for the telecommunication environment. Network services run on a Distributed Processing Environment (DPE). TINA-C computational objects provide an operational interface and a stream interface that allows separating the data flow from the control flow.

5. DISCUSSION AND FUTURE WORK

In this paper, we described a service model that facilitates service deployment on multi-execution environment nodes. The service specification is locally mapped to the node environment in order to exploit the specific features of the different execution environments. Moreover the service specification supports heterogeneous networks, because it is node independent. As a consequence, the same service specification can be sent to different types of active nodes.

The local service creation engine, which maps the service specification to the node environment, uses locally available information from the node information base and local policies for the mapping in order to take advantage of specialised execution environments provided by a node. The presented algorithm is greedy in the sense that on each level of building the service tree, a decision is made on which sub-service to use. Therefore, the algorithm may not find a globally optimum mapping. A simple extension, based on backtracking, would allow performing an exhaustive search of possible service mappings. We expect the mapping space to be relatively small. Therefore, we consider an exhaustive search to be a valid approach.

Future work will investigate the service deployment on a network level and its integration with the work proposed in this paper. Moreover, we will evaluate whether our assumption about the mapping space is valid. Also, we will extend and generalize the XML schema that we use in our prototype implementation for the specification of services. Furthermore, we plan to evaluate Chameleon within the testbed of the IST-FAIN project (5th EC

Framework Program) [10], which aims at developing an architecture for future high performance active networks.

ACKNOWLEDGEMENTS

We gratefully acknowledge the feedback we received on an earlier version of this paper and during discussions with the FAIN project members. This work is funded by ETH Zürich and Swiss BBW under grant number 99.0533. It is part of ETH's contribution to the IST Project FAIN (5th EC Framework Program).

REFERENCES

- [1] Decasper, D., Dittia, Z., Parulkar, G., Plattner, B. Router Plugins - A Software Architecture for Next Generation Routers. IEEE/ACM Transactions on Networking, February 2000.
- [2] Stiller, B., Class, C., Waldvogel, M., Caronni, G., Bauer, D. A Flexible Middleware for Multimedia Communication: Design, Implementation and Experience, IEEE Journal on Selected Areas in Communication, vol. 17 no. 9, September 1999.
- [3] Bossardt, M., Stadler, R. Service Deployment on High Performance Active Network Nodes. TIK Technical Report 122, ETH Zürich, Switzerland, 2001.
- [4] Keller, R., Choi, S., Decasper, D., Dasen, M., Fankhauser, G. and Plattner, B. An Active Router Architecture for Multicast Video Distribution. Infocom 2000, Tel Aviv, Israel, 2000.
- [5] Fankhauser, G., Dasen, M., Weiler, N., Plattner, B., Stiller, B. WaveVideo - An Integrated Approach to Adaptive Wireless Video. ACM Monet, Vol. 4 No. 4, 1999.
- [6] Haas, R., Droz, P., Stiller, B. Distributed Service Deployment over Programmable Networks. DSOM 2001, Nancy, France, 2001.
- [7] Kohler, E., Morris, R., Chen, B., Jannotti, J. and Kaashoek, M.F. The Click Modular Router. ACM Transactions on Computer Systems 18(3), August 2000, pages 263-297.
- [8] Lockwood, J.W., Naufel, N., Turner, J.S., and Taylor, D.E. Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX). ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001), Monterey, USA, February 2001.

- [9] Da Silva, S., Florissi, D., and Yemini, Y., Composing Active Services in NetScript, position paper, DARPA Active Networks Workshop, Tucson, AZ, March 9-10, 1998.
- [10] Galis, A., Plattner, B., Moeller, E., Laarhuis, J., Denazis, S., Guo, H., Klein, C., Serrat, J., Karetsos, G., Todd, C. A Flexible IP Active Networks Architecture. International Working Conference on Active Networks (IWAN 2000), Tokyo, Japan, October 2000.
- [11] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns. Addison-Wesley, 1995.
- [12] Vincente, J., Denazis, S., Biswas, J. L-interface Building Block API. IP Working Group IEEE P1520, www.ieee-pin.org, March 2001.
- [13] TINA-C. www.tinac.com.
- [14] Druschel, P., Peterson, L.L., Fbufs: A High-Bandwidth Cross-Domain Transfer Facility, ACM Symposium on Operating Systems Principles, Dec. 1993.

I

PLATFORMS

Driving Trends in Information and Communication Technology – Markets and Technologies

Reda Reda ¹⁾ and Harmen R. van As ²⁾

¹⁾ Siemens AG Austria, Vienna, Austria; ²⁾ Vienna University of Technology, Vienna, Austria

Abstract This paper presents an analytical study of the development of the information and communication technology (ICT) market from the marketing as well as from a technological point of view. The trends and their impact on the information communication networks and consequently smart networks are discussed. Despite the current global economic disaster in almost every single industry, there are still a lot of business opportunities within the ICT industry, e.g., revenue-generating new features, applications and value-added services, soft-switches, plug-and-play network elements, network security, e- and m-business, micro-payment systems, personalized blue-tooth bandwidth applications and web-based add-ons. Moreover, there are still enormous business chances to adapt, to optimize, and to exploit in the already existing network-infrastructures in Eastern Europe as well as in the Third World.

1. HARDWARE AND SOFTWARE EVOLUTION

To understand the ICT evolution and the corresponding market development and trends, we present a short survey of the hardware and software evolution during the last decade and a corresponding prognosis for the next decade. The physicist Niels Bohr once said: "Prediction is very difficult, especially about the future". The following examples display this fact very impressively. Tom Watson Sr., Chairman of IBM, said in 1943, "I think there is a world market for maybe five computers" and Irwin Dorros, Vice President of Long Lines ATT, stated in 1981, "ISDN will be ubiquitous by 1985". Despite the mentioned fact that predictions about the future are

indeed difficult, this section invites to have a look at possible future aspects concerning hardware and software.

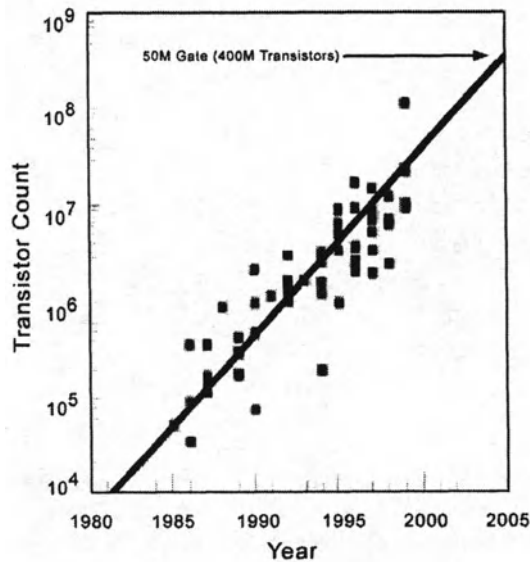


Figure 1. Trends in integrated circuit complexity

1.1 Hardware Evolution

Integration density and performance of integrated circuits have gone through an astounding revolution in the last couple of decades. In the 1960s it was predicted that the number of transistors that can be integrated on a single die would grow exponentially with time. This prediction, later called Moore's law, has proven to be amazingly visionary. Its validity is best illustrated by a set of graphs. Figure 1 plots the integration density of integrated circuits and Figure 2 displays the integration density of memory as a function of time. As can be observed, integration complexity doubles approximately every one to two years. As a result, memory density has increased by more than a thousand-fold since 1970. An intriguing case study is offered by the microprocessor. From its inception in the early seventies, the microprocessor has grown in performance and complexity at a steady and predictable pace (Figure 3 for Moore's law). Clock frequencies double every three years and have already reached into the GHz range. This is illustrated in Figure 4, which plots the microprocessor trends in terms of complexity and performance at the beginning of the 21st century. An

important observation is that, as of now, these trends have not shown any signs of a slow-down.

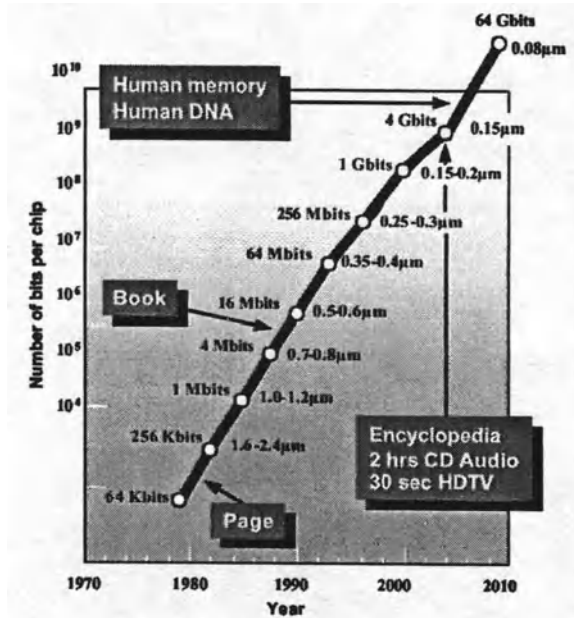


Figure 2. Trends in memory complexity

Beginning with Moore's law to the vanishing of today's personal computers, the following part tries to deliver an overview of various technological trends in the field of future hardware. The microprocessor performance will still double every 18 months, or even faster – more likely every 14 months. This is also known as Moore's Law. In addition, transmission capacity will even double every 12 months. Considering the Internet, its number of addresses will double every six months. To solve the problem of this ever-increasing demand of IP addresses IPv6 was introduced, through which it would be possible to address not only every single chip in a interplanetary Internet, but even every single atom in the universe. In this respect, Metcalfe's Law says that a network's usefulness, or value, goes up as the square of the number of its users, i.e., with only one user a network is no utility; thousand users form a corporate Local Area Network (LAN); America Online (AOL) has 29 million customers; and everyone accessing the Internet means 6 billion users. Most importantly, exponential growth means exponential benefit. Miniaturization and performance will be more and more the main driving factors for the hardware market. Other facts are that portable mobile devices will gain multimedia capabilities, that worldwide networking and supercomputing will

come along with a considerable growth in bandwidth, and that smart card applications such as multi-functional and programmable devices will evolve on the market. We will see an ascending integration of telephone, TV and personal computer together with mobility and complex network. Convergence as the buzzword will not only comprehend technologies or networks, but also every single industry and business. Today, 30 million chips are produced for PCs every year, whereas more than 250 million chips are built into other devices every year. The average American household comprises approximately 40 microprocessors giving an 'ambient intelligence', and familiar surroundings become an interface to a worldwide communication infrastructure. Today's PCs start to vanish.

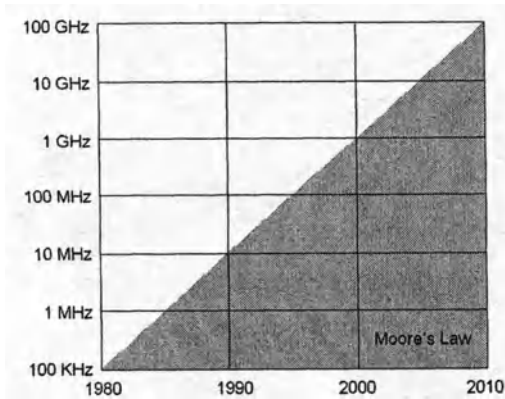


Figure 3. Microprocessor clock frequency: 100x per decade

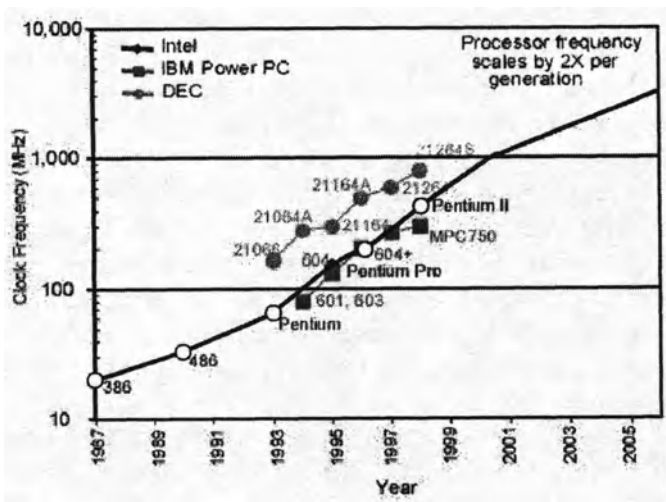


Figure 4. Trends in clock frequency

The examples of hard disk capacity and CPU speed illustrate these enormous achievements. The hard disk capacity increases by a factor 100 per decade, whereas access speed only increases by a factor 10 in the same period. This implies that the time to scan an entire hard disk increases by a factor 10 every decade. Figures 5 and 6 show the incredible development of microprocessors by means of different layout-designs from two Intel microprocessor generations.

1989	20 MB	\$650	1974	2 MHz
1994	345 MB	...	1985	16 MHz
1999	50 GB	...	1994	75 MHz
2000	60 GB	...	1999	700 MHz
2001	100 GB	\$250	2001	2000 MHz

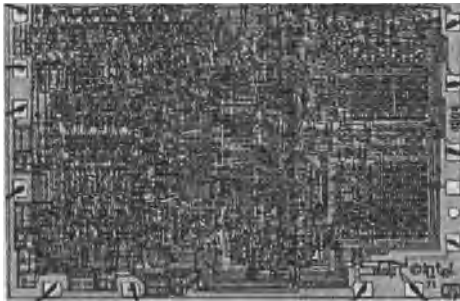


Figure 5. Intel 4004 microprocessor (1971, 108kHz, 4kB of address. memory)

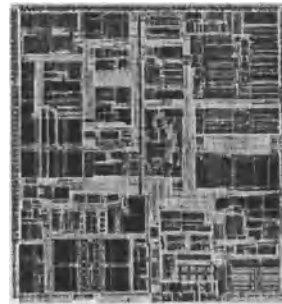


Figure 6. Intel Pentium II microprocessor (1997, 233MHz, 4GB of address. memory)

1.2 Software Trends

The following part gives a survey of some evolution milestones in the field of software.

Media: Digitization (and increase of processing power and memory size needed) of all different types of media leads to an enormous increase of information available and many different forms of direct electronic communication. Consumers will also digitize their own media such as photographs and video.

System architecture: Worldwide distributed systems allow to maintain information in a decentralized manner, enabled by layered and modular architectures, where hardware and communication specific details are

hidden. This reflects the demand for more efficient heterogeneous network management systems.

Standard Software: Operating systems, programming languages with graphical programming tools and databases tend to follow more and more a homogenous paradigm. In operating systems, we will see the survival of just two systems, UNIX and Windows. This also leads to an increased reusability of software modules and a better interoperability of systems.

Software process: The whole software industry benefits from the homogenous paradigm. New approaches such as participatory designs, workflow procedures, computer supported cooperative work, and reusable components are common now. But good software design takes time. The tools needed only improve in small steps because object-oriented software design is not achieving its promise. Software development costs increase. For these reasons, software projects take more time and cost more money, since documentation and maintenance is expensive, therefore quickly leading to legacy systems.

Applications: User-assisting applications are needed in an increasingly number of fields. But they have a higher time-to-market and competitive pressure leads to software, which is full of errors, decreasing quality and reliability. Modeling of human behavior and thinking is still a hard problem, creating the biggest gap between expectation and reality.

User interfaces: Great developments from character-based terminal to window-based interfaces (WYSIWYG – What You See Is What You Get) have been achieved, now followed by multi-modal interfaces, virtual-reality, and speech recognition. But interfaces are still a hot topic. It needs interdisciplinary approach, including psychology and sociology. Software must become fault-tolerant, self-repairing, remotely manageable, and remotely deployable.

Finally, Nathan's law says software is a gas. It expands to fill the volume available. Concluding, Figure 7 shows the well-known hardware/software life-cycle, which is the main driving factor for future software and hardware development.

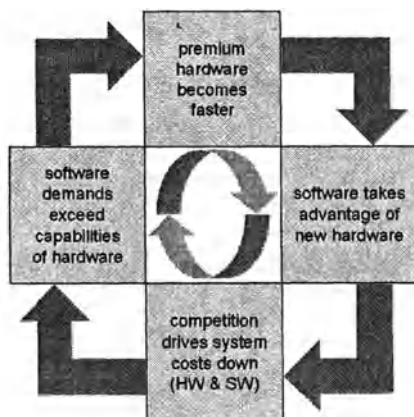


Figure 7. Hardware / software life-cycle

1.3 Hardware and Software Research Trends

Research in the field of hardware and software technology will supposedly focus on the following topics in the near future: interfaces, e.g., multilingual e-commerce applications, personalization, and natural interactivity; embedded systems; integrating and linking fixed and mobile communication; distributed computing; architectures for information integration; as well as privacy and security

1.4 Taking a Decade-Leap in Time

The following part takes into consideration the facts and predictions stated above and tries to predict what hardware and software of the next decade may look like. A personal computer of the next decade might have an 1,5 THz processor, 100 GByte of memory, a 10 TByte hard disk and a wall-sized display with 8192 x 8192 pixels. A Server of the next decade might consist of an interconnected system of 64 processors at 1,5 THz, 5 TByte of memory, a 1 PByte hard disk, and a 100-Gbit/s Ethernet network interface. Operating systems in the next decade might show that Microsoft could no longer afford to enhance Windows, because the complexity became overwhelming; companies like IBM, HP, and Sun have abandoned their own Unix for Linux with private extensions, since the cost of building an own operating system became too high; and devices have small, specialized operating systems.

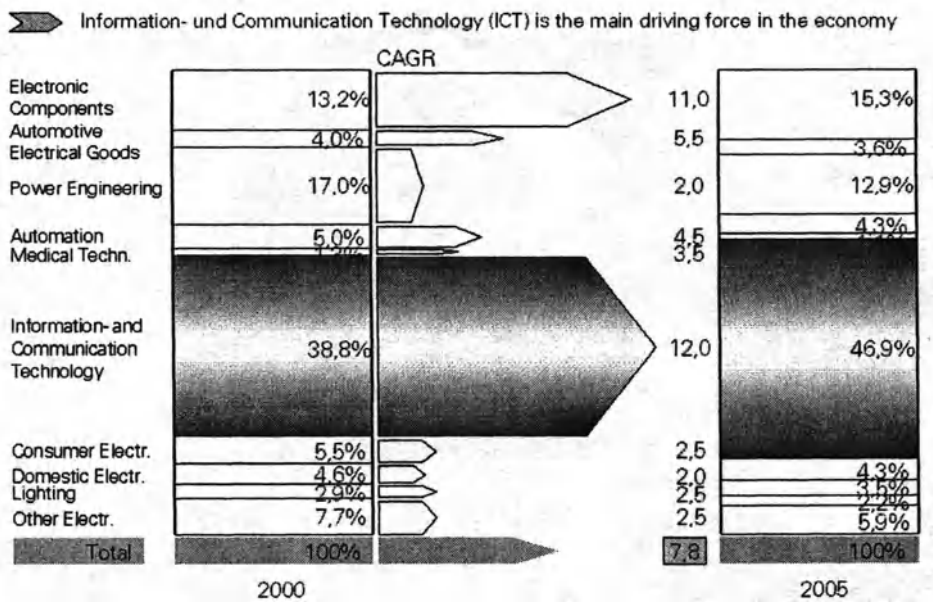


Figure 8. World electronics market – development 2000-2005

2. ICT MARKET POTENTIAL

The development of the information and communication technology (ICT) market over the recent years has been exceptional, perhaps even unique in the history of industrial change. According to recent studies, the ICT world market in 2001 is expected to exceed the 1 Billion Euro mark (approx. 1,106 Billion Euro), with a quite impressive combined annual growth rate (CAGR) of 12%. The ICT market represents almost 39% of the total world electrical market. Thus, this is the main driving force in the progress of the electrical industries. Figure 8 illustrates the breakdown of the total market of electrical equipment and gives a prognosis for the year 2005.

2.1 Worldwide ICT Market Trends

Considering the relative growth of the ICT market, a slow-down and uncertainty has been observed in the second half of 2001, but the European market growth is still higher than in the US and Japan, as can be derived from Figure 9. Furthermore there are still good ICT business chances in Eastern Europe and the Third World, despite the global economical crises

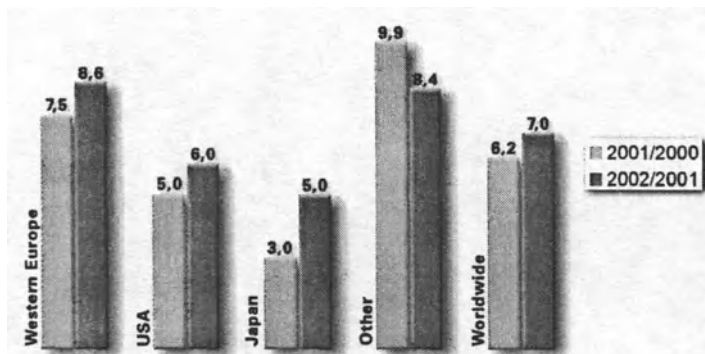


Figure 9. Worldwide IT market growth by region 2001-2002 (%)

The major factors effecting this slowdown are the global economic slowdown and the dotcom disappearance; the heavy 3G (Third Generation) license fees cutting or delaying carriers investment programs; delays in availability of GPRS and 3G, coupled with relative saturation of the GSM market; pricing and tariff competition squeezing revenues streams; low market acceptance of ASPs (Application Service Providers) and e-marketplace models; ongoing skill shortage hampering e-business development.

From the other side, the major factors effecting the ICT market growth are: from dotcom euphoria to real e-business for “brick-and-mortar” companies; increasing adoption of Internet-centric solutions for e-business; integrated services moving from back-office into front-office operations; unmetered Internet access favoring consumer and professional usage; growing mobile data expectations; broadband (ADSL and WLL) diffusion driving Internet and multimedia applications and services; e-government programs notably at local community level.

2.2 ICT Market Growth in Western Europe

The ICT market for Western Europe showed a serious slow-down in growth in 2001. This trend will continue for the telecommunications market in 2002, whereas the IT market will be able to slightly increase its growth. More interesting seems the fact, that in 2001 and 2002 the IT market overtakes the telecommunications market with respect to growth potential for the first time ever, as can be derived from Figure 10.

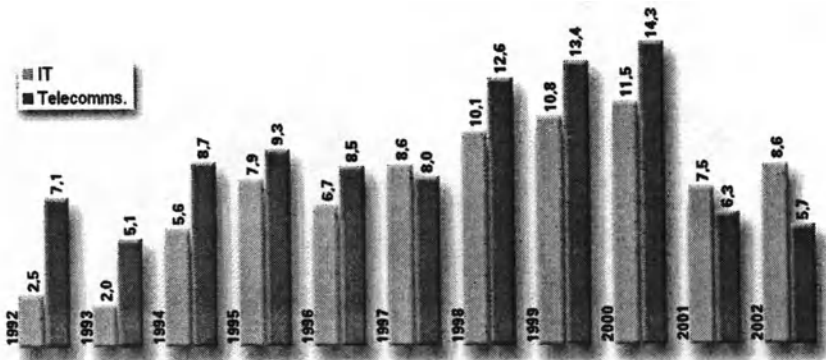


Figure 10. Western Europe ICT market: actual growth and forecast 1992-2002 (%)

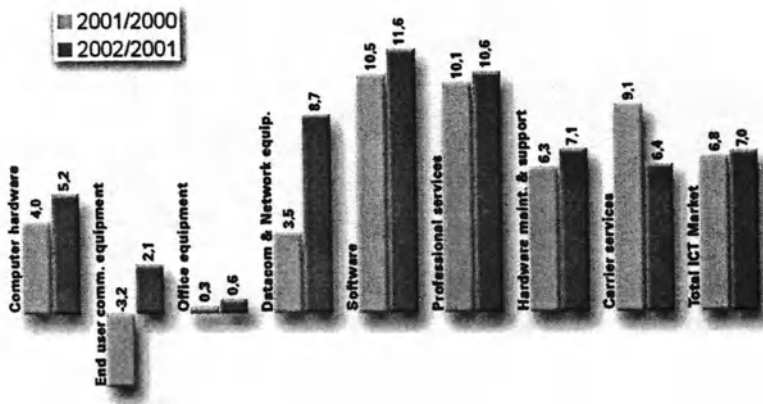


Figure 11. Western Europe ICT market growth by product segment 2001-2002 (%)

Dividing the Western European ICT market into its product segments shows some interesting facts (Figure 11): All product segments including computer hardware, network equipment, end user communications equipment (which really had a bad year in 2001) and software will be able to increase their market growth in the year 2002. But carrier services will decrease its growth by approximately 30% in 2002.

Figure 12 shows a daring look into the future of access platforms in European homes up to the year 2010. As expected the plain old telephone service (POTS) will loose its importance with regard to home Internet access and decrease to approximately 12% in the year 2010. ISDN access should vanish by the year 2005, mainly because of the tremendously increasing market share of DSL, Coax-Cables, Fibre-Hybrid and Fibre-Pure accesses. Finally, other technologies like satellite access or FWA will not gain importance at all and stay below 2% in 2010.

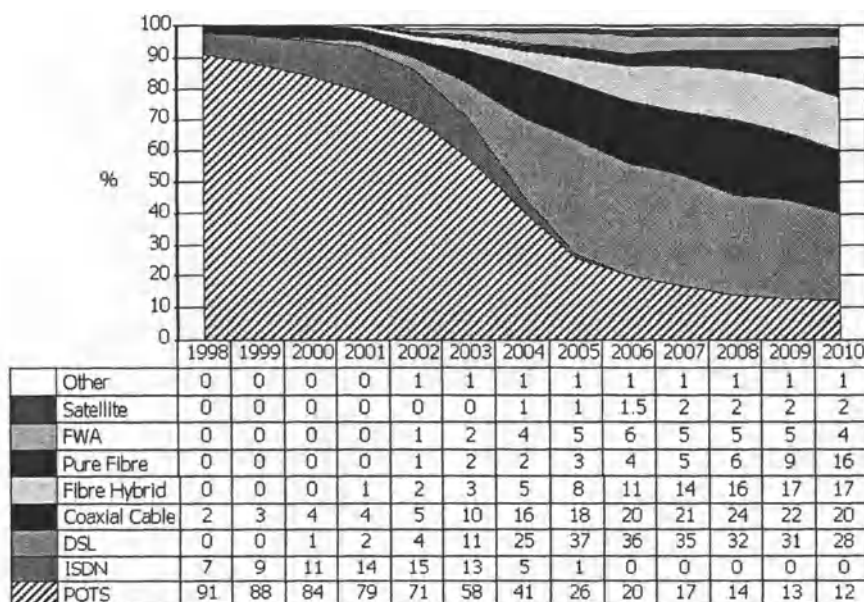
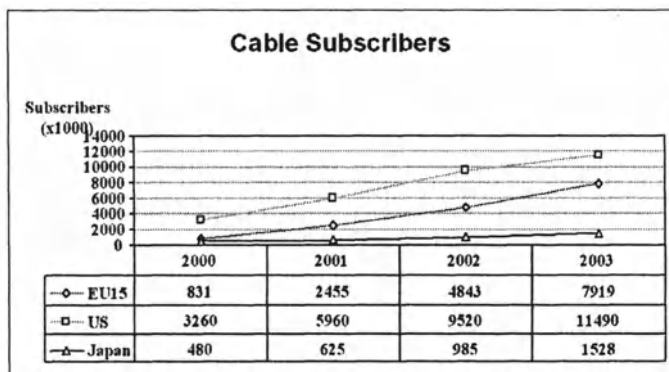


Figure 12. Predicted market share of access platforms to EU homes / small medium enterprises (SME)

The predicted growth of household broadband, DSL and broadband cable in the EU 15, US and Japan is plotted in Figure 13. Remarkable is the fact that Europe and Japan will not be able to make up the technological gap regarding broadband access in the near future. 22% of US households will possess broadband access, whereas just 11% of European homes will own a broadband access in 2003.



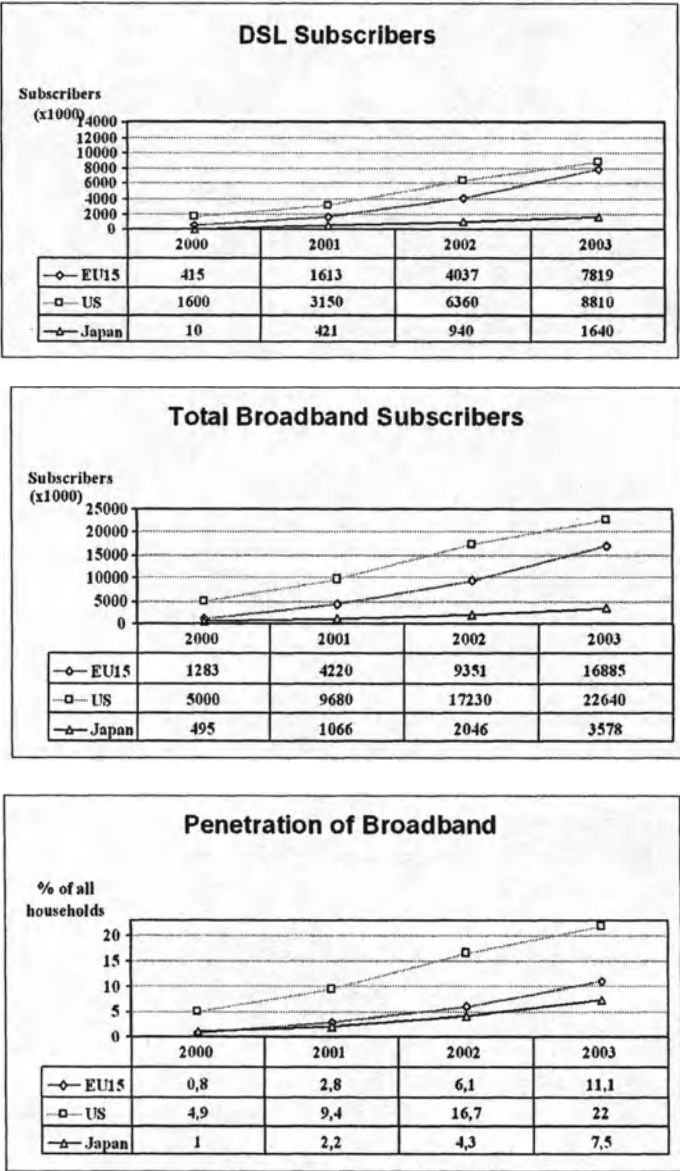


*Figure 13. Predicted growth of household / SME
Broadband, DSL and Broadband Cable in the EU 15, US and Japan*

3. THE E-BUSINESS IMPACT ON ICT

E-business is more than the latest rage in information technology, it is the new order. For the purpose of building a picture of the impact of e-business

on the ICT market and consequently on the global economy at macro- and micro-economic level, we first go through the market trends of e-business.

3.1 E-business Consolidation

After reaching its euphoric peak in mid 2000, e-business suffered a drastic downfall by the end of 2001. Moreover, the e-business industry will now enter a phase of massive problems and disillusionment. According to latest studies even 75% of all projects in the field of e-business may fall short of their expectations in the near future. This downfall will hit 'brick and mortar' as well as dotcom companies. However, the temporary crash of e-business does not implicate an absolute decline of e-commerce, e-procurement, or e-business in general. In fact, the exact opposite will occur. The reasons for the collapse of e-business projects in the next years are mainly immature business models, strategic errors, and implementation problems. The aftermath of this development will lead to 'real' e-business companies approximately in 2004. Between 2006 and 2008 the majority of enterprises will have completed their economic metamorphose to so-called 'brick and click' companies.

Figure 14 provides a snapshot of the inevitable cycle of hype and disillusionment that accompanies the technological path to e-business maturity. It consists of the following phases:

Technology trigger: A breakthrough, public demonstration, product launch, or other event generates significant press and industry interest. Quantum computing and other non-silicon computing techniques are still firmly in the research laboratories, but will represent the future of computing once today's chip manufacturing techniques hit the limits of physics. Audio mining will make previously untapped data sources (e.g., recorded meetings, phone calls, and conversations) available for search and analysis through speech-to-text technology. Synthetic characters will enable users to interact in natural language with a computer-generated personality to access online information and services, although today's characters are severely limited in scope. Digital ink, which allows dots of ink to be flipped like screen pixels, will convert a broad range of surfaces (e.g., packaging and signs) into rewriteable displays. Biometric identification, which uses physical characteristics (e.g., face and voice) for authentication, is being boosted by Microsoft's adoption of the BAPI standard. Voice portals, which provide automated speech access over the phone to information and messaging

services, will explode onto the market during the next year, with hundreds of new, typically free, branded offerings.

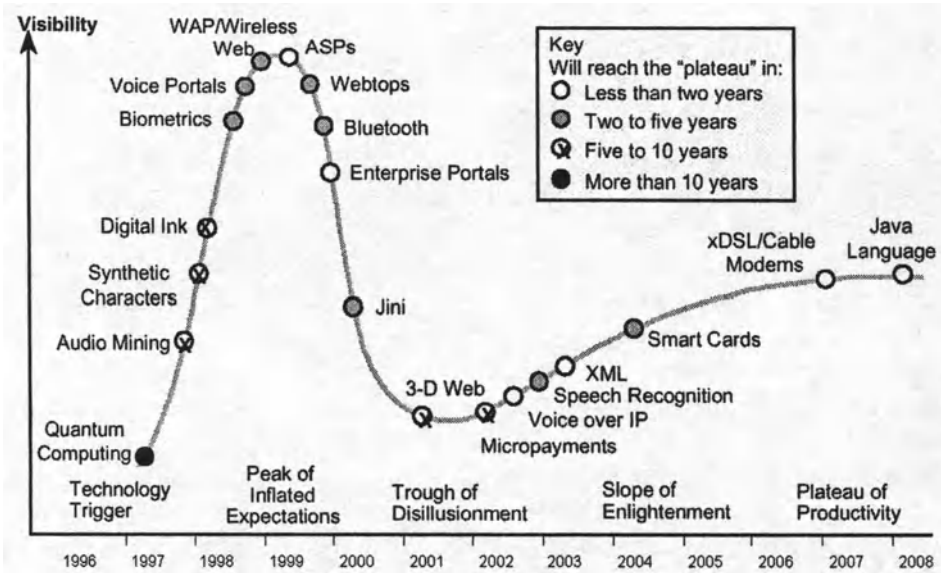


Figure 14. E-business hype-cycle

Peak of inflated expectations: During this phase of over-enthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes but more failures as the technology is pushed to its limits. The only enterprises making money are conference organizers and magazine publishers. Wireless Web/WAP phones are available from leading vendors as a means of accessing Web information from mobile devices. ASPs, which outsource the development and hosting of application software and hardware, are springing up in every area of IT, from SAP to speech recognition. Webtops are virtual desktops resident on the network that give users access to their information and applications from wherever they gain access to the Internet.

Trough of disillusionment: Because the technology does not live up to its over-inflated expectations, it rapidly becomes unfashionable and the press abandons the topic. Bluetooth chips are beginning the difficult path toward mass adoption. Enterprise portals have combined personalization, search and document management capabilities, but offer fundamentally nothing new. Jini, which promises true plug-and-play for computing peripherals, still needs to be translated into meaningful products. 3-D Web technologies for visualizing and exploring objects and spaces are still hampered by bandwidth and a shortage of compelling applications.

Slope of enlightenment: Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the technology's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools become available to ease the development process. Climbing the Slope. Micropayments are overcoming the disillusionment surrounding e-wallet schemes and re-emerging in new forms, such as billing to existing payment aggregators (e.g., phone companies and ISPs). Voice over IP is reaching levels of quality needed for applications such as integrated phone/Web contact centers. Speech recognition is providing significant paybacks in automating telephone transactions (but is not widely accepted on the corporate desktop). XML is broadly accepted as a standard in Web products for data and content exchange. Smart card technology is well-established in some countries for mobile phone, prepayment and bank card applications, although trailing in the United States. Cable modems and xDSL are well established as viable options for high-bandwidth connectivity.

Plateau of productivity: The real-world benefits of the technology are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generation. The final height of the plateau varies according to whether the technology is broadly applicable or benefits only a niche market.

Success on the long run can only be assured by a strictly integrative e-business concept. Thus, enterprises should be selectively aggressive in the early adoption of technologies with the potential for high impact, but should ride out the hype-cycle for less-critical technologies.

3.2 Top E-Business Trends

E-business market research predicts a rather balanced growth for the next two years, whereas an optimal growth rate is expected thereafter. Figure 15 presents the market development of e-business until the year 2004. The driving force of the e-business evolution is the Business-to-Business (B2B) sector, which is expected to exceed the 4.000 Billion Euro limit by 2004.

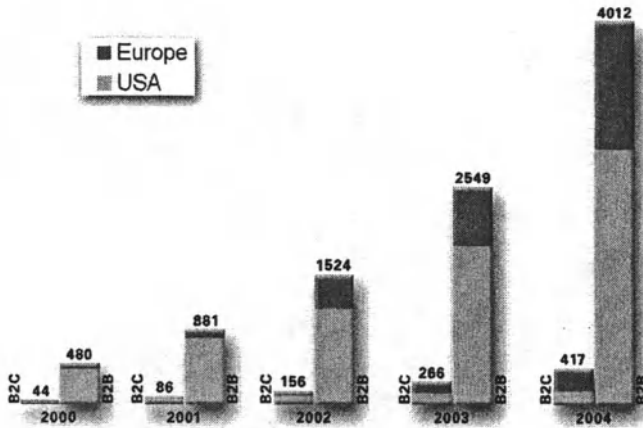


Figure 15. B2B dominates the e-business evolution

The following part presents a summary of the most important trends determining the e-business market in the near future. Major companies are preparing to greatly expand their investment in e-business. Thus, the next three years will see rapid growth of companies using e-business to improve the quality of customer and supplier interactions. The benefits of doing business electronically will attract increasing interest as companies realize the importance of not merely doing business better (e-commerce) but doing business differently (e-business). Priorities will focus on information sharing to better serve customers. Companies will strive to optimize customer relationships, creating the loyalty that is the basis for efficient profit growth and competitive advantage. E-business security and privacy are critical to companies' success in attracting online business. Especially security is still an open issue. It will either be a big business chance for R&D companies or it will herald the end of e- / m-business. Companies will be expected to explain their policies for protecting customers from risk and disclosure of personal information. The realization that a successful e-business is more about business and less about technology will grow as more executives grapple with the pressure to join electronic with traditional business structures. Priorities will include the tax, legal, risk and compliance issues that will allow companies to outdistance rivals. CEOs may find themselves managing the regrouping of company functions in response to increasingly flexible information and distribution channels and the disassembly of the value chain. PWC studies confirm that executives are savvy to the sweeping implications of e-business and are becoming educated themselves about their options. Companies want to see more successful e-business examples before embarking on such endeavors themselves. These e-business role models are likely to come from information-based industries such as high tech and financial services. E-business will continue to be a "net cash out" proposition

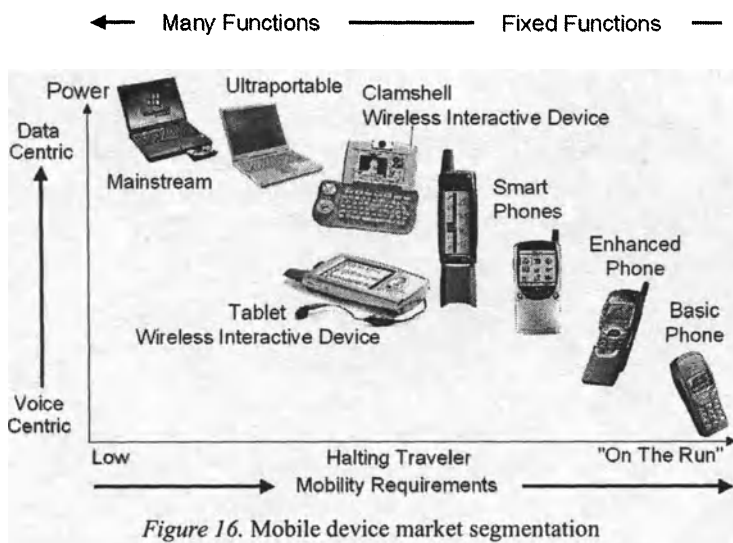
as companies build the infrastructure needed to succeed. The future winners will continue to invest while finding ways to evolve their existing investments in channels and infrastructure to become more e-business dominant. Effective partnerships and alliances are a key feature of internet-based organizations. They can lower start-up costs and share risks while accelerating progress toward becoming a true e-business. World Wide Web achieves global status. This is largely linked to the increasing globalization of business, which relies on an Internet environment where there are no geographic boundaries

4. M-BUSINESS

Mobile business is simply Internet + Wireless + E-business. This means changing the paradigm from a tethered PC-centric model towards a mobile-person-centric model. That is to say availability anytime and anywhere.

4.1 Mobile Device Market

Mobile business is going to raise the need for small, but increasingly powerful mobile devices. Figure 16 gives an overview of the market segmentation in the field of digital terminal mobile devices.



4.2 Mobile Subscriber Market

The number of mobile subscribers worldwide is increasing very rapidly. It is expected that by the year 2004 the number of mobile subscribers will exceed the number of fixed mainlines worldwide. Just one year later the number of mobile internet users will outrun the count of fixed internet users. Finland was the first country in the world (1998), where the mobile subscriber density exceeded the fixed subscriber density. Figure 17 displays the mobile subscriber density and the fixed subscriber density for selected countries.

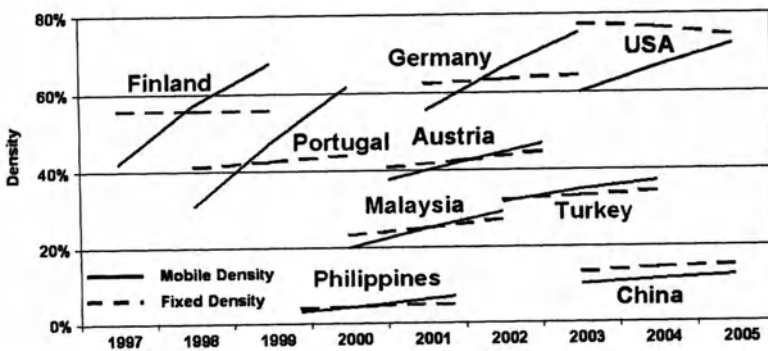


Figure 17. Selected mobile versus fixed densities

Despite the global economic slowdown the market for mobile subscribers will continue to grow and will reach a high of approximately 1.4 Billion mobile subscribers worldwide. Figure 18 displays the development of the mobile subscriber market from the year 2000 to 2005. Interesting fact is that the number of contract subscribers will be outrun by prepaid customers by 2005. Furthermore Figure 18 shows the overgrowth of the number of wireless internet users, which is mainly driven by m-business.

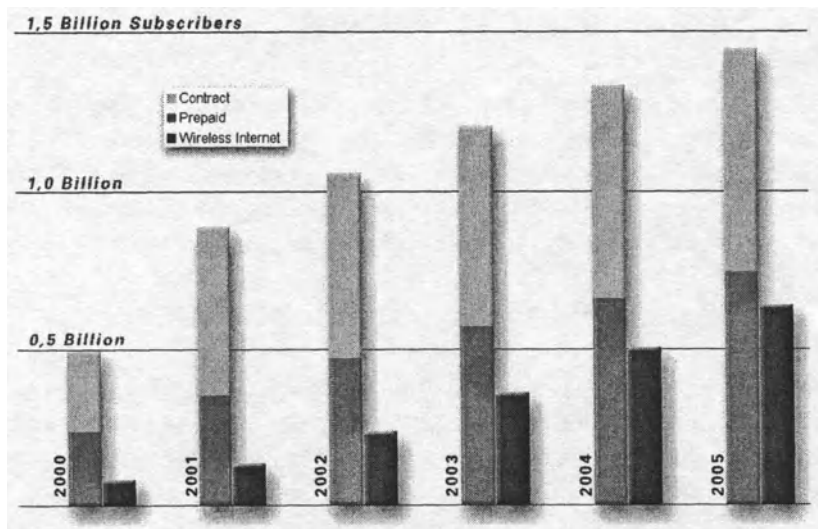


Figure 18. Evolution of mobile contract and mobile prepaid subscribers versus wireless internet users

4.3 Mobile Phone Market

These are challenging times for the mobile phone industry. After a decade of surging growth, global sales of handsets fell for the first time in the second quarter of 2001 by more than 8 per cent, as operators reduced handset subsidies and consumers held back from making replacement purchases. But if the current state of the market is a concern for manufacturers, the future looks terrifying - or at least terrifyingly difficult to predict. Manufacturers are having to grapple with the technological issues involved in designing the new handsets for GPRS and 3G systems without knowing how much attention each should receive relatively, with little hard data on how many units the operators will need or when they will want them, and a growing sense that end-users are quite happy with the functionality of their existing phones. On top of all that, those handset operators that are also in the infrastructure business know that 3G may arrive sooner than later only if financially-pressed network operators share equipment, thus reducing equipment sales. Steering a course through all this will be a severe test of manufacturers' product development, design and production strategies. The first hurdle is expected with Christmas 2001. Shipments of GPRS have already been delayed, and are urgently needed by operators and retailers if the holiday season is to provide the hoped-for boost to handset sales.

5. TECHNOLOGICAL TRENDS

Essential three directions drive telecommunications technologies today: (1) The advances in photonic networking with an evolution of ever-increasing bandwidth; (2) The tremendous progress in mobile technology with the evolution to global mobility; (3) The fast evolving pace of IP-networking as basic paradigm for an universal, ubiquitous switching method.

5.1 Evolution of the Ever-increasing Bandwidth

The Information and Communication Technology is the most rapidly growing market in the world. The increase of mobility and the demand for application bandwidth were the driving forces for the evolution of improved bandwidth technologies. Starting with some 100 kbit/s and reaching the range of Tbit/s. The main part of this evolution was provided by optical networks which now cover the globe. The amount of data that can be carried in each strand of fibre in these networks is growing. Dense Wavelength Division Multiplexing allows 40 channels of 40 Gbit/s to be sent down on one fibre – enough to transmit the whole of the movie 'Gladiator' in a fraction of a second. Every year, more wavelengths (or colors) will be available and each will carry more information. There is enough raw transmission capacity, at least on main trunk routes, to satisfy the world's traffic requirements several times over. For future data applications the available peak bit rate will be more important than the pure average bit rate. A user who wants to download a file will do that not permanently but when he tries to do it, he will not be willing to wait too long. As a forecast the average bit rate per application will be 0,1 to 0,5 Mbit/s in 2005 with streaming applications excluded (they will need a much higher bit rate). The corresponding peak bit rate per application should be 2 to 5 Mbit/s. Bit rate demands and the capability of the technology are interrelated. Whenever high bit rates can be provided at low cost, a corresponding application will appear and cause, after some time needed to get used to it, a demand for an even higher rate. This raises the question for the limit of that process. A preliminary answer is that for a single application all information which can be received by a person can be coded into 100 to 200 Mbit/s. Estimations however are highly uncertain and cannot be used as the single criteria to decide about future technology. This evolution will never satisfy the everlasting bandwidth demands. Figure 19 presents a survey of the milestones development in the core, fixed, mobile and satellite networks.

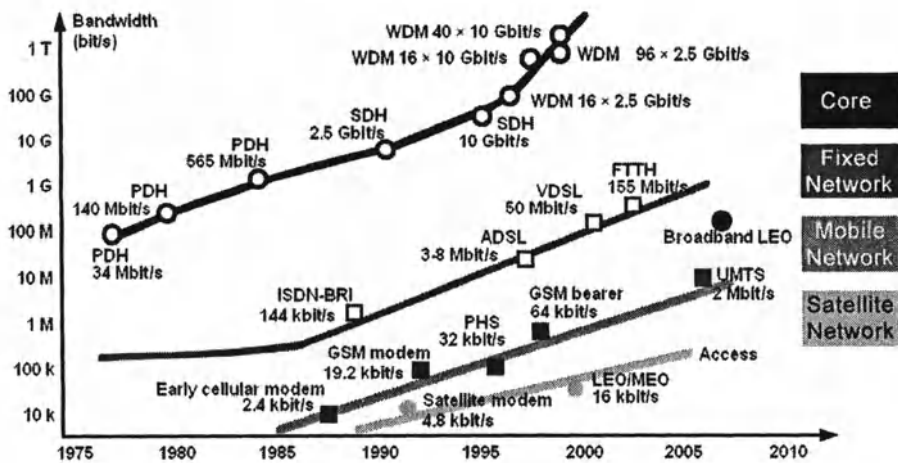


Figure 19. Evolution of the ever-increasing bandwidth

5.2 Evolution of Mobility

The telecommunications industry has developed in the last decade a number of new solutions to handle the complexity introduced within the network by the growing demand for new services. For mobile communication networks, new radio technologies in UMTS (Universal Mobile Telecommunication System) make roaming more powerful but at the same time more complex. New core network technologies, such as packet switching and IP multimedia, complicates roaming also from a service control perspective. Continuing the success of roaming achieved in GSM is a significant challenge and will be dependent on continued efforts by operators and manufacturers in creating the necessary high-quality standards for the future mobile communication system. The main emphasis is on mobile networks but there are still many benefits that accumulate both to the customer and to a combined fixed/mobile network operator, from attempting to share common infrastructure components between networks. Seamless services have been much discussed but the concept runs into problems due to detailed network disparity, different network management, and product portfolios. While not necessarily attempting to use the same network components between fixed and mobile networks, it is clear that there are many similarities in the technologies adopted by the fixed network. In the context of mobile telecommunication, three important aspects characterize a concept of personal communication support (PCS): support of mobility, including both terminal and personal mobility, service customization, and service integration/interworking. PCS is an improved form of universal

personal telecommunication (UPT) and will allow a UMTS user to customize his call handling policies.

Personal mobility is the ability of the end user to access services and to receive invitations to services on different terminals in different locations. In contrast, terminal mobility is the ability of a moving terminal to access telecommunication services from different locations and the capability of the network to identify and locate that terminal.

Service customization enables a user to define various conditions for his subscribed services by many parameters, such as time-dependent, service-dependent, and user-dependent call handling. The customized parameters of a user are usually stored in a user profile, which defines all services to which the user has access and usage, the way in which service can be handled, and all other user-configurable communication properties.

Service integration/interworking is required because UMTS will support personal mobility where the user can register from many kinds of terminals with various capabilities. Therefore, it will be necessary for the PCS to be able to convert an incoming call to adapt with the capabilities of the terminal at which the user is currently registered. Basically, the user can configure a set of attributes that define how an incoming call is to be handled by the system. This functionality is similar as in the virtual home environment (VHE).

Basically, the VHE concept in the third-generation mobile communication network like UMTS will allow the support service mobility, also called service roaming and service portability, respectively. The basic idea is that a roaming user will be able to customize his own environment individually at different terminals within different networks. The VHE will permit service mobility for the user, which means that the user takes his subscribed and customized services with him while he is roaming. VHE and its associated tools provide an ideal platform for use by independent entities (such as private networks and Intranets) that seek to deliver telecommunications-based services to their subscribers. The technology also enables to create totally new content and value added services. The flexibility introduced by VHE not only increases the range of services that can be offered to subscribers and users, but also allows them to be tailored to different user types and market sectors, even down to an individual user. The personal profiles can be extended beyond "any one, any time, any place" so that the services are consistent but also able to adapt to the users' circumstances as well as the capabilities of their terminals and access networks. Some of the requirements to realize in such a VHE environment are: (1) The capabilities of any available terminal within any accessed network to be sufficient for performing the services of the registered user; (2) Service conversion that can be useful to provide a service on a terminal,

which cannot support the original service. The VHE will enable a visited network to obtain information about the user's service profile as well as other information, such as the user's personalized service profile and identification of the service capabilities needed for the execution of service profile-specific services. The VHE can be seen as a (middleware) layer, which hides the concrete network capabilities and differences from the user and provider systems capabilities.

The third generation (3G) mobile networks are under development to support for diverse wireless access technologies, innovative applications that can seamlessly integrate data and voice, and service portability and mobility. These new activities in IETF (Internet Engineering Task Force) and the 3G communities indicate that future mobile telecommunication networks are rapidly migrating toward an all-IP technology. The all-IP-based 3G mobile network architectures can be viewed as consisting of a core backbone network (IP network) connected to many overlay networks, each with its own specific characteristics and different access technologies. The universal wireless communication networks (UWCN) consist of overlaid super-cells (via satellite), macro-cells, micro-cells and pico-cells that cover widely varying geographic areas, ensure global personal connectivity, and support different data rates and access technologies for mobile terminals within them. A key role of IP in 3G mobile networks will probably be the provision of efficient and cost-effective interworking between overlay networks. On the other hand, the availability of advanced IP mobility protocols makes Internet technology attractive for both interworking between different networks and for the definition of new mobile network architectures.

5.3 Evolution of IP-Networking

The largest network providing broadband capabilities is the Internet, and it is believed that future broadband networks will be based mostly on IP technology. The set of requirements for a generic broadband network is larger than the set of capabilities provided in today's Internet. Thus, it became necessary to eliminate some weaknesses by enhancing the internet protocols. An important new feature is the provisioning of quality of service (QoS), which is required for real-time services such as voice and video conferencing. Two protocols have been developed for QoS provisioning in the Internet: integrated services and differentiated services. The integrated services protocol is primarily designed for the access domains, whereas the differentiated services protocol is better suited for the core domain. QoS

routing and bandwidth management capabilities are provided by the Multiprotocol Label Switching (MPLS) protocol.

6. CONCLUSIONS

The analytical market study of the development of the information and communication technology (ICT) showed the different trends and their impact on the information communication networks and consequently smart networks. The paper showed that, despite the current global economic difficulties in almost every single industry, there are still a lot of business opportunities within the ICT industry and there are enormous business chances to adapt, to optimize, and to exploit in the already existing network-infrastructures in Eastern Europe as well as in the Third World. Technologically, three directions drive telecommunications today: photonic networking, global mobility, and IP-networking.

REFERENCES

- [1] Harmen R. van As, *Telecommunication Network Intelligence*. Kluwer Academic Publishing, 2000.
- [2] Ravi Kalakota and Marcia Robinson, *M-Business: The Race to Mobility*. McGraw-Hill, 2002.
- [3] European Information Technology Observatory, 2001.
- [4] *Next Generation Optical Networks, The Convergence of IP Intelligence and Optical Technologies*. Prentice Hall PTR, 1st edition, August 2001.
- [5] Jerry Wind, Vijay Mahajan and Robert Gunther, *Convergence Marketing: Strategies for Reaching the New Hybrid Consumer*. Prentice Hall, 1st edition, October 2001.
- [6] Stewart McKie, *E-Business Best Practices: Leveraging Technology for Business Advantage*. John Wiley & Sons, April 2001.
- [7] *Hacking Exposed: Network Security Secrets & Solutions*. McGraw-Hill, 3rd Edition, September 2001.
- [8] H. Aghvami, B. Jafarian, *A Vision of UMTS/IMT-2000 Evolution*, IEE Elec. & Comm. Eng. Journal, Vol. 12, No. 3, June 2000, pp. 148-152.
- [9] M. D. Cookson and D. G. Smith, *3G service Control*, BT Technol. J, Vol. 19, No 1, Jan. 2001, pp. 67-79.

- [10] Bruce S Davie, Yakov Rekhter, MPLS. Technology and Applications, Morgan Kaufmann, 2000.
- [11] T. Li: MPLS and the Evolving Internet Architecture, IEEE Communications Magazine, Vol. 37, No. 12, Dec. 1999, pp. 38-41.
- [12] 3GPP home page: <http://www.3GPP.org>.
- [13] Datamonitor: <http://www.datamonitor.com/>
- [14] European Information Technology Observatory: <http://www.eito.com/>
- [15] Financial Times: <http://news.ft.com/>
- [16] Forester Communications: <http://www.forester.net/>
- [17] Gartner Group <http://www4.gartner.com/>
- [18] Global Internet Project: <http://www.gip.org/>
- [19] IEEE: <http://www.ieee.org/>
- [20] IEEE Communications Society: <http://www.comsoc.org/>
- [21] International Data Corporation: <http://www.idc.com/>
- [22] International Engineering Society: <http://www.iec.org/>
- [23] International Softswitch Consortium: <http://www.softswitch.org/>
- [24] Internet Society: <http://www.isoc.org/>
- [25] Internet Surveys: <http://www.nua.com/>
- [26] InterPlanetary Network Special Interest Group: <http://ipnsig.org/>
- [27] Mobile Wireless Internet Forum (MWIF), <http://www.mwif.org/>
- [28] Next Generation Internet: <http://www.ngi.gov/>
- [29] OVUM: <http://www.ovum.com/>
- [30] Smau Italy: <http://www.smau.it/>
- [31] Telco Research: <http://www.peregrine.com/>
- [32] Tele Dot Com: <http://www.teledotcom.com/>
- [33] Total Telecom: <http://www.totaltele.com/>
- [34] WorldCom: <http://www.worldcom.com/>

Investigating Service Discovery, Management and Network Support for Next Generation Object Oriented Services

Bilhanan Silverajan, Joona Hartman, and Jani Laaksonen

Dept. of Information Technology, Tampere University of Technology, P.O. Box 553, FIN-33101 Tampere, Finland

{bilhanan | hartmanj | jllaakso}@cs.tut.fi

Abstract The network computing industry has eagerly embraced technologies, welcoming an ever-increasing variety of new service discovery protocols and object architectures. With this abundance now offered across a wide collection of environments, technologies that offer standardized interfaces for the discovery process, while supporting communication for several different types of service access technologies, will provide the greatest achievable interoperability and resilience in the long-term. In this paper, we introduce a distributed architecture based on using directory services to significantly reduce the complexity of managing the information and services required to support next-generation networked applications, by providing automatic service discovery and a single coherent model for representing the data managed by supporting services. Standards-based solutions are used, and a prototype implementation of the CORBA Naming Service that has been designed to illustrate how the architecture incorporates distributed object models, directory services and multicast-based dynamic service discovery is presented.

Keywords: Distributed Object Architectures, Next-Generation Applications, Directory Services, Service Discovery

1. INTRODUCTION

Recent years have witnessed the network and distributed computing industry embracing various technologies, media and content as a direct consequence of converging technical and business interests of telecommunications networks and Internet service providers. In the long run, this will likely lead to integration of all types of communications: a bewildering array of devices such as mobiles, fixed phones, PDAs, embedded devices, workstations and PCs will all be used to provide seamless communications between users and services. This has led to significant investment by many service and content providers into developing and furthering network and protocol-oriented technologies, for it is widely accepted that such a convergence will likely result in a structurally stable core packet-switched public network such as the Internet, having many edges acting as private gateways to proprietary and subscriber-based intranets offering a multitude of services using several different kinds of technology. These private networks may be both circuit and packet switched networks and will comprise subscriber-based IP connections, enterprise nodes and services, fixed and mobile telephone connections, intelligent network services and so on.

In this paper, we introduce a distributed architecture based on using directory services to significantly reduce the complexity of managing the information and services required to support next-generation networked applications, by providing automatic service discovery and a single coherent model for representing the data managed by supporting services. The architecture promises to be scalable and flexible enough to address basic issues that will arise in distributed computing as well as user, device and application mobility in the future. Section 2 discusses the type of network connectivity and support necessary to cope with the rising demand of mobile networked users and applications. Section 3 then sets out the aims and objectives of the proposed architecture while Section 4 provides an overview and a discussion of the proposed architecture, as well as describing a prototype that was implemented. Section 5 presents some performance measurements that were made, with the conclusions in Section 6.

2. REQUIREMENTS FOR NETWORK SUPPORT

Regardless of how services are utilized, two dominant scenarios are nevertheless heavily anticipated to influence and drive the development of next generation networks: User (and device) mobility and application mobility.

2.1 User mobility

Network connectivity and support for user mobility is a necessary measure which is already being provisioned for in many organizations. The overall demand for mobility has indeed not shown any significant decrease, as evident by the tremendous popularity and ever-increasing adoption of low cost portable devices such as mobile computing devices, PDAs and phones for computing and networking needs. This trend will very likely be even more widespread, given that roaming agreements signed amongst mobile operators may not only include current demands for seamless GSM-based voice or data calls, but might encompass future GPRS and UMTS enabled service.

In supporting user mobility, user identities, profiles, and sessions will need to be preserved across network boundaries. Very often user mobility is associated with device mobility, in which the mobile device retains its identity when roaming. The user may thus have an associated device which moves with him but the user may remain unaware of the underlying networking issues such as whether the device assuming temporary network addresses on roaming networks or retaining its address during base station handovers.

However, some applications resident on the device may need an awareness of the current location to access local services such as printers and file servers and might thus have to perform service discovery to a certain extent. Furthermore, visiting devices may also enrich the network being visited by offering their own services to the various applications and users that are resident on that particular network. Access control mechanisms would thus need to be exerted by the network hosting the user to ascertain exactly which local services are visible and what levels of service utilization are permitted.

There are several technologies which exist today that inherently support user and device mobility. These include wireless LAN technologies such as 802.11b, Bluetooth technology for piconet and scatternet oriented ad-hoc networking, as well as movement across networks as specified by mobile IP.

2.2 Application mobility

Application mobility can occur independently of user and device mobility, in a sense that all or certain parts of an application may autonomously or semi-autonomously migrate across heterogenous network spaces. An application may also consist of numerous non-mobile components or objects which transparently reside in several different parts of a network, with each component offering well known services to the others.

Network connectivity and support for application mobility on the other hand is foreseen in the medium to long-term future. It can be regarded as the next major wave in mobility after user and device mobility, which will have an important bearing on the underlying infrastructure.

Such an approach for application development is being undertaken in mobile code and agent technologies and frameworks, as well as distributed object-oriented architectures. Current examples include Java programs, automatic downloads of codecs by multimedia applications, Jini[1]-enabled programs, CORBA [2] applications and the forthcoming distributed applications of the Microsoft .NET framework [3]. Location awareness will be of lesser importance for migration when compared to computing resource availability of the present host and network.

Owing to the transparent, distributed and seemingly autonomous nature of such applications, it is rather unlikely they would enjoy as rapid and widespread an adoption across untrusted commercial networks as user and device mobility would. Rather, it is more probable that application mobility would garner more support and fill a demand for important niche services in enterprise level computing and service provisioning, within corporate intranets as well as private research networks of a more academic nature.

2.3 Supporting requirements

Such an environment of connectivity and mobility will have a significant impact on the way next-generation applications, both autonomous and interactive are implemented, discovered, managed and accessed. One can easily perceive that both user mobility and application mobility will become killer scenarios for existing networks and services. These will create an overwhelming strain in the way current networks and legacy applications manage, interact and serve the necessary information that needs to be delivered to the end-users and applications. In both cases, a more robust supporting infrastructure for handling next-generation applications needs to be developed in a standardized manner that must also prove scalable and adapt to accommodate the research and development challenges that will manifest themselves.

The overall design must possess some flexibility, as applications being built today are likely to function as part of bigger, integrated systems tomorrow. The architect of tomorrow's distributed systems will face large-scale distributions both in terms of the number of users and connected systems running atop heterogeneous networks. There may not be a revolutionary change in the way networks and services will be designed for supporting next-generation applications. In fact, all indications point to the development being evolutionary, with network architects and scientists

examining, using and extending existing solutions and standards to support new applications and services as the need arises.

The huge potential of being able to harness an increased subscriber base of a converged market has led to several sets of standards continuously being introduced for communication protocols and architectures. In many ways, the objectives and target applications to be supported by these technologies can be remarkably similar. Even the technologies may mirror each other. However, in reality, getting applications and services supported by one set of standards to communicate with another pose interoperability challenges. This also leads to an unfortunate effect of leading to fragmented or duplicate solutions. As an example, IETF's Service Location Protocol [4], Sun Microsystems' Jini, Microsoft's UPnP [5] and wireless technologies promoted by IrDA and Bluetooth all implement their own service discovery mechanisms for potentially common types of services such as printing. Another notable example would include an object location service such as CORBA and RMI naming services.

Thus, a service designer targeting a product for multiple networking environments, object services or service discovery methods may be faced with the dilemma having to develop and managing several different kinds of applications, each supporting a specific protocol or object model but in all probability having duplicate information models.

Instead, trying to achieve the same goal by supporting protocol bridging and conversion methods for several service discovery methods and location services, with different front-ends using a common back-end application-level information repository storing service records and attributes, would be a far less harrowing experience.

The challenge then becomes one of developing a unified yet scalable service information management system that stores and manages service data in a standardized and generic manner, but still provides the appropriate data to requesting application via a specific discovery method, using gateways and information mapping and translation.

3. AIMS AND OBJECTIVES

A prototype distributed environment that would enable devices, object-oriented client applications, and services to automatically search for a particular capability, and to request and establish interoperable sessions with other clients or servers was investigated to meet the following objectives:

- To provide applications, services and devices a standard method for describing and advertising their capabilities

- To allow mobile users and applications the possibility to discover local services upon application start-up or movement within network space or administrative zones automatically or with as little static service discovery configuration as possible
- Interworking non-invasively or transparently with existing code and applications
- Ease of administration and maintenance by providing a single consistent model of available services that can scale, at the very minimum, to meet enterprise computing needs, thus reducing or eliminating the need for duplicity of information needed to support various access methods
- Integration into existing or commonly prevalent network resources, supporting components and infrastructure that may already be in place, but remaining extensible enough to meet future needs
- Well-known user and application access control and security mechanisms can be enforced, and existing authentication practices and encryption methods within the organization can be used.

4. ARCHITECTURAL OVERVIEW

Figure 1 illustrates a prototype implementation of a CORBA Naming Service [6] that has been designed to fulfill the objectives mentioned in section 3. CORBA has become increasingly important in distributed computing for the Internet as well as telecommunications owing to its language independence and separation of the object interface away from its implementation, and the increased use of distributed objects needing fundamental CORBA based object services such as the Naming Service cannot be ignored.

In this prototype, the back end service repository was implemented as a directory service using the Lightweight Directory Access Protocol (LDAP) [7]. This allowed the possibility to leverage the existing prevalence of using LDAP-based services and servers based on a well-known IETF-standard protocol already commonly deployed at the organizational level. Moreover much research is being done to make LDAP services secure, scalable, distributed and fault-tolerant. Clients would be able to retrieve information about services either natively through LDAP, or through a lightweight gateway/proxy that will implement the specific mappings necessary to support the particular type of service technology (in this case the CORBA Naming Service) the client would be using. As the figure shows, the model would be capable of supporting other kinds of front-ends as well.

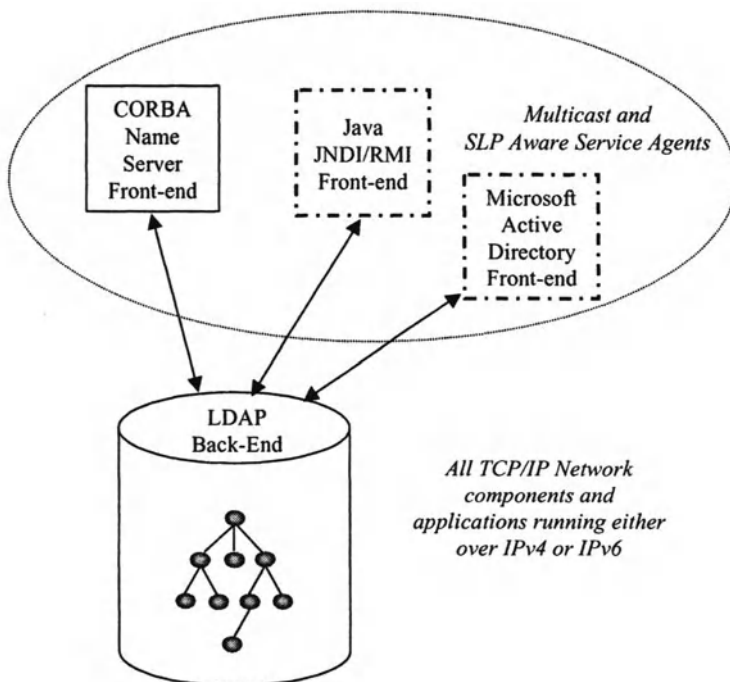


Figure 1. Prototype directory-based CORBA Naming Service

The discovery of the CORBA Naming Service by CORBA client and server objects was made possible using the Service Location Protocol (SLP) as a bootstrapping protocol for multicast-based for service discovery.

A brief description of SLP, LDAP and the CORBA Naming Service together with how they have been used or implemented are discussed in the following subsections.

4.1 Automatic service discovery using SLP

The Service Location Protocol (SLP) provides a flexible and scalable framework for providing hosts with access to information about the existence, location, and configuration of networked services. Traditionally, users have had to find services by knowing the name of a network host or its network address [4]. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service and a set of attributes which describe the service. Based on that description, SLP resolves the network address of the service for the user.

SLP models client applications as User Agents (UAs), while services are advertised by Service Agents (SAs). Applications which would like to advertise their respective services use IETF standardized API functions to

contact the SA running on the same host to register a Uniform Resource Location (URL) to be advertised, such as "service:ftp://ftp.company.com/". A service can moreover have attributes that describe the service (or the URL) in more detail. As an example, a web server would have its URL as "service:http://www.company.com" with attributes possibly listing the name, email address or a telephone number of the webmaster. SLP also allows services to be administratively grouped into scopes that could then be controlled for service provisioning to UAs.

Client programs (UAs) that would like to find services in a network also use standardized IETF function calls to find available services in a network and are capable of querying available service types, specific service type URLs as well as attributes associated with a specific URL. Directory Agents (DAs) are an optional third category of agents that SLP defines which allows for scalability by caching SA advertisements for direct interaction with UAs. Service discovery can either be statically configured or allowed to take advantage of multicasting or broadcasting features in a network for dynamic configuration.

The choice of using SLP as a primary service discovery protocol was influenced by the fact that it has been specified by the IETF, and hence did not need to be adapted to be used over TCP/IP. Moreover, it is generic and independent of any particular language and object architecture. SLP's simplicity combined with its feature set also enables it to interwork with various other service discovery methods and environments such as Jini using bridging [8] and the Apple Computer's next-generation AppleTalk [9]. Much standardization work is also in progress, such as efforts in combining SLP with LDAP [10], and extensions for SLP for use over IPv6 [11].

An open-source implementation of SLP, named OpenSLP [12] that implements SLP version 2, was primarily used in developing the SLP UA and SA components of the architecture. In the later stages of development and testing, the SLP system libraries of Sun's Solaris 8 were also used successfully. As only a minimum configuration was involved, no Directory Agents were used, and the architecture was developed to use dynamic discovery and communication between UAs and SAs using multicasting instead.

4.2 Information storage and retrieval using LDAP

LDAP was originally developed as a front end to X.500, the OSI directory service. X.500 defines the Directory Access Protocol (DAP) for clients to use when contacting directory servers. Currently at version 3, LDAP however has evolved to eventually provide most of the functionality of DAP at a much lower cost and defines a reasonably simple mechanism for

Internet clients to query and manage an arbitrary database of hierarchical attribute/value pairs over a TCP/IP connection. LDAP has rapidly gaining significant Internet support, including the support of many companies, such as Novell, Sun, HP, IBM, SGI, AT&T and Banyan, and is the focus of much standardization activity in the IETF.

The LDAP directory service model is based on entries. An entry is a collection of attributes that has a name, called a distinguished name (DN). The DN is used to refer to the entry unambiguously. Each of the entry's attributes has a type and one or more values. The types are typically mnemonic strings, like "cn" for common name, or "mail" for e-mail address. The values depend on what type of attribute it is. Directory entries are arranged in a hierarchical tree-like structure that reflects political, geographic, and/or organizational boundaries, representing people, organizational units, printers, documents, or just about anything else one can think of.

The information model stored in the LDAP directory backend of our prototype was loosely modeled upon the campus computer science department building, in both the physical and organizational sense. The top level root entry (o=CS,c=FI) represents the computer science building which was then subdivided into four floors representing the structure, personnel, laboratories, services and IP subnets resident on these floors. The basic idea in storing service and object information in the LDAP directory in this manner was not only to store the access method of the object or the URL of the service, but also to append the geological or organizational location information of the respective services and objects. Any LDAP browser could then be used in obtaining detailed information about services offered within particular subnet or subnets (such as printers), by inspecting the relevant attributes of the LDAP entries, such as their network addresses, physical location as well as different access methods (network printing, Bluetooth/IrDA connectivity, Appletalk, and so on). The OpenLDAP [13] project provided the necessary tools and libraries needed for this part of the architecture.

4.3 Object location discovery using CORBA Naming Service

The Naming Service was implemented in C++ with MICO [14], an open-source CORBA implementation which conforms to CORBA 2.3, using the DSI (Dynamic Skeleton Interface) to dynamically handle the object invocations. Apart from being designed specifically to aid in the development of gateways, the CORBA DSI functionality will allow the possibility to expand the interface of the CORBA naming service to possibly

serve other kinds of naming services, since type-checking is done at runtime. Clients are free to use either the DII (Dynamic Invocation Interface) or the SII (Static Invocation Interface) to invoke the methods on the server object.

The CORBA objects are stored in the directory as stringified object references. LDAP offers a schema for representing CORBA Object References in an LDAP Directory [15], with each entry storing a textual description of the CORBA object as well as its IOR (Interoperable Object Reference). The IOR is the attribute with the single-most relevance in this case, as it is the primary means used by CORBA clients to locate and communicate with CORBA server objects in a network. Figure 2 displays the LDAP directory as seen by a freeware LDAP browser, Softerra LDAP Browser [16] running on Windows2000, after the Naming Service was launched and was used to register a simple CORBA Messaging server application, showing its various attributes, location as well as its DN in the window titlebar.

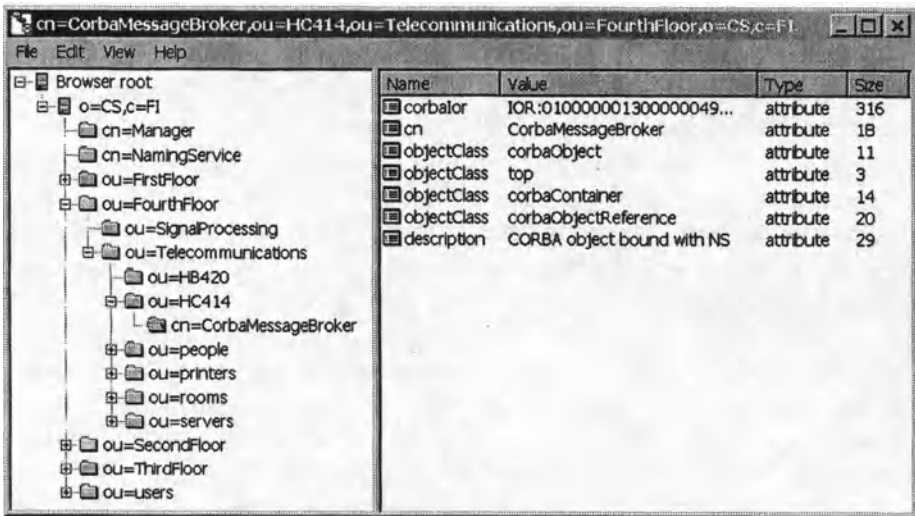


Figure 2. LDAP Browser showing an entry representing a CORBA object

Because the CORBA Naming Service that has been specified by the Object Management Group directly supports the notion of name-to-object associations using name bindings relative to a naming context [6], an entire Naming Service is conceptually a naming graph having hierarchical relationships among parent and child nodes representing contexts and directed edges forming names. This relationship is directly supported by the hierarchical nature of an LDAP directory structure. Also, a name is comprised of a concatenation of components, where components each consist of an *id* attribute and a *kind* attribute, with the last component

denoting the bound object. The Naming Service specifications do not restrict the way in which the naming system should interpret, assign or manage these attributes. Thus our prototype maps these structures to be directly relevant to the entries in the LDAP tree, in such a way that the *kind* attribute is mapped to the corresponding LDAP entry's attribute type, such as "cn" or "ou", and the *id* attribute is mapped to its value. Therefore, if the CORBA Naming Service binds itself to the LDAP Tree with the DN of "o=CS,c=FI", then

"ou=FourthFloor,ou=Telecommunications,ou=HC414,cn=CorbaMessageBroker" would actually be a valid name string for binding or resolving an object using our prototype Naming Service, as it would convert the given name sequence to a string representing the DN of the LDAP entry.

At the moment, the Naming Service implements four methods: *bind*, *unbind* and *rebind* are used by CORBA server objects to register themselves, and *resolve* is used by CORBA client objects to find the IORs of server objects corresponding to their names from the Naming Service.

4.4 Example usage scenario

For starting the Naming Service, the following steps are undertaken:

1. OpenLDAP server daemon slapd, must first be running.
2. The OpenSLP server daemon, slpd is launched on all machines which desire to host SAs.
3. The CORBA Naming Service application is launched, and binds to a well known location to the LDAP back-end. This will be known as the root context of the Naming Service. At the moment, this is configured statically. However this can be easily extended to be dynamic as well.
4. The Naming Service registers itself to its local slpd, passing its URL (e.g. "service:namingservice://myhost.company.com") and its stringified IOR as an attribute.
5. The Naming Service is now ready to serve CORBA objects.

Registering a CORBA server application with the Naming Service would proceed in the following manner:

1. CORBA Server starts up, and behaves as a UA which multicasts a Service Request packet to the Administratively Scoped SLP Multicast Address [17], 239.255.255.253, requesting for a service type "namingservice". Currently, UAs use default scopes with a multicast packet TTL value of 8.
2. The SA responsible for the Naming Service unicasts a Service Reply back to the requesting UA, returning its URL and thus specifying the location of the Naming Service.

3. Armed with the URL, the UA once again multicasts an Attribute Request packet requesting for the IOR attribute for that URL.
4. The SA once again unicasts directly to the UA with an Attribute Reply, furnishing it with the IOR attribute registered with that URL.
5. Armed with enough knowledge now, the CORBA Server application contacts the Naming Service directly with the IOR it possesses, and registers itself with the Naming Service using the bind call with 2 parameters: its name and its IOR.
6. The Naming Service maps the information into an LDAP call and stores the information in its LDAP database backend.

In order for a CORBA client finding a CORBA server, it does the following:

1. The CORBA client first discovers the Naming Service in a similar way to the CORBA Server.
2. The Client calls the resolve call of the Naming Service, passing the name of the CORBA server object as a parameter and obtaining an IOR from the Naming Service with which the client then invokes methods on the CORBA server object.

5. RESULTS

The basic network topology used for developing and testing the prototype implementation is depicted in Figure 3. All 3 subnets were multicast aware, with the FreeBSD machine hosting the LDAP, Naming Service, CORBA server objects and SLP Service Agents.

As an initial indication of the performance of the prototype, the timing measurements that were recorded are tabulated in Table 1. All SLP and CORBA client calls in Table 1 were made from the SunBlade machine to the FreeBSD machine. For the Naming Service, the time taken to execute a single resolve call was measured. The time taken to execute the same call was also measured against MICO's native naming service, which runs as a standalone daemon in the network. For SLP measurements we timed the discovery of the Naming Service. No DAs were employed in this topology, hence communication occurred directly between the UA in the SunBlade machine and the SA in the FreeBSD machine. The measurements were taken over a period of approximately 3 days.

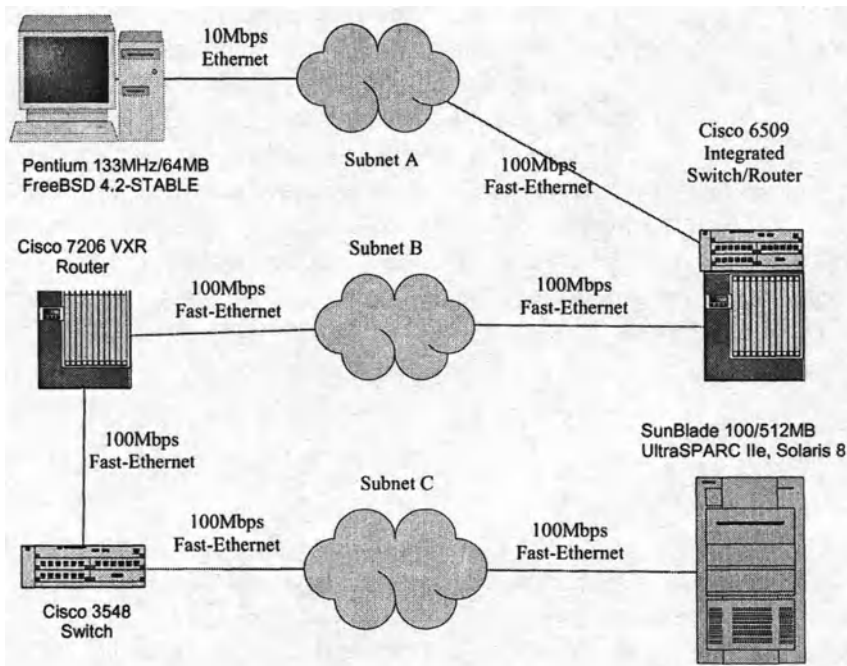


Figure 3. Physical Network Topology

Approximately two-thirds of the execution time of the resolve call was consumed by the LDAP operation, while a small fraction was consumed in the Naming Service front-end that serves as a gateway by mapping parameters and return results to and from the CORBA Naming Service API and the OpenLDAP API. On the other hand, to use MICO's own Naming Service, it is necessary to initially run an object adapter daemon called micod, create an entry for the naming service in an implementation repository and pass the clients the address of the naming service. A fair bit of manual intervention, static configuration and using commandline options are hence necessary. Also, the entries of the Naming Service are stored in memory.

Table 1. Timing measurements in milliseconds

	Prototype	MICO's Naming Service	SLP
Average duration	36.5843	3.60773	4.24144
Maximum duration	148.506	34.252	10.203
Minimum duration	34.289	3.213	3.960

Thus, bearing in mind the objectives laid out in Section 3, we firmly believe that the offset in performance of our prototype implementation is justified by virtue of the scalable, replicating and fault-tolerant properties

inherent from the LDAP component of the architecture. Automatic service discovery is also ideal for object services, owing to their more volatile and migratory nature as compared to fixed network services. The design of the architecture with the components described also allows for the flexible interaction amongst them, depending on the needs of the organization. As an example, the OpenLDAP server could be modified to become SLP-aware so that even the Naming Service front-end could perform service discovery to dynamically discover its backend, as opposed to the configuration that our prototype used. Interworking between SLP and LDAP is being studied, so that if Directory Agents are used in the architecture, SLP service URLs and attributes can be stored in an LDAP Directory. [10]

6. CONCLUSIONS

Almost all the software components used in designing, building and testing the prototype implementation have been based on code obtained from highly active open-source projects. This has made troubleshooting and bug-solving relatively easy through open mail-list forums and by simply browsing through the code. Minor problems, such as UAs in OpenSLP insisting on multicasting to search for DAs even when configured not to, were quickly solved, and more optimal service discovery times were achieved (from a very rough estimate, UAs using Solaris's SLP library for service discovery took far longer, about 12 seconds).

At the moment, SLP over IPv6 within our prototype has yet to be tested. However, the following are changes required to have the Service Location Protocol work over IPv6. These changes include [11]:

- Eliminating support for broadcast SLP requests
- Address Specification for IPv6 Addresses in URLs
- Use of IPv6 multicast addresses and IPv6 address scopes
- Restricted Propagation of Service Advertisements

The architecture proposed in this paper also does not preclude the use of any security models and can remain fully conformant to any and all security mechanisms standardized by the many RFCs and other specification documents for its various components. As an example, SLP authentication and LDAP security and access control mechanisms can be used to enhance the architecture to allow employees and regular users to use all standard services of the organization's internal network, but restricting visitors and anonymous users to a smaller subset, perhaps by presenting a different location service front-end during service discovery which has a more limited level of access and visibility of the LDAP directory.

Being able to store, access and manipulate data stored in a common directory format with LDAP also has a huge advantage in being able to use common LDAP browsers (with varying levels of quality) supported in many platforms. This allows a far greater ease of management and maintenance than having to maintain several different service models, each having their own customized administration tools. This also significantly reduces the learning curve for efficient tool use.

The network computing industry has eagerly embraced technologies, welcoming an ever-increasing variety of new service discovery protocols and object architectures. With this abundance now offered across a wide collection of environments, technologies that offer standardized interfaces for the discovery process, while supporting communication for several different types of service access technologies, will provide the greatest achievable interoperability and resilience in the long-term. In this respect, the proposed architecture in this paper holds good promise for supporting enterprise-level next-generation computing needs.

REFERENCES

- [1] Sun Microsystems: Jini Network Technology, <http://www.sun.com/jini/>
- [2] OMG: The Common Object Request Broker: Architecture and Specification. CORBA V2.3, June 1999.
- [3] Microsoft Corporation: .NET <http://msdn.microsoft.com/net>
- [4] IETF: RFC 2608, "Service Location Protocol, Version 2", June 1999.
- [5] Microsoft Corporation: Universal Plug and Play forum, <http://www.upnp.org>
- [6] OMG: Naming Service Specification, February 2001.
- [7] IETF: RFC 2251, "Lightweight Directory Access Protocol (v3)", Dec 1997.
- [8] Erik Guttman, James Kempf: Automatic Discovery of Thin Servers: SLP, Jini and the SLP-Jini Bridge, IECON, San Jose, 1999.
- [9] Apple Computer Inc.: Mac OS X Server - Network & Security <http://www.apple.com/macosx/server/networksecurity.html>
- [10] IETF: RFC 2609, "Service Templates and Service: Schemes", June 1999.
- [11] IETF: RFC 3111, "Service Location Protocol Modifications for IPv6", May 2001.
- [12] The OpenSLP Project, <http://www.openslp.org>
- [13] The OpenLDAP Project, <http://www.openldap.org>
- [14] MICO - Mico Is CORba, <http://www.mico.org>

- [15] IETF: RFC 2714, " Schema for Representing CORBA Object References in an LDAP Directory", October 1999.
- [16] Softerra LDAP Browser, <http://www.ldapadministrator.com>
- [17] IETF: RFC 2365, "Administratively Scoped IP Multicast", July 1998.

Virtual Private Components

An Approach for Leasing Components

Alek Opitz, Torsten Mielke¹⁾, and Hartmut Koenig

*Brandenburg University of Technology at Cottbus, Department of Computer Science,
PF 101344, D-03013 Cottbus, Germany*

¹⁾ *Torsten Mielke is now with IONA Technologies Ltd. Dublin.*

Abstract Component technology has become an important approach to set up distributed applications and systems. Components possess exactly specified or standardized interfaces and explicit context dependencies. They can be purchased at component markets and integrated into the application. The purchase of components, however, might be not the onliest solution in future. If a component requires large resources, an expensive maintenance, or a high degree of topicality it might be more favourable like in everyday life to lease a component for a certain time. This can save purchase cost and give the customer the possibility to always use the newest version of a component. The leasing of components requires a particular framework. Beside a contract which defines the conditions of the component use, a technical infrastructure is required which ensures the availability of the service and the enforcement of the contract agreements. In this paper we present a technical framework for the leasing of components called Virtual Private Components. We further present an architecture to support the leasing of components and discuss implementation issues. Finally, we present a prototype implementation.

Keywords: Component technology, Software Leasing, Virtual Private Resources, Middleware, Long-term service customer-provider relationships

1. MOTIVATION

The communication infrastructure of the Internet is constantly improving. Advanced communication technologies and protocols allow the use of services which are provided at distant places of the earth. This more and

more allows the construction of distributed applications and systems based on such services.

The dynamic integration of services of various providers into an application and the acceptance of possible updates require standardized interfaces for their integration and for the access to these services. The component software technology represents a promising approach to provide techniques to solve these problems. A component is a piece of software with exactly specified or standardized interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties [Szyp98]. Components will be offered and traded on open component market places. Customers can purchase needed components and import them, possibly via the Internet, into their application. Current component technology is characterized by the purchase of components, which then become property of the customer, and their direct integration into the application [Szyp98], [Mons99], [Wimm99], [Sieg00].

As in everyday life there might be situations where the leasing of components will be more favourable. This will be the case for services with large resource and maintenance requirements, and high demands on the topicality. Examples of such services are encyclopedias, information services, e. g. for stock exchange data, geographic information systems; simulations and calculations on supercomputers, high-performance and reliable web servers, dynamic security management functions and others.

To set up composite services which use remote services stable and reliable relations to the providers are required. Unlike purchased components which are directly integrated into the application leased components require mechanisms for an installation limited in time. Contracts have to be specified which define the conditions of the service provision and use. Mechanisms are required which guarantee the fulfillment of the contract. In addition, means must be provided to advertise these components on component markets as well as measures to protect consumer interests.

In this paper we present an architecture for the leasing of components called Virtual Private Components. It extends the concept of the Virtual Private Resources [Preu99b], which supports a long-term binding of services. The paper focuses on architectural aspects of the Virtual Private Components approach. In Section 2 we discuss aspects of the leasing of software. Section 3 introduces the basic principles of the Virtual Private Resources approach needed for the further discussion and summarizes the motives for a redesign of the approach. In Section 4 we describe the concept of Virtual Private Components and outline the differences and benefits compared to Virtual Private Resource approach. In Section 5 we present an architecture to support Virtual Private Components. Section 6 describes a

prototype implementation and discusses implementation issues. The final remarks give an outlook on next research steps.

2. LEASING OF SOFTWARE

The leasing of software (also called application service providing (ASP)) is a new developing sector in the software market. It provides customers similar benefits as the leasing of products in everyday life. Software leasing saves purchase cost like usual leasing and gives the customer the possibility to deploy the latest versions with the newest features each time. This is especially advantageous for small companies or organizations which cannot afford large expensive software systems. In addition, it avoids installation errors by new patches or update versions. Examples for leased software systems are office applications, websites, task management systems, enterprise software solutions and others. For the leasing of software, there are two possibilities: the temporary installation of the software received from the ASP and the invocation of a service which is provided remotely. In this paper we consider the latter case. The permanent binding of a service reveals problems concerning the availability, the security, and the billing of the service. In the following discussion we focus on the problem of the constant availability of services.

Service mediation and binding in open service markets is mainly based on the trading concept [ISO97], [OMGS98]. The trading approach supports a short-term relation between the service customer and the provider. The service customer addresses to the trader on demand, i.e. in an ad hoc manner. The trading concept does not guarantee that the same service with defined service properties is available for a longer period of time and can be called several times. If a service customer again calls the service it cannot be guaranteed that the same service will be bound again. In addition, the service provider may withdraw a service offer at any time. The trading concept is therefore only suited for the single use of a service. In many cases, especially, when users spontaneously address to the market, this mode will meet the requirements of the users. However, for users who need certainty that a particular service is provided, e.g. for composing a new service with defined properties, this is insufficient. An approach to permanently binding services is the Virtual Private Resource concept proposed in [Preu97]. It is shortly introduced in the next session.

3. VIRTUAL PRIVATE RESOURCES

Virtual Private Resources (VPRs) are an approach for the constant binding of services with a defined Quality of Service (QoS) fixed in a contract between the service customer and provider. A VPR represents a service or a set of services that can be bound from a remote service provider for a longer period of time, e.g. weeks, months, or even years. The conditions of the VPR use are defined in the *VPR contract*. A VPR gives the customer the illusion that the service is permanently available and can be used like a private resource. This resource, however, is a virtual one. It is located in a remote domain and is mostly shared with other applications (see Figure 1). The service provider guarantees that the service is available at a defined time and assures the provision of the agreed service properties. VPRs were designed for the outsourcing of services. They can be also used to establish permanent customer-supplier relationships in electronic service markets [Preu99b] similarly to fixed supplier chains for just-in-time production.

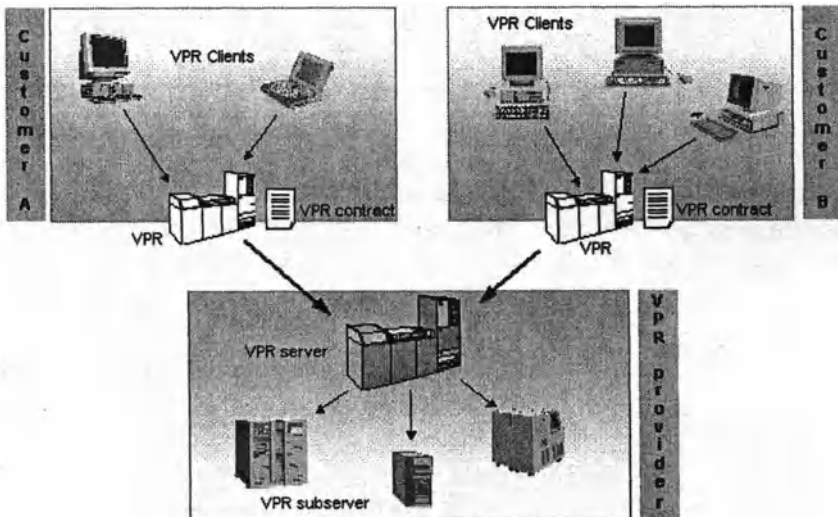


Figure 1. Virtual Private Resources

3.1 VPR Architecture

The transparent use of VPRs requires particular measures for their provision at the provider side. Figure 2 shows an architecture for the provision of VPRs. It distinguishes two domains: the customer domain and the provider domain. *VPR customers* are organizations and enterprises that

have contracted VPRs. Applications and/or human users belonging to that organizations are called *VPR users*. The users can be supported by a *VPR manager* for adaptation purposes, e. g. if the VPR contracts has been renegotiated. VPR users and the VPR manager access the VPRs via the *VPR proxy* which represents the VPRs in the customer domain.

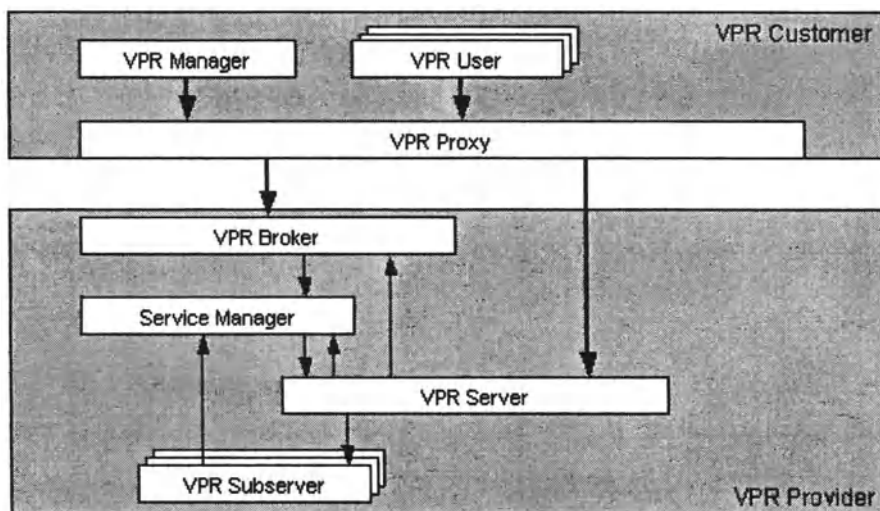


Figure 2. VPR Architecture

The counterpart of the proxy in the provider domain is the *VPR broker*. It mediates the requests from the *VPR proxy* to appropriate servers which implement the related services. The mediation remains hidden to the VPR users. Note that the *VPR broker* is not a trader. The binding of the server is based on resource requirements and availability. A VPR is set up by the *service manager* according to the specification received from the broker. The *service manager* can bind further subservers if required. After setting up a VPR the reference to the respective server is returned to the *VPR proxy*. It calls the server and starts the service execution. The results of the service are delivered via the proxy to the VPR user.

The presented VPR platform was implemented using ORBIX 2.3 based on CORBA 2.1. It was used to set up a traffic information system as Virtual Private Resource [Preu99a].

3.2 VPR contracts

The *VPR contract* defines the conditions for the VPR use. It specifies among others the duration of the availability of the service, the frequency of

its use, the QoS provided, and the cost of the service use. The VPR contract is the most crucial feature of the VPR concept, because it specifies the obligations of the VPR provider. The contract has to be negotiated between VPR customer and provider before binding the service. This can be agreed by representatives of both sides in conventional way or automatically if such means are available. In order to exactly specify the contract conditions the range of service properties was extended [Syrb98]. Additional properties were introduced to specify the scope of the service, the results, the execution, the duration of the provision, the required capacities, and system aspects. Each property describes a range of values limited by maximum and minimum, and threshold values to define the optimal operation range for the property.

The contract conditions are enforced by the VPR provider for the whole period of the contract validity. Two different kinds of measures are applied: reactive and post-use measures. Reactive measures are taken by the node managers which observe the operation of the servers. Examples are the assignment of other priorities to VPR invocations, the activation of standby resources, and the migration to other servers. Post-use reactions are based on monitoring the execution process. The observers constantly measure the values of the dynamic service properties and return usage profiles as feedback to the VPR broker. These profiles are used for new VPR invocations to select the servers. Detailed descriptions of the VPR mechanisms and the related procedures are given in [Syrb98], [Preu99a].

The concept of Virtual Private Resources mainly aims at the provision of complex services rather than smaller partial services which can be combined with other services. The approach requires a pretty tight relationship between the customer and the provider which makes a change of the provider difficult. This concerns less the conditions of the VPR contract which can be negotiated flexibly, but more the technical aspects of the customer and provider relation. The application has to support the mechanisms for binding and calling the VPR. Currently these mechanisms must be directly coded into the application and linked together with class libraries of the VPR provider. The change of the provider requires an experienced system engineer to handle the technical details at both sides. Similar problems appear when the interface of the service changes. Therefore a redesign of the concept has been considered appropriate.

4. VIRTUAL PRIVATE COMPONENTS

4.1 Basic Characteristics

Virtual Private Components (VPCs) aim at the leasing of components which are provided at a remote location in the Internet. A VPC represents a partial service needed to set up more complex distributed applications or services. It can be bound for a certain time and combined with other bought or leased components [Miel00]. The leasing period is supposed to be a longer period of time, e.g. several months or years. During this time the provided service is supposed to be always available. It can be activated with a constant QoS as often as needed according to the agreements of the *VPC contract*. Unlike the VPR concepts the VPC approach allows that an application can simultaneously bind components from different VPC providers. It also can change a VPC provider if required.

The upper part of Figure 3 gives an example of a possible VPC application: a credit card checker to verify the trustworthiness of the credit card holder. When the holder wants to pay with his/her card the bank proves whether the client still has credit and whether the payment corresponds to his/her usual card deployment. For the latter, client profiles are used which indicate the average sum of payments, preferred goods, countries and frequency of the card deployment, and other criteria. Instead of developing its own verification software a bank can use a service offered by a VPC provider which is specialized in such checks. The bank calls the service within its payment handling procedure. It is thus the *VPC customer* and the checking company the *VPC provider*. The credit card holder and the shop assistant are not affected by this. They do not even know that a VPC is used. Further examples of possible VPC applications are traffic information systems which inform drivers about the actual states of the streets when he/she enters a new region to help them to navigate through this region, web servers for small businesses which provide the whole hardware and security infrastructure, virtual printer services to handle print orders of different clients, and many others [Opit00].

What are the benefits of using VPCs? A VPC gives the customer the opportunity to benefit from varying offers of the VPC providers. So a bank can change, for instance, the credit card checker service when an improved verification algorithm appears on the market. Analogously to the selection of the cheapest provider at the telephone market, the VPC customer may select the service which best meets its requirements. Selection criteria may be the service quality, security levels, availability, topicality and of course the prize.

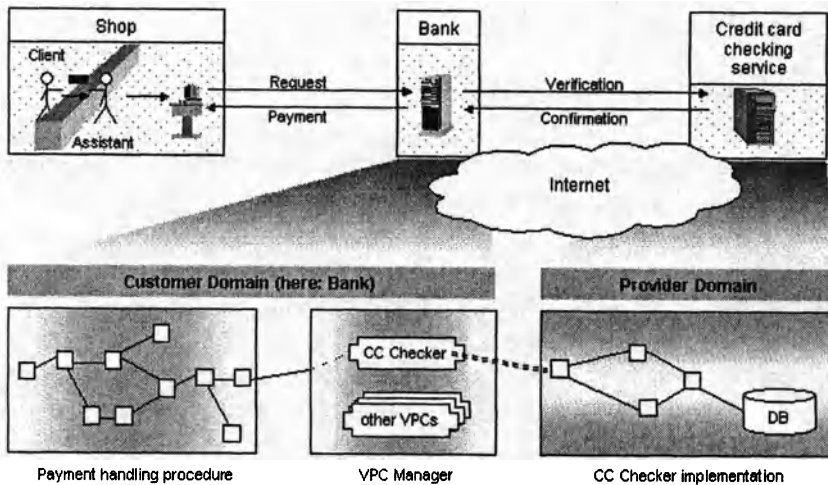


Figure 3. Credit card (CC) checking service

4.2 Requirements to a VPC architecture

An architecture to support Virtual Private Components has to meet different requirements related to functionality, performance, and security.

Functionality

The VPC architecture should be flexible and independent of different providers as well as of the middleware platforms on which it will be implemented. It should meet in particular the following demands:

- *Change of service providers*

The application should be able to select other service providers which provide better contract conditions. The technical efforts to change the service provider should be minimized. The customer should not require a specialized technical knowledge. Changes in the source code and recompilation should be avoided.

- *Replacement of leased and imported components*

It should be possible at any time to replace leased components by imported ones and vice versa to allow the client to buy a component when this becomes more advantageous than leasing (or vice versa).

- *Simultaneous use of different service providers*

Analogously it should be allowed to simultaneously deploy VPCs of different providers. The architecture should not confine the selection of the VPC providers. Each leasing decision should be independent of the others.

- *Platform independence of the components*

VPCs should be independent of a certain computing platform (i.e. independent of a certain hardware, operating system, or middleware) to facilitate their installation in the customer domain.

Performance

A VPC has to reduce the runtime overhead caused by the VPC mechanisms in the customer domain and in the interaction with the VPC provider. The functionality at the customer should be as lean as possible. For the interaction with the VPC provider, an efficient protocol is required. The frequency of interactions can be reduced by caching.

Security

Security is another important aspect of the VPC approach. It concerns the customer domain and the interaction with the VPC provider. The VPC architecture has to ensure that the VPCs do not endanger the customer domain. They should not provoke system breakdowns or influence the functionality of applications running in this domain. A secure communication between VPC customer and provider is complicated to realize, because this communication crosses domain borders. It has to pass the firewalls which protect the intranets borders. Furthermore, the communication has to include known security such as authentication, authorization, protection against data changes, data encryption, and auditing.

Further features

Additional features are required to facilitate the leasing process:

- *VPC exchange by plug&play*

The exchange of the service provider as well as of components should be allowed at runtime without interrupting or disturbing the application. The VPC platform should support plug&play mechanisms for a dynamic replacement of the service provider or other components. Plug&play techniques should be also used for automatic maintenance and updating of the bound services [Aage99], [Raza99].

- *Component market places*

In order to give the customer the opportunity to select among various service providers places are required where the customer can inform about the actual offers and make their choice. Service mediating entities are required to manage the available service offers and to support the customer in its selection process including contract negotiation. The implementation of such a market place does not belong to the VPC approach in closer sense. It is an infrastructure feature to support the concept.

In this paper we focus on architecture aspects. We discuss the functionality needed to install leased components in an application.

5. A VPC ARCHITECTURE

In the section we present a possible architecture to support Virtual Private Components. The proposed architecture is generic. It is independent of a certain middleware platform. Aspects of the implementation on existing middleware platforms are considered in the next section.

5.1 Location of the VPC proxy

An important aspect in the design of the VPC architecture is the location of the VPC proxies. In the VPR approach the proxy was placed in the customer domain (see Figure 2) to mainly hide the provider activities and to check the access rights of the VPR clients. This solution, however, is not obligatory, because most of the functions can also be accomplished using a provider based proxy [Opit00]. It makes no difference for middleware platforms like CORBA whether the client application accesses to an object at the customer or the provider side. For the VPC architecture, nevertheless a customer based solution has been chosen (see Figure 3) for the following reasons:

- Customer based proxies are more flexible in the communication with the VPC provider. They better support the selection of the most appropriate communication paradigm, e. g. to fulfill different security requirements.
- They make the communication with the VPC provider more efficient. An intelligent caching can shorten the response time by reducing the number of method invocations. Appropriate data compression methods can further increase the speed of the interaction.
- The VPC proxies can provide service specific graphical user interfaces in addition to their management interfaces.

All in all the customer based location of the VPC proxies gives more flexibility related to speed, security and operation. It also better reflects the idea of a virtual and private service [Opit00].

A VPC consists thus of two parts: the VPC proxy and its implementation at the provider side. The communication between the VPC proxy and the provider is determined by the provider. A VPC can provide several VPC interfaces to the VPC customer. These interfaces must be specified in the VPC contract and can be used from the customer for different applications (*VPC clients*). A client application can simultaneously access to several VPCs which can be provided by different providers. To unify the access to the VPCs we use like in other component concepts [Mons99], [Sieg00] a provider independent framework. This function has been assigned to the VPC manager (see **Error! Reference source not found.4**) which played a minor role in the previous VPR approach. The VPC manager handles the integration of the VPC proxies in the customer domain. It provides functions to install and to remove VPCs and to make needed assignments. The enforcement of the VPC contracts and the billing of the VPC use remain at the provider side as described in Section 3. We can apply here the same mechanisms as used for VPRs (see [Syrb98]).

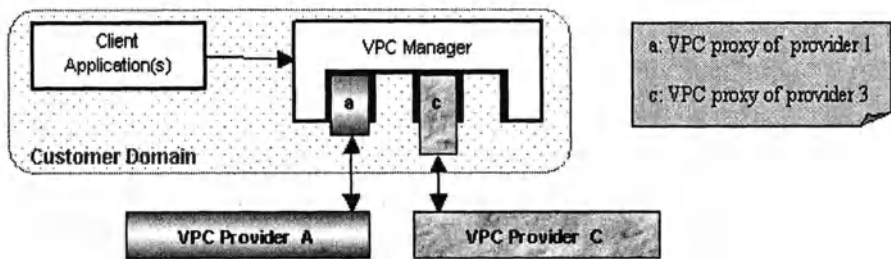


Figure 4. Integration of VPCs into the customer domain

The differences between the VPR and the VPC concept thus mainly concern the customer domain. The mechanisms at the provider side remain the same. Therefore we only consider the customer domain in the sequel.

5.2 VPC manager

The VPC manager comprises all procedures and interfaces to integrate the VPCs into the customer domain and to make the respective services available to the client application according to the requirements formulated in Section 4. Figure 5 shows the structure of the VPC manager. It consists of runtime environment to take in the VPC proxies, several management modules and 3 interfaces. These components are described next.

The **VPC runtime environment** is the container for the VPCs proxies. It represents the execution environment of the VPC proxies and provides the

access to various management functions to support the integration of the VPCs into the client application.

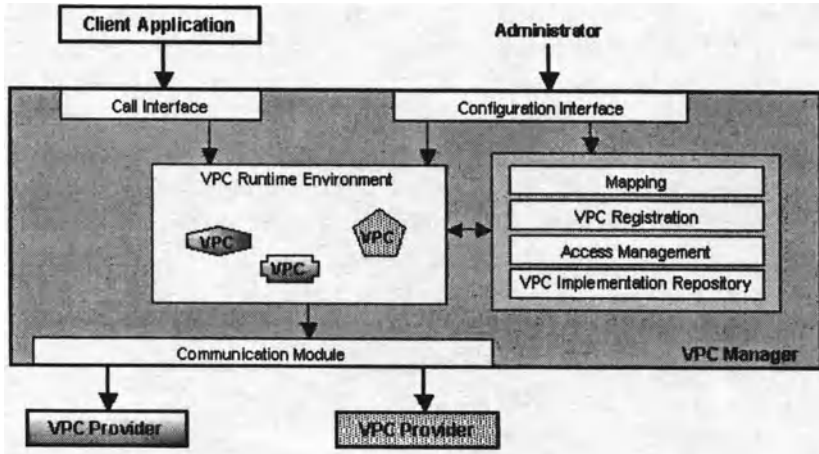


Figure 5. Structure of the VPC Manager

The **mapping module** has the task to map service IDs onto the corresponding VPCs or VPC interfaces, respectively (see Figure 6). Service IDs are used to facilitate the change of the VPC provider. A client application does not directly refer to a VPC. It uses a service ID. When another provider is selected a new VPC proxy has to be installed and the service ID is set to it via the configuration interface. The old proxies can remain in the VPC manager for the use in other applications. The mapping between service ID and VPC proxy is not always one-to-one, because several service IDs can refer to the same VPC proxy.

The **VPC registration** specifies the VPCs connected to the VPC manager. For each VPC, it contains an IDL description of the supported VPC interfaces. The registration is used for the mapping of the service IDs onto the VPCs.

The **access management** verifies the access rights of the client application to a VPC. The access rights are defined in the VPC contract. It is useful to connect the access rights with the service IDs. The inclusion of the access management into the VPC manager supports the easy change of the VPC provider, because the access rights must not be reset when installing a new VPC.

The **implementation repository** contains the VPC proxy implementations. When a VPC is called, the respective implementation is activated.

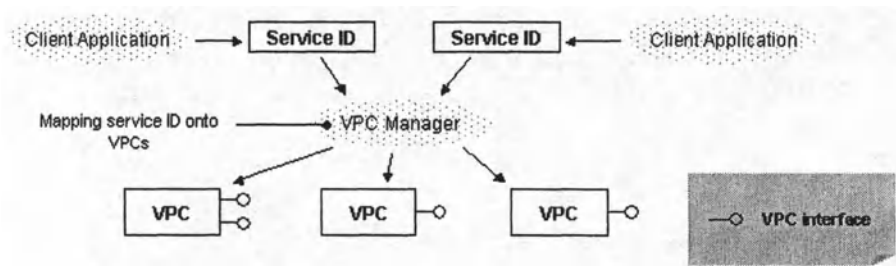


Figure 6. Mapping of the service ID onto VPCs

The **call interface** is the access point to the VPCs for the client application. The call is forwarded via the VPC proxy to the VPC provider.

The **configuration interface** serves for the installation and removal of VPCs. The interface can be used by the application designer to install VPC proxies, to configure the VPCs (if required), to assign service IDs to VPCs interfaces, and to determine access rights. The change of a VPC provider is also carried out via this interface.

The **communication module** is responsible for the interaction with the VPC providers. It can run different communication protocols. Depending on the applied security strategy several security levels can be applied to the communication with the VCP provider. The separation of the communication module makes the VPC proxies independent of the deployed communication protocol. So the VPC proxy only has to implement the access to the communication module.

6. PROTOTYPE IMPLEMENTATION

The VPC manager has been implemented as a prototype. For this implementation, we compared different middleware platforms to examine their suitability [Opit00]. We considered COM+, Java, Enterprise Java Beans (EJB) and CORBA. The limited space of the paper does not allow to describe this discussion so that we can only summarize the results of the comparison here. A COM+ based implementation would support the approach in many ways, but it is confined to Microsoft operating systems, in particular to Windows 2000. A pure Java implementation would require to implement the whole infrastructure. Enterprise Java Beans give the best support. An EJB container would also support the parallel handling of VPC calls. A CORBA implementation has the shortage that the components are not platform independent and would have to exist in several variants (see Figure 7A). We decided for a CORBA implementation with the use of a Java ORB. This implementation combines the benefits of CORBA with the

platform independence of Java (see Figure 7B). The selection was lastly also determined by the availability of licenses in our group. The implementation uses ORBIX 2.3 and JDK 1.3 including the contained Java ORB of Sun Microsystems.

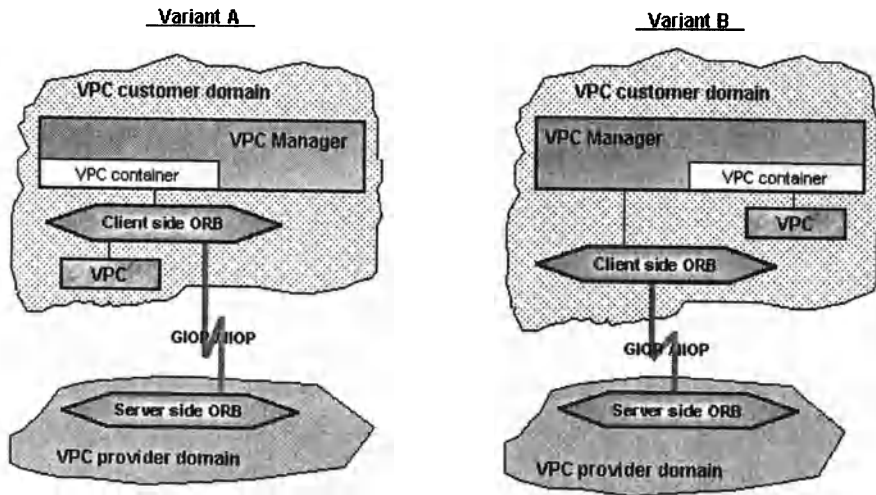


Figure 7. Implementation variants for the VPC manager using CORBA

The structure of the prototype implementation of the VPC manager is depicted in Figure 8. The white fields indicate the implemented classes which are shortly described below. The class *Communicator* has not been implemented yet. Next we give some details about the class implementations. The subtitles refer to the logical structure of the VPC manager of Section 5.2.

Call Interface: The class *CallInterface* connects the client application with the called VPC. It addresses to the runtime environment to receive the reference of a matching VPC. The *CallInterface* returns this reference to the client application which uses it for further calls of the VPC. The returned reference is a reference to a CORBA object. It is registered in the CORBA name service. The use of CORBA in class *CallInterface* makes the implementation of the client application independent of that of the VPC manager. This also concerns the selection of the programming language for the client application which can be freely selected.

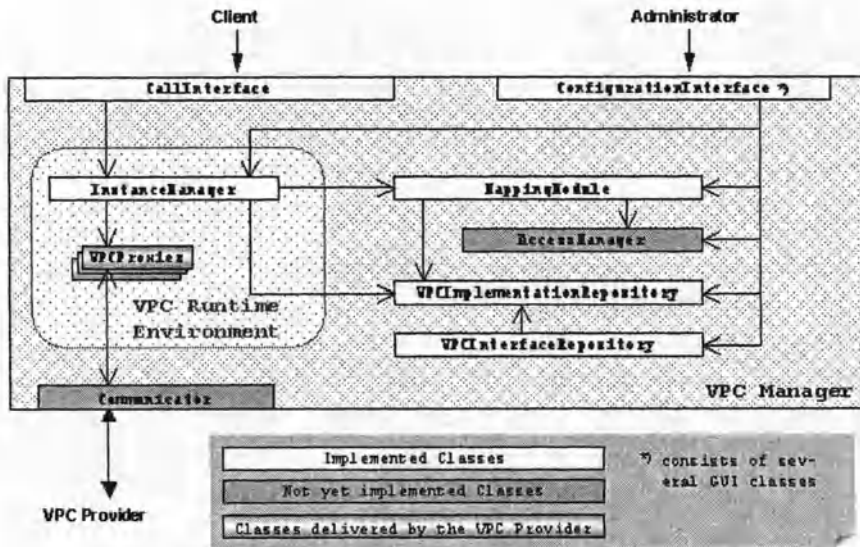


Figure 8. Structure of the prototype implementation of the VPC manager

Configuration interface: This interface consists of several GUI classes to allow the access to the different management modules. Via different windows the administrator can input all management data. It also provides a window for the instance manager to supervise the installed VPCs.

VPC Runtime Environment: The environment represents the kernel of the implementation. It contains the instance manager and the VPC proxies.

The **instance manager** implemented by the class `InstanceManager` administers the VPC instances. The class can be implemented differently using sophisticated management mechanisms. In the current implementation a simple strategy has been applied which creates a new instance for each invocation. When a VPC is called via the `CallInterface` it addresses to the `MappingModule` to map the service ID (see Section 5.2) on the matching VPC. After that the `VPCImplementationRepository` is called to determine from which VPC provider a VPC is supposed to be instantiated. The `InstanceManager` then creates the new instance. It also releases this link later on. To create a new instance the Java starting class of the VPC has to be instantiated first. This object creates a CORBA object for the VPC which is then used in the client application. To avoid name conflicts and to support runtime searches an own class loader is deployed for each VPC.

The **VPC proxies** are installed in the VPC Runtime Environment. The environment represents itself towards the VPC proxies as a set of interfaces via which they can access the different management modules and the communication module. If these interfaces are standardized, platform

independent VPCs can be defined. Note that it is not necessary to standardize the whole VPC manager, because the applications can principally use different managers, but the interface must be the same. In our prototype implementation the VPCs have free access to the Java ORB which is, however, not completely standardized. Therefore the VPCs are not allowed to use ORB specific extensions. Beside the CORBA interfaces the VPCs are only allowed to access to the Communicator.

Mapping Module: The class `MappingModule` implements the mapping of the service IDs onto a concrete VPC interfaces stored in the `VPCInterfaceRepository`.

VPC Interface Repository: The class `VPCInterfaceRepository` registers the VPC interfaces. For each interface, an IDL description has to be specified. The `VPCInterfaceRepository` generates the respective Java interfaces using the IDL compilers of the ORB. The registrations are persistent.

Access Manager: The class `AccessManager` checks the access rights to the VPCs for the given service IDs. It has not been implemented so far. The access rights are defined in the VPC contract. They relate to the conditions agreed between the VPC provider and customer and do not concern the principle work of the VPC manager.

VPC Implementation Repository: The class `VPCImplementationRepository` contains the implementation of the installed VPC proxies. It works similarly to the VPC interface repository. To install a VPC it is necessary to store the IDL descriptions and a Jar file with the Java classes. The IDL descriptions are used to generate the Java classes for the respective stubs and skeletons which fit to the used ORB. The Jar file contains the logics of the VPC proxy. One class has to be marked as starting class for the VPC manager to generate a VPC instance. Furthermore, the interfaces supported by the VPC have to be specified as well as a name has to assigned to the VPC. After that the VPC can be installed. All required steps are executed by the instance of the class `VPCRepository` in connection with the IDL compiler of the ORBs.

There are two possibilities to initialise the VPC at the provider. This can be either done when the VPC proxy is installed in the VPC manager or when the VPC is called for the first time.

Communicator: The `Communicator` implements the communication between VPC proxy and the VPC provider using the IIOP. Currently this is not contained in our implementation. The interaction of the VPC proxies with the providers is simulated.

7. FINAL REMARKS

In this paper we have presented with the Virtual Private Components an approach for the leasing of components. It extends the current component paradigm which is based on the purchase of components to leasing. Like in real life the leasing of components will give the user more flexibility and freedom. Component leasing is useful, if the purchase of components is (too) expensive or if topicality demands are very high. The Virtual Private Component approach extends the Virtual Private Resources concept. It enhances the applicability of the concept and makes its application more customer friendly and flexible. The strong dependencies between provider and customer were removed. It is much more easier to install a new VPC now. This allows the customer to adjust to changing market situations and to select other services when more favourable offers are available. The service provider, on the other hand, can make its relation to the service users more efficient to maintain and update the service.

We proposed a possible architecture to support the use of Virtual Private Components and described a prototype implementation of the VPC manager. Our next research steps aim at the completion of the VPC platform. Current research is focusing on the development of a security concept for VPCs. Future research will deal with refinements of the contract enforcement mechanisms, the integration of plug & play techniques, and resource accounting.

REFERENCES

- [1] [Aage99] Aagesen, A.; Helvik, B.: Towards Plug&Play Architecture For Telecommunications. In Kutvonen, L.; König, H.; Tienari, M. (eds.): Distributed Applications and Interoperable Systems II. Kluwer Academic Publishers, 1999
- [2] [ISO97] ISO, IS 13235-1, ODP Trading Function (1997)
- [3] [Miel00] Mielke, T.: Deployment and Evaluation of Virtual Private Resources. Diploma thesis, BTU Cottbus, Department of Computer Science, 2000 (in German)
- [4] [Mons99] Richard Monson-Haefel, R.: Enterprise JavaBeans™. O'Reilly & Associates, 1999
- [5] [OMGS98] OMG, „CORBAServices: Common Object Services Specification“, <ftp://ftp.omg.org/pub/docs/formal/98-12-09.pdf> (1998)
- [6] [Opit00] Opitz, A.: Design and Implementation of a VPC kernel. Technical Report 10/00, BTU Cottbus, Department of Computer Science, 2000 (in German)

- [7] [Preu97] Preuß, T.; Syrbe, J.-H.; Koenig, H.: *Virtual Private Resources - An Approach for Long-term Binding of Services*. In Proc. IEEE EDOC, Brisbane, IEEE CS Press, 1997, pp. 216 - 226.
- [8] [Preu99a] Preuß, T.: *Long-term Service Supplier Chains for Electronic Markets*. Ph.D thesis, BTU Cottbus, Department of Computer Science, Shaker, Aachen, 1999 (in German)
- [9] [Preu99b] Preuß, T.; Syrbe, J.-H.; König, H.: *Permanent Customer-Provider Relationships for Electronic Service Markets*. In Steinmetz, R. (eds.): *Kommunikation in Verteilten Systemen*. Springer, 1999, pp. 326 - 337.
- [10] [Raza99] Raza, S. K.; Bieszczad, A.: *Network Configuration with Plug and Play Components*. In *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, 1999.
- [11] [Sieg00] Siegel, J.: *CORBA 3 – Fundamentals and Programming*. Object Management Group, John Wiley & Sons, Inc., 2000
- [12] [Syrb98] Syrbe, J.-H.; Preuß, Th.: *An Enterprise-level Analysis of Customer-Provider Relationships & Implications for QoS Enforcement*. In Proc. IEEE EDOC, San Diego, IEEE CS Press, 1998.
- [13] [Szyp98] Szyperski, C.: *Component Software – Beyond Object-Oriented Programming*. Addison-Wesley, 1998
- [14] [Wimm99] Wimmers, M.; Mester, A.; Krumm, H.: *A Component Framework for the Configuration Management of Networks*. In Kutvonen, L.; König, H.; Tienari, M. (eds.): *Distributed Applications and Interoperable Systems II*. Kluwer Academic Publishers, 1999

A New Organisational Framework for Network Modelling Using a Multi-Agent System

Marc Lemerancier and Dominique Gaïti

¹ *Laboratoire de Modélisation et de Sécurité des Systèmes (LM2S)*

University of Technology - Troyes (UTT) – France, 12 rue Marie Curie, BP 2060, Troyes Cedex

Email : { marc.lemercier, dominique.gaiti }@utt.fr

² *LIP6*

University of Paris 6, 8 rue du Capitaine Scott, 75015 Paris – France

Email : dominique.gaiti@lip6.fr

Abstract In this paper, we present a platform called MadKit that allows the generic development of multi-agent system based on the organisational concept. Three main objects are defined: agents, groups and roles. We propose to illustrate this concept of organisation in a telecommunication environment. Our main objective is to show the possible use of the MadKit platform in the context of network modelling. In a first approach, we propose a model and implement the superposition of ATM traffics with the help of the MadKit synchronous engine. In a second and third approach, we have modelled the active network architecture (ANTS) and a LEO telecommunication satellite network. We propose the modelling of our three examples by organisational structures. These solutions fill the gap of the actual modelling processes when the network becomes complex (different Quality of Service, resources management, ...) and dynamic.

Keywords: Network modelling, multi-agent platform, ATM, Active Network, Constellation of satellites.

1. INTRODUCTION

The increase of throughput proposed by current telecommunication systems makes network evaluations difficult to realise. At the same time, users ask for a real Quality of Service (QoS) and for more and more services.

More traffic, a better QoS and a large service offer are not design to work in a coordinated manner in the actual network. New concepts such as the active network, able to modify the equipment behaviours, brings a new dimension in the dynamic of the network and a response to our problem. But nowadays, no simulator is able to handle the changes of network behaviours in face of a breakdown, a routing modification or an intelligent flow control able to adapt itself to the network state. It is obvious that we have to analyse systems in a dynamic manner and no more in a static manner. All these evolutions are difficult to handle. Many simulation environments allow the modelling of telecommunication systems [OPNET 2001], [HyPerf 2001], [Modline 2001], [NS2 2001]. But these techniques have limits when the environment becomes dynamic.

One of our project named MACSI [RNRT 2001] has for purpose to study and to propose new methods for modelling and simulating telecommunication networks. These methods will take into account the dynamic aspect and the evolution of networks as well as the notion of behaviour. A second objective will be to give an evaluation of the quality of results in this dynamic context. Our project has to propose a new methodology of network simulations taking into account the environment by using a multi-agent approach. This solution considers that users' traffics and the behaviour of the network result from interactions between the different elements (actors) in a telecommunication system. We choose to design a model of the system by different actors (agents of modelling) and to simulate this system. Each actor is autonomous and is defined by its knowledge and its interactions.

In this paper, we focus on the organisational approach to model a telecommunication model. Behaviour modelling constitutes another approach and is not the purpose of this proposal. The remainder of this paper is organised as follows. Section 2 presents a multi-agent platform called MadKit. This platform is useful to describe our system in an organisational manner. We show different concepts used by MadKit and a way to simulate complex systems. Some applications in a network environment are proposed in section 3. Three examples are proposed and described: a simulation of the ATM traffic, a model of the ANTS active network and an organisation of a constellation of satellites. Section 4 concludes this paper.

2. THE MADKIT PLATFORM

Several multi-agent platforms have been proposed allowing the development of complex system with the help of agents. The main insufficiency of these approaches is the lack of an organisational structure

for the agents. Some researchers [Ferber 2000] have proposed a multi-agent platform named MadKit based on three concepts: agent, group and role. The generic development of a multi-agent system and the agents' organisation constitute the central proposal of this platform [Madkit 2000]. Two structure levels are proposed: the group and the role. An agent belongs to one or several groups, and inside a group an agent can play one or several roles. A role can be seen as a particular function of an agent.

From the agents' cooperation point of view, these organisation concepts allow to structure dialogues between agents. An agent can communicate directly with an other agent identified by its address or can broadcast the same message to each agent with a given role in a group.

2.1 Architecture of the MadKit platform

The platform has been built with respect of the conceptual model based on notions of agents, groups and roles. It is based on a Java little kernel responsible of the management of agents with the help of three components: a "*group and role manager*", a "*synchronous engine*" and a "*local messaging*". To lighten the kernel, a certain number of its services has been agentified (i.e. transformed into MadKit agents), such as the "*GraphAgent*" or the "*AgentLister*" of the madkit distribution [Madkit 2000]. The independent graphic interface is based on the Java Beans specification. Thus, each agent is responsible for its graphic interface. It is possible to use several functional modes for the platform such as simple consoles, Java applets, or the MadKit default environment: the G-box [Gutknecht 2000]. It allows to launch the execution of agents and to visualise their behaviours.

2.1.1 Creation of agents

The MadKit platform proposes useful services for the creation of generic agents using the Java class *Agent* of the MadKit kernel. These agents constitute the solution for the development of a multi-complex agent system. The agent initialisation in term of groups/roles is described in the method *activate()* automatically launched by the MadKit kernel. The manager of the system has many primitives grouped in the Java API of the platform that will be able to be called from the method *activate()*. Thus, an agent can create and integrate groups and roles, has the capacity to launch and afterwards to kill others agents. This method *activate()* can be seen as the constructor of the agent. Then the agent activity is described by its method *live()*.

In its kernel, the MadKit platform integrates several message formats allowing dialogues between agents. The approach followed by MadKit is both textual messages or document, classically used in an object context

(StringMessage class or XMLMessage class) and more structured messages from the artificial intelligence area (ActMessage class, ACLMessage class and KQMLMessage class).

A method allows all agents to send a message, directly to another agent whose address is known with the primitive *sendMessage()*, or to broadcast this message to all agents with a given role in a group using the method *broadcastMessage()*. Concerning the reception of messages, each agent has a mailbox where received messages are memorised. Two primitives of low level allow the access to messages. The non-locking primitive *isMessageBoxEmpty()* looks into the mailbox and the locking primitive *nextMessage()* extracts and returns the first message from the mailbox.

We used the development kit of the MadKit platform to illustrate its main concepts: the notion of organisation brought by the membership to groups and roles, the exchange of messages between agents.

2.1.2 Remarks and statement

The elements used by MadKit for the transfer of information between agents imply that multi-agent systems constructed from this platform are characterised by a total distribution of knowledge, partial results and methods used to get a result. In this case, agents have a view more or less precise of the other agents of the system. They have therefore to be able to get and to represent intentions and commitments of the other agents. The main difficulty of this approach is generally the representation of each agent's knowledge as well as the update of this knowledge [Pujolle 2000].

Agents extending the MadKit *Agent* class are executed inside a Java thread. Although this approach constitutes a less expensive solution than to process classical systems, the execution in parallel of several hundreds of agents is not foreseeable. This solution is useful only for small applications of multi-agent systems. In a complex system simulation context, it is necessary to have another kind of agent, closer to the notion of a Java object, which do not need the use of a thread. The following paragraph presents solutions proposed by the MadKit platform in the case of a complex system in general or a multi-agent simulation.

2.2 A MadKit proposal for complex simulations

A new architecture called "*The Synchronous Engine*" is proposed in the platform. It is based on five elements:

1. The ReferenceableAgent. In the framework of the synchronous engine, an agent can be planned or monitored. It is necessary to have a reference allowing a direct access to the agent what was not possible for security

reasons with the first release of MadKit. Therefore, the designer has to specify these intentions by using the *ReferenceableAgent* class;

2. The Scheduler agent. The Scheduler agent is a standard MadKit agent that has to manage all *Activator Agents* for the execution of synchronized agents;
3. The Activator class. The Activator tool defines a planning policy. It works with the *Scheduler agent* to obtain the list of Referenceable agents. An *activator* belongs to a group and has a role. Its method *update()* allows it to discover dynamically the agent implementation from a given group and role. The *Scheduler agent* will invoke its method *execute()* to launch the execution of the simulation;
4. The Watcher agent. The *Watcher agent* manages a list of probes. This is not a threaded agent as the *Scheduler agent*, but it has therefore to be executed in association with a scheduler;
5. The Probe class. The *Probe* class allows defining a code for the exploration of a *ReferenceableAgent*.

Figure 1 presents elements of this synchronous engine such as it is explained in the MadKit documentation [Gutknecht 2000].

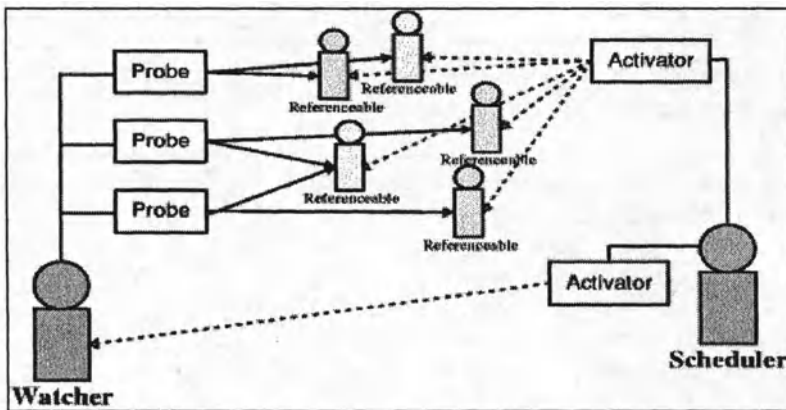


Figure 1. MadKit synchronous engine

3. APPLICATIONS TO NETWORK

In the context of our MACSI project [MACSI 2001], we wish to conceive a simulator able to take into account both the complexity of current high throughput networks as well as an active network in which the behaviour of equipment can be modified by the traffic encountered. The

main objective of this study is to validate if possible the use of the MadKit platform for complex network simulations.

This paragraph presents a model and an implementation of ATM traffics with the help of light agents (*ReferenceableAgent*) of the MadKit synchronous engine. We then model the active network ANTS and a constellation of satellites.

3.1 Example 1 : ATM traffic

To illustrate the possible use of the MadKit synchronous engine in a network context, we have chosen to simulate a superposition of ATM traffics. In an ATM network, time can be split in time-slots equal to the time of a cell emission. The chosen traffic model is a Bernoulli source with parameter P ($P \in [0, 1]$). Thus, to each time-slot, a source sends an ATM cell with a probability P . MadKit Referenceable agents represent these sources.

3.1.1 Step 1: creation of a Bernoulli class

Figure 2 presents the code of an agent implementing the Java *ReferenceableAgent* interface that represents a Bernoulli(p) source. The variables (*emission* and *cells*) are defined as attributes of the class to be then observed with the help of probes (see step 4).

```

public class Bernoulli extends AbstractAgent implements ReferenceableAgent {
    public double parameter = 0.0;
    public int emission, cells;
public Bernoulli(double p) { parameter = p; }
public void activate () {
    joinGroup ("ATM");
    requestRole ("ATM", "Bernoulli");
}
public void walk () {
    double draw = Math.random();
    if (draw <= parameter) { emission = 1; } else { emission = 0; }
    cells = cells + emission;
}
    public void setParameter (double p) { parameter = p; }
}

```

Figure 2. Bernoulli agent

The method *activate()* insures the recording of the agent in the *ATM* group with the *Bernoulli* role. The *walk()* method represents really the traffic: to each call of this method, we undertake a random draw that is then compared to the value of the parameter *p*. The attributes *emission* and *cells* are updated correspondingly.

3.1.2 Step 2: The activator

Figure 3 shows the implementation of the activator agent. The role of this agent is to execute a slot-time of simulation by invoking the methods *walk()* of all Referenceable agents already defined. The platform API memorises to this end the set of agents in a table. The *setParameter()* method allows to change the parameter value of Bernoulli sources during the simulation.

```
public class ATM_Bernoulli_Activator extends Activator {
    public ATM_Bernoulli_Activator (String group, String role) {
        super (group, role);
    }
    public void execute () {
        for (int num = 0; num < agents.length; num++)
            ( (Bernoulli) agents[num] ).walk();
    }
    public void setParameter (double p) {
        for (int num = 0; num < agents.length; num++)
            ( (Bernoulli) agents[num] ).setParameter (p);
    }
}
```

Figure 3. Activator code

3.1.3 Step 3: The Scheduler

The Scheduler agent possesses an essential function in the synchronous engine. The code (figure 4) of the method *activate()* presents its main responsibilities: record an agent in the group "ATM" with the role "scheduler", creation of all Bernoulli agents using the *lauchAgent()* primitive, then creation of an observer (see step 4). The method *live()* presents the scenario of simulation. The method *execute()* of the *sources Activator* and the *Observer Activator* are launched in a loop until the end of the simulation.

```

public class MySchedulerAgent extends Scheduler {
    int sourcesB = 10, delay = 100;
    double bparameter = 0.1;
    boolean end = false;
    ATM_Bernoulli_Activator a1;
    MyWatcher mObs;

    public void activate() {
        foundGroup ("ATM");
        requestRole ("ATM", "scheduler");
        for (int indice = 0; indice < sourcesB; indice++) {
            launchAgent (new Bernoulli (bparameter), "B", false);
        }
        mObs = new MyWatcher();
        launchAgent (mObs, "my_watcher", true);           // true = GUI
    }
    public void SetParamB (double p) {
        a1 setParameter (p);
    }
    public void live () {
        a1 = new ATM_Bernoulli_Activator ("ATM", "Bernoulli");
        SingleMethodActivator a2 = new SingleMethodActivator (
            "watcherBernoulli", "ATM", "watcher");
        addActivator (a1);
        addActivator (a2);
        update();
        while (!end) {
            pause (delay); a1.execute(); a2.execute();
        }
        mObs.stat();
    }
}

```

Figure 4. The scheduler agent

3.1.4 Step 4: the watcher

This agent has to observe the Referenceable agents with the help of probes which accumulate statistical data on some Referenceable agent attributes. The code of this agent (figure 5) allows us to see that two numerical probes observe *emission* and *cells* attributes of all Bernoulli sources.

```

public class MyWatcher extends Watcher {
    NumericProbe p1, p2;
    int slotATM = 0;
    int pb[] = new int [20];
    public MyWatcher () {
        p1 = new NumericProbe ("emission", "ATM", "Bernoulli"); addProbe(p1);
        p2 = new NumericProbe ("cells", "ATM", "Bernoulli"); addProbe(p2);
    }
    public void activate () {
        joinGroup ("ATM"); requestRole ("ATM", "watcher"); update();
    }
    public void watcherBernoulli () {
        int nbe = (int) p1.getSum();
        slotATM++;
        if (nbe < 20) { pb[nbe] = pb[nbe] + 1; }
        println ("slot[" + slotATM + "]:Ar=" + nbe + " total=" + p2.getSum() );
    }
    public void stat () {
        for (int indice = 0; indice < 20; indice++)
            println ("P(Ar=" + indice + ")=" + (float) ((float) pb[indice] / slotATM));
    }
}

```

Figure 5. The watcher Agent and the probes

3.1.5 Remarks and statement

This task describes the process of the MadKit synchronous engine in a network context. It is possible to complete this simulation by the definition of others ATM traffic models activated by new Activator agents. In order to complete a multi-agent simulation, the ATM switches would have to be represented by MadKit standard agents able to exchange management information on their states and their behaviours.

The main element to retain is the fact that Referenceable agents are successively activated by an Activator agent. Thus, to each simulation step, each agent executes its task defined in the method *walk()*. The real-time of the simulation step corresponds therefore to the sum of the execution time of all the methods *walk()*. By defect, there is no constraint of time for an agent. The agent has therefore to take a quick decision on the partial realisation of its activity using this method.

It is important to notice that the behaviour of the agent (or its activity) is represented by its *walk()* method that is then invoked to each simulation time unit. The activity of the agent has therefore to be repetitive or to be split into N sub-tasks achievable successively to each call by supposing that the agent knows its execution context.

In practice, in a context of simulation, *Referenceable agents* are messages, capsules in an active network context or cars in a car-traffic simulation. The benefit of the MadKit platform in this approach is the notion of probes able to get quantitative data on Referenceable agents in a very fast way.

3.2 Example 2 : ANTS Active Network

The ANTS active network architecture is a proposal of the MIT where capsules (packets containing data and code) are sent by users' applications and executed in the active nodes of the network. In practice, an active network is a network constituted with traditional IP routers and active nodes having an execution environment. For security reasons, the designers of ANTS have limited the possibilities offered to user's traffics. The manager of the active network develops and installs a certain number of generic routines identified by references. The ANTS capsule is in fact a packet containing data and references to routines of the active network. These capsules are grouped in protocols that are units of management for the system [Wetherall 1998].

Capsules of an active traffic are automatically executed on all active nodes located between the transmitter and the receiver. A mechanism based on mobile code insures the migration of routines in case of necessity. More precisely, a capsule containing the address of the receiver is encapsulated in an IP packet whose address of destination corresponds to the address of the next active node to cross.

An active node proposes some forwarding functions to find the road to the next active node. The development of a service by a user necessitates the definition of the different capsules and routines. Active nodes propose a set of primitives usable by all capsules using a Java API. An example of service can be information caching, a forwarding policy, etc. Other function of a node is to manage resources consumed by a capsule. Each capsule comprises a field "*resources remaining*" in relation to the resource level of the capsule. Active nodes decrease the number of resources consumed by each capsule. A capsule having no more resources is destroyed.

3.2.1 A MadKit model for an ANTS network

We present in this sub-section a MadKit model for an ANTS network. The ANTS network is a classical IP network with standard or active routeurs and a set of telecommunication links. On the network circulate both IP packets and capsules. We propose therefore to represent each node of the ANTS network by a MadKit standard agent. For passive nodes, a classic

MadKit agent is used belonging to the *PassiveNode* group and having the *manager* role. Similarly, for each active node, we create a classical agent belonging to the *ActiveNode* group and having a *manager* role.

IP packets and capsules are numerous in the system, and we choose therefore to represent them by MadKit Referenceable agents. An IP packet is a light agent belonging to the group *UserMessage* and playing two roles: *passivePacket* and *traffic_ID*. The first role is to distinguish a passive packet from an active capsule and the second role is to identify the traffic to which belongs this data-message. The *traffic_ID* identifier can be built from the knowledge of IP address of the transmitter and the receiver. A capsule is therefore a light agent of the group *UserMessage* and has the roles of *activePacket* and *traffic_ID*.

In an ANTS network, each active node can receive many packets and capsules from an active traffic. We know that the capsules' context of execution is independent between traffics. We have chosen to create another MadKit agent for each active traffic crossing an active node.

The role of this new agent (figure 6) is to manage all packets and capsules of its associate active traffic, and to be in fact the delegate of this traffic beside the *manager* agent of the active node. This delegate is able to characterise and define the behaviour of the active traffic with the help of probes, and to negotiate network's resources beside the agent *manager*. This delegate is therefore a standard MadKit agent of the *activeNode* group with *AN_ID* (active node ID) and *traffic_ID* roles. The identifier *AN_ID* designates an active node. Thus, an active node is represented by an agent *manager* and possesses one agent per active traffic crossing it. This model leans on structures of society that we present in figure 7.

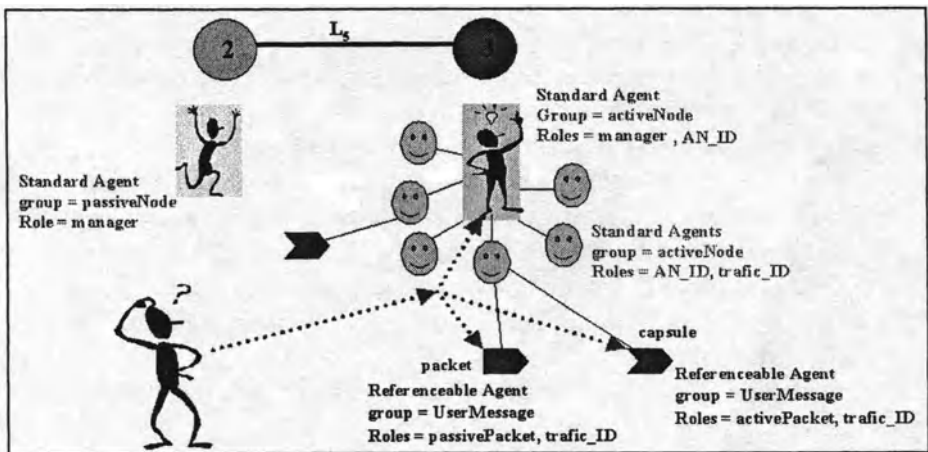


Figure 6. An ANTS active node MadKit model

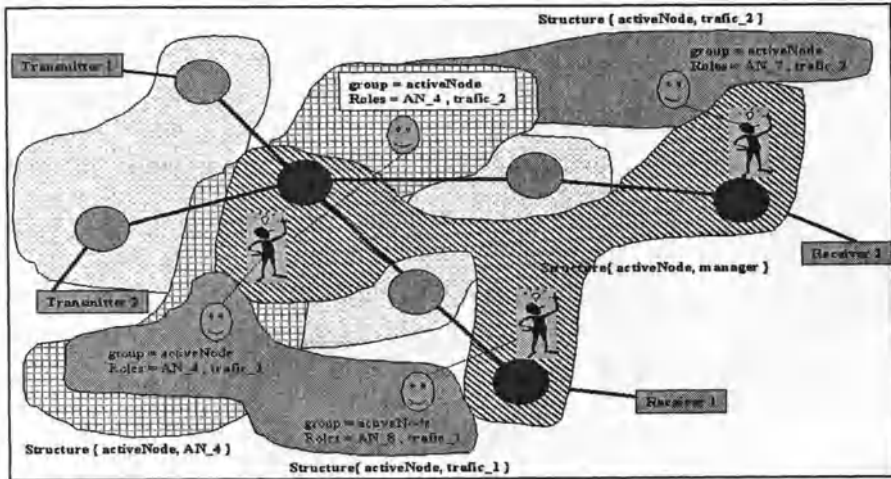


Figure 7. Multi-agent organisational structures

Five entities can be found:

1. IP packets and capsules from an active traffic. Packets and capsules of a same active traffic belong to the same *UserMessage* group and possess in common the *traffic_ID* role. There are Referenceable agents on which a *MadKit Observer agent* can put some probes;
2. Active traffic. In each active node crossed by an active traffic, an agent {group: *activeNode*; roles: *AN_ID* and *traffic_ID* } is created by the *manager agent* of the node. Although different manager agents on different active nodes create agents for a same communication, they take part of multi-agent organisational structures. This proposal allows the possibility of routine exchange proposed in the ANTS architecture. Agents can also represent a same active traffic to exchange messages of management. There are therefore as many structures of delegates as open active traffics in the network;
3. Active node (*AN_ID*). The *manager* agent and all active traffic delegates belong to the group *activeNode* and have an *AN_ID* role. This solution allows organising data traffics both between the *manager* and delegates, and between delegates. This last possibility is interesting because in the ANTS architecture no interaction between the capsules is anticipated;
4. Passive node. Passive nodes belong to the *passiveNode* group and have a *manager* role;
5. Active nodes. Similarly, a set of active nodes belongs to the *activeNode* group and has a *manager* role.

3.2.2 Organisational model for an active node

This model of an ANTS network leans on organisational concepts of the MadKit platform. We have associated to each element of the ANTS architecture, an agent or a group of agents with respect to the main constraints of the ANTS network. We can see that the use of the MadKit development kit allows constructing a simulation of this active network. For that, we need to describe the network topology, and some scenarios of simulation on this network. These descriptions can be written in a XML document to insure an inter-operability between several multi-agent systems (MadKit, DIMA, etc.).

All along this paper we focus on the organisational approach. Another approach concerns the behaviour-based model of an active node. This approach is currently under study in the University of Technology of Troyes and in the University of Paris 6 [Merghem 2001]. A part of activities of an ANTS node can be represented by a reactive approach. Indeed, the *manager* agent has to arbitrate traffics' requests from all active traffic delegates. However, it seems interesting that this *manager* be able to represent the behaviour of its node and to construct a view of the behaviour of the other active nodes of the network. In order to do that, the MadKit platform proposes the use of a cognitive agent (JessAgent). This improvement is for the moment under integration to the platform. An expert system engine is integrated to a traditional agent. The behaviour of an agent can thus be described by a set of rules.

3.3 Example 3 : Constellation of satellites

The satellite network model is inspired by the one proposed by the French national centre for the space research (Centre National de Recherche Spatiale (CNES)) for our project 'Constellation of Satellites for Multimedia applications'. The 72 LEO satellites are equally distributed among 9 orbits of radius 1603 km and 50 degree equatorial inclination, and have a minimum elevation of 17.5 degrees. Each satellite is equipped with up and downlink transceivers of 155.5 Mbit/s bandwidth, and 4 bidirectional intersatellite links (ISL) also of 155.5 Mbits/s [Sigel 2000].

Thus each satellite has a permanent connection with the previous and the next satellite on the same orbital plan, and two other satellite links to its left and its right. These two last links are not permanent. Several roles are necessary to model the organisation of agents. We have chosen the following notation:

- a) O_{id}: orbital plan number id (id in [1,9])
- b) S_{xz}: satellite in the orbital plan x (x in [1,9]) and rank z (z in [1,8])

c) N_s: neighbourhood of the satellite S

Figure 8 presents five other MadKit agents that we propose to describe the neighbourhood of a satellite. The definition of satellite neighbourhoods produces a great number of roles. We are going to begin with a description of agents located on one satellite (in our example the satellite number 44). The first agent is a creator agent (C) that is the superintendent of the role N_44 (neighbourhood of the satellite 44). This agent is a MadKit agent that belongs to the satellite 44 (group1, role S_44) and does not participate to the communications between satellites (group1, role OUT). It is the creator of the neighbourhood 44 (group2, role C and N_44). The second agent is an agent P (PREVIOUS), located on the satellite 44 (group1, role S_44) but belonging to the neighbourhood of the satellite 43 (group2, role P and N_43) that is the previous satellite on the same orbital plan. This agent is the delegate of the neighbourhood N_43 centred on the satellite 43. So, a satellite has one delegate for each four neighbours. The third agent is an agent N (NEXT). It is the delegate of the neighbourhood of the satellite following on the same orbital plan (N_45). And, the last two agents are delegates of the two closer satellites on others orbital plans.

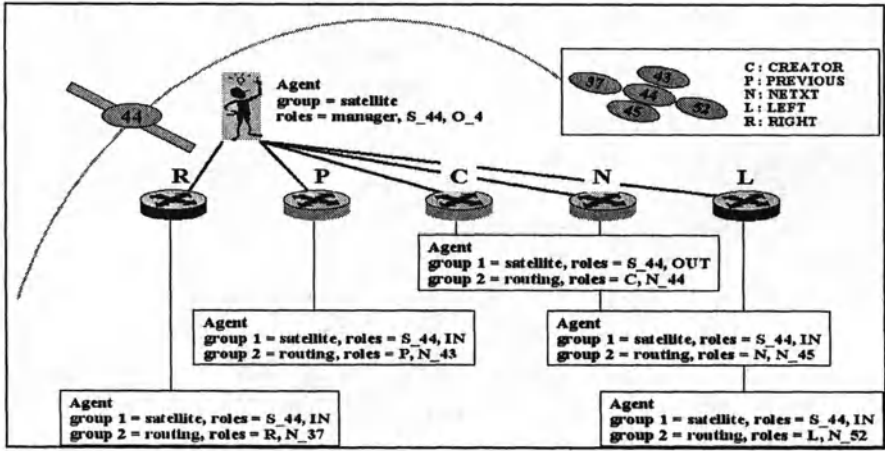


Figure 8. MadKit agents for neighbourhood definitions

This set includes the satellites 44, 37, 43, 45, 52. These agents belong to a first group *satellite* allowing a precise identification of these agents (membership to a given satellite), and belong also to a second group *routing* that identifies the function of this agent in the definition of a neighbourhood.

This model defines several MadKit organisational structures and allows the information exchange between agents of even groups with the same role. We notice that this representation is extremely rich and allows thus to define several groups of distribution

4. CONCLUSIONS

In this paper, we proposed the use of the MadKit multi-agent platform in our telecommunication system context. We have presented our main objectives namely the organisational modelling of complex network configurations. This approach allows the conception of several societies of agents in order to organise exchanges of messages without any disruption for all the agents in the simulation.

Other studies have to be performed on the area of the network device modelling. Indeed, the MadKit platform proposes only the language Java for the development of agents and does not propose libraries for network equipments. We work currently to the construction of this generic library usable with MadKit and also with others multi-agent systems. As presented here, the problem of the network behaviour has not been described. Indeed, we work now on the definition of a set of behaviours (careful, careless, etc.) allowing the description of the intern functioning of a network router [Merghem 2001]. Then, a high-level manager agent knowing these behaviours will be able to alter the policy of a router according to criteria of traffic QoS. The final goal of our study is to provide an agent-based network simulation techniques useful when the network becomes dynamic.

REFERENCES

- [15][Ferber 2000] "Multiagent systems for telecommunications: from objects to societies of agents", Jacques Ferber, networking 2000, Paris.
- [16][Gutknecht 2000] "MadKit Development Guide", Olivier Gutknecht, MadKit documentation, version 2.0. <http://www.madkit.org>
- [17][HyPer 2001] <http://www.hyperformix.com/products/products.htm>
- [18][MACSI 2001] <http://www-lm2s.utt.fr/macsi/>
- [19][Madkit 2000] "Madkit official web site", <http://www.madkit.org>
- [20]<http://community.madkit.org/>
- [21][Merghem 2001] Leila Merghem and Dominique Gaïti, "Active Network Modelling and Simulation: a Behavioural Approach", Smartnet 2002, Finland.
- [22][Modline 2001] <http://www.simulog.fr/>
- [23][NS2 2001] <http://www.isi.edu/nsnam/ns/>
- [24][OPNET 2001] <http://www.opnet.com/products/modeler/home.html>
- [25][Pujolle 2000] Guy Pujolle "Les réseaux", 3^{ème} édition, book in french. Eyrolles 2000.

- [26][RNRT 2001] Official Web Site of RNRT projects
<http://www.recherche.gouv.fr/technologie/reseaux/rnrt.htm>
- [27][Sigel 2000] Eric Sigel, Bruce Denby, Sylvie Le Hégarat-Masclé, "Application of ant colony optimization to adaptive routing in a LEO telecommunications satellite network", IEEE Transactions on Networking, July, 2000
- [28][Wetherall 1998] David J. Wetherall, John Guttag, and David L. Tennenhouse "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", IEEE OPENARCH'98, San Francisco, CA, April 1998.

I3- A New Intelligent Generation for Internet Networks

Nadia Boukhatem¹, Benoit Campedel², Hakima Chaouchi³, Vincent Guyot^{1,3}, Francine Krief⁴, Thi Mai Trang Nguyen^{1,3}, and Guy Pujolle³
1 ENST, 2 ISEP, 3 LIP6, 4 LIPN

Abstract This paper introduces new concepts to be applied for future Internet generations to better manage, react and control the network. These concepts are based on the intelligence aspect and the policy-based networking management paradigm. We first describe a policy-based networking management environment and some new protocols allowing an easy control of the QoS and the mobility. Then, we describe the Internet Third Generation (I3) platform that is our suggestion for an intelligent agent-based implementation of the policy based networking management.

Keywords: Network Policy Management, Policy server, PDP, PEP, COPS, SLS, SLA, COPS-SLS, Mobile IP, Multi-Agent system, AALAADIN, MADkit Platform.

1. INTRODUCTION

During the 80s, most academic centers exchanged messages via the "Net". The period 1990 through 2000 saw the arrival, for the public at large, of commercial offers allowing communications with high-speed servers. This was the Internet called Internet First Generation (I1). The Internet I1 is the system that we are using nowadays from our terminals. The Internet called I2, Internet Second Generation arrived at the beginning of the year 2000. A new Internet, called Third Generation (I3) of which primary concepts are described in this paper.

I1 comes from the initial work achieved by some scientists in the world believing in a universal communicating system. However, in that Internet throughputs were too slow, servers do not offer any guarantee to deliver the desired information. At the end of the 90s, the promises of increase of the

throughput are real; costs become almost accessible and many new and attractive services appeared.

I2 arrives with its high throughput, its procession of services, its best stability against breakdowns and its ubiquitous access.

Between 2005 and 2010, the passage to I3 intends to lead to a more stable system and protocols, imperceptible breakdowns to the user, a large number of available multimedia applications, and a generalized access.

What directions to push on? The intelligence. The computer intelligence debate is not new. We believe that the system must be intelligent because the distributed nature of the network management requires an additional component: the time; the system must react on real time to unexpected situations and has to make decisions and adapt itself to a new situation and to control critical states automatically.

Intelligence is the chosen word to group several actions well distinct: to learn, communicate and infer. Several ways can be envisaged, with their own roots: Intelligent Networks (IN), smart networks and active networks. Intelligent Networks form an already old architectural concept where the intelligent word meant adaptation. The system has to be adaptable to the user demand. Even if some experiences were tested in telecommunication systems, we are still only in the first steps. In this area, the data-processing could induce a too large delay when using languages like Java that disseminate pieces of software in a very large number of points. These pieces of software can possibly move by themselves.

The second field is supported by smart networks. It is necessary to develop specific components to bring their expertise to solve problems and to control infrastructures and flows. As a first answer, there are some known solutions: they come from the intelligent agents field. An agent is an autonomous entity able to communicate with other agents, to perceive and to represent its environment. The whole agents in interaction form a multi-agent system. Many criteria allow to classify these systems: the size of agents, the number of agents in interaction, the mechanisms and the types of communication, the behavior, the organization and the representation of the environment.

The third area corresponds to active networks. The intelligence in active networks, situated inside the nodes of the system, as well as in the user workstations, can adapt the nodes and the workstation to the type of information received. For example, the header of the packets can contain programs that must be executed in the routers. The nodes have to fit instantaneously to the arriving packet, what seems possible with new reconfigurable processors.

I3 research project focus on an intelligent-based control scheme belonging to the second class mentioned above (smart networks). The basic

problem concerns the architecture of the system to integrate the "intelligent" components, so that they could be simple and cooperating to achieve the policy-based network management.

This paper presents the development of an intelligent platform based on a policy-based networking management system. Section 2 introduces policy concepts. Section 3 deals with a new protocol for controlling the configuration of hosts. Section 4 introduces the policy based mobility management within the global system. Then, section 5 is introducing the I3 model and platform

2. AN INTRODUCTION TO POLICIES

Policies can be defined as a "set of rules to manage and control access to network resources". The works dealing with policies come from different domains, sharing common needs. The most advanced works are found in the quality of service (QoS) domain, with the definition of a new protocol (COPS) and a new architecture. To make this specific approach more general, a working group has been created by the IETF to specify the information model to be used, as well as the general architecture. The goal of the information model is to define a general model that can be specialized into different domains, independently of any equipment or implementation.

The Policy Core Information Model (PCIM) is an extension to the DMTF CIM model [ref]. The network is seen as a state-machine, policies being used to control state changes. It must be able to identify/model the current state (objects) and define possible transitions (rules). This model simply defines rules as a set of conditions and a set of actions. An action is being activated when a rule condition is verified. The model includes a few more elements, to be able to define roles, priorities and execution order, but stays in a quite abstract form concerning objects.

Current works around QoS define two levels of extension: the QoS Policy Information Model (QPIM) and the QoS Device Datapath Information Model (QDDIM). The first one integrates QoS specific notions, to be able to create formal representations of abstract policies, such as: "If packet's protocol is http and its destination is in the EXECUTIVES user group, then assign IPP 7 to the packet header". To this aim, the model defines Policy Actions (RSVP – Resource Reservation Protocol, provisioning, PHB – Per

Hop Behavior) and traffic profiles, to specify processing to be done on requests and flows. The QDDIM model extends the previous one to define actions to be done on the equipments (configuration), where QPIM defines actions to be done on packets.

The architecture defines a centralized model for management, policies storage, decision-making, and configuration distribution. This architecture is described in Figure 1.

The Policy Decision Point (PDP) is responsible for making decisions, to its own initiative, or by reacting to a request from a network element. It has to determine which configurations need to be applied to the resources to satisfy the network policies. The main functions deals with the determination of what rules are relevant to the different Policy Enforcement Points (PEPs), the conversion in an adapted format (PIB – Policy Information Base, MIB – Management Information Base, etc.) and the guarantee of their right distribution.

A PEP is a logical entity that enforces policy decisions. It corresponds to a resource offering services that can be used to apply network policies. Main functions of a PEP consist to link the external representations (PIB, MIB, etc.) to the internal equipment configurations, and to maintain local policies integrity while monitoring their enforcement. When integrated in an admission control entity, the PEP is also responsible for validating the requests (ex. RSVP) by soliciting the PDP.

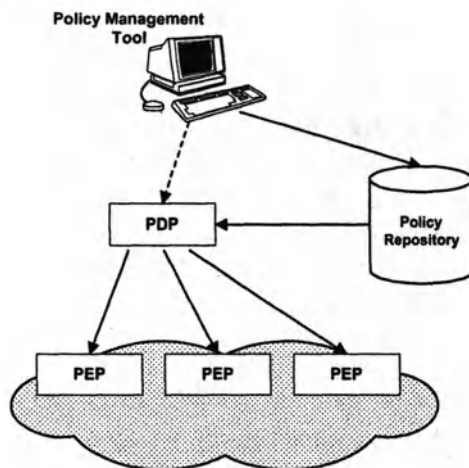


Figure 1. Policy-Based Management Architecture

The architecture model does not require any specific communication protocol or storage method for policies. Nevertheless, the COPS protocol and directories using Lightweight Directory Access Protocol version 3 (LDAPv3) seem to be the preferred ways.

3. COPS AND COPS-SLS

In this section, we present a new protocol we developed to dynamically control SLS negotiation between the network provider and its customers. This protocol named as COPS-SLS is an extension of COPS protocol. It gives I3 platform a means to negotiate (with the network provider) the client's needs, such as QoS parameters, security requirements, mobility needs, etc..

3.1 COPS protocol

COPS (Common Open Policy Service) [1] is a simple request/response protocol based on TCP, proposed by the IETF Resource Allocation Protocol working group to convey policy information between the PEP and the PDP.

The PEP is the policy client and the PDP is the policy server. The local PDP is an optional entity acting as a cache. When the PEP needs policies to manage network resources, it sends a Request (REQ) message to the PDP. Upon receipt of the Decision (DEC) message from the PDP, it enforces network policies.

Two models are defined in policy control: Outsourcing model and Configuration model, a.k.a. Provisioning (PR) model [3]. In Outsourcing model (e.g. COPS-RSVP), the PEP does not know policies before the arrival of a resource request event. When a resource request arrives (e.g. a RSVP message arrives at an RSVP router), the PEP sends a REQ message to the PDP to obtain the decision to accept or reject this resource request. In Configuration model (e.g. COPS-PR for Diffserv), the PEP installs policies before the arrival of resource request event. When a resource request arrives (e.g. a Diffserv packet arrives at a Diffserv router), the PEP enforces installed policies (e.g. remark the DiffServ Code Point, put a packet in best-effort queue...) without sending a REQ message to the PDP.

COPS protocol itself defines 10 messages and objects only for general purpose of policy information transport. In each policy domain, where COPS is used for a specific purpose, an extension must be defined and distinguished by the Client-Type field in the common header of COPS message. There are many extensions of COPS protocol but only COPS-RSVP [4] is assigned by IANA with a 'Client-Type = 1'. Other extensions, such as COPS-PR for DiffServ router configuration [5], COPS usage for IP Traffic Engineering [6], COPS usage for IPsec configuration [7], are currently works in progress.

COPS-SLS Service Level Specification (SLS) [15] is a set of parameters and their values which together define a service offered to a traffic stream. It is the technical part of the Service Level Agreement (SLA). To negotiate a

level of service with an ISP (Internet Service Provider), the client sends its desired SLS to the network. The network can accept or reject the request, or propose another level of service to the client.

COPS-SLS [2] is an extension of COPS protocol. The idea of COPS-SLS is to apply Policy-based networking in SLS negotiation. This extension gives I3 platform a means to negotiate dynamically between a client and a network provider about client's needs, such as QoS parameters, security, mobility, etc. Figure 2 presents the COPS-SLS model.

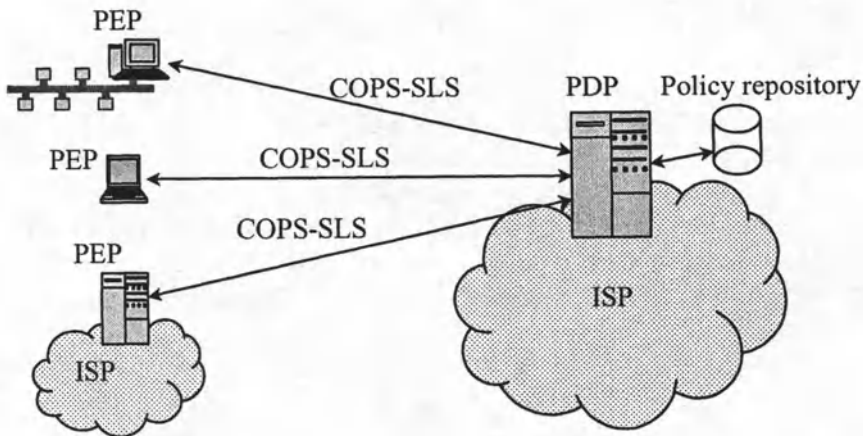


Figure 2. COPS-SLS Model

The SLS-PDP entity represents the network provider and the SLS-PEP entity represents the client. The client here is just a logical entity who requests, possibly on behalf of other entities, network resources. It may be an end-host (connected to the network via a modem), or a gateway of a local network, or another ISP. The fact that the PEP may be put in an end-host broadens the working domain of Policy-based Networking while earlier systems consider the PEP only as network element. We have not only policies to manage network devices in a domain, but also policies to manage each type of client in the domain.

COPS-SLS comprises two phases: Configuration phase and Negotiation phase. The organization of SLS negotiation process in two phases makes the negotiation very dynamic. The Configuration phase decides how the Negotiation phase takes place. For example, the network supplies the client with information about the negotiation mode, time interval to renegotiate, PIB classes to be exchanged, etc. [2]. After successfully installing the configuration supplied by the PDP, the PEP can start the Negotiation phase

by sending a request for its desired level of service. The PDP can accept or reject the request or propose another level of service to the client. The PEP installs policies in the decision and sends a report. If both decision and report are positives, the contract is established and the user data obtain the negotiated level of service. At any time, the network can send an unsolicited decision to change the parameters of the Negotiation phase. This phase will be guided with the updated configuration.

SLS information exchanged between the PEP and the PDP is represented by a named data structure, a.k.a. a PIB [3]. The use of a PIB for a SLS representation gives a possibility to take into account all desired negotiation parameters of all providers. Common classes may be used to specify common parameters of the service and private classes may be used by network providers to personalize their SLS parameters.

For intra-domain application, COPS-SLS can be used to manage user levels of service in the domain. For inter-domain application, it is interesting to establish and manage level of service of inter-domain communications. We hope that COPS-SLS can make the SLA really dynamic both between the client and the network and between ISPs.

4. MOBILITY AND COPS-MU

One aspect the platform I3 has to deal with is user mobility management. In this section, we present an extension of COPS protocol to deal with such a issue.

User mobility [8] concerns the terminal mobility and the personal mobility, which is the ability of the user to use different terminals and access his personal services in his home network from anywhere [9].

We identify four issues in the user mobility management which are: terminal registration, user registration, services portability and QoS negotiation. We propose to have a policy enforcement point (PEP) on the terminal called Terminal PEP (TPEP) which interacts directly with the Foreign PDP (FPDP) for terminal registration, user registration, services portability and QoS negotiation.

4.1 Terminal and user registration

Terminal registration must be achieved only if a terminal is located in a foreign network, if it is a fixed terminal or a mobile terminal located in its home network then terminal registration is not needed. A user registration must be achieved every time a user logs in a terminal even if a user is in his home network

Terminal registration consists of maintaining an association in the Terminal Home Agent (THA) between the terminal Care of Address (CoA) and its home address. User registration consists of maintaining an association in the User Home Agent (UHA) between the user identifier and the IP address of a terminal he is using. The user mobility binding allows to the user to be reachable on the terminal he is using, and to use subscribed personal home network services from anywhere. The later deals with the service portability management.

Steps explained below are the suggested steps for policy based terminal and user registration management illustrated in Figure 3:

1. TPEP interacts directly with the Terminal Foreign PDP or TFPDP (rep. User Foreign PDP or UFPDP) for terminal (albeit user) registration request policy decisions;
2. Mobile Terminal (MT) or the TPEP sends a registration request to the THA (albeit UHA);
3. Terminal Home PEP or THPEP (albeit User Home PEP or UHPEP) interacts with the Terminal Home PDP or THPDP (albeit User Home PDP or UHPDP) for terminal registration request (albeit User registration request) ;
4. HA sends registration reply message to the MT;
5. TPEP interacts with TFPDP (albeit UFPDP) for terminal (albeit user) registration reply policy decisions.

In this scenario the terminal has a co-located CoA such as in IPv6 wich is a routable address. If the terminal has a CoA, it achieves it registration through the FA [10].

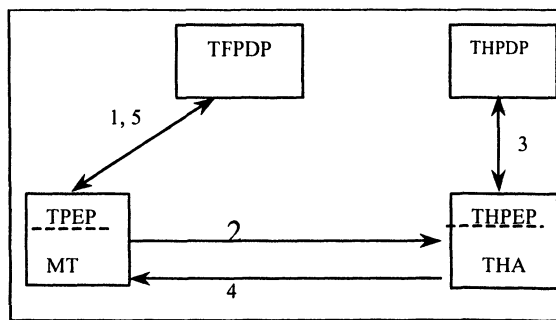


Figure 3. Policy-based Terminal registration

4.2 Service portability

Service portability is related to using personal home network services from anywhere and from any terminal. Third generation of mobile networks deals with service portability using Virtual Home Environment (VHE).

We suggest using a policy concept to deal with service portability in IP networks. Thus, TPEP has to interact with FPDP, UHPDP, and THPDP for service portability and QoS negotiation. This negotiation may be achieved either at a same time with the terminal and user registration or, at any time when a user needs to use a home or a foreign network service.

In a wireless network access we have to minimize traffic on a wireless link. We suggest reporting negotiation process in a fixed part of a network. TPEP interacts only with one PDP; either a HPDP or a FPDP, and make PDPs negotiating to support the user requests for service portability.

Figure 4 illustrates personal home network service portability negotiation. Numbered steps on this figure are explained bellow:

1. TPEP interacts with a UHPDP for service portability. TPEP sends to a UHPDP:

- the terminal profile;
- the required home service.

2. UHPDP interacts with its UHA to determine the required user service features using user profile services. UHPDP based on previous criteria interacts with FPDP and THPDP to decide where to carry out a required home service. FPDP sends to UHPDP its:

- network access features which are related to the QoS that may be offered by a fixed or wireless network;
- foreign network access resources and services cost.

THPDP sends to UHPDP the terminal profile if it is not sent by the TPEP.

3. Based on UHPDP policy decisions, UHPDP may interact with UHPEP to configure its local resources using UHPEP to provide user home services in the home network, FPDP may interact with FPEP to configure its local resources to support personal home service portability, and THPDP may interact with TPEP to configure the MT to support personal service portability.

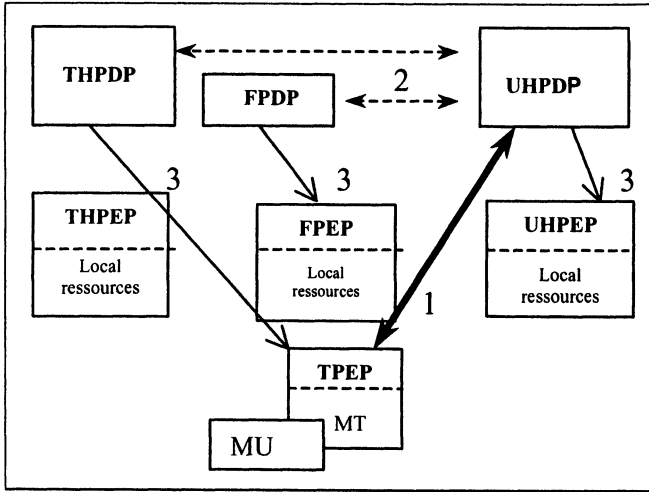


Figure 4. Policy-Based service portability management

QoS parameters defined in fixed networks are: bandwidth, delay, jitter, and, packet loss. User mobility also defines other parameters related to mobility management such as packet loss due to handoff, and, handoff blockage probability. When a MU needs to use one service with a required QoS, the TPEP interacts with the FPD for QoS negotiation. We are working on the extension of the wired access networks QoS negotiation protocol COPS-SLS presented in the previous section to support the negotiation of QoS in wireless access networks.

4.3 COPS-MU policy objects

In the policy based user mobility management we need to define these objects:

- Objects defined in Mobile IP related to terminal registration;
- New Objects defined in extended Mobile IP to support user registration;
- New Objects defined to transport terminal parameters, access network parameters, personal services parameters, user mobility behaviour for service portability and QoS negotiation.

It is important to note that COPS support security mechanisms such as authentication and data integrity. These features are interesting in the user mobility management. Paper [11] suggests that the authentication procedure for Mobile IP can be achieved by another entity than HA or FA. In our model, we suggest that the authentication procedure will be supported by the policy server PDP. Thus, we have to define authentication objects to achieve Terminal PEP, Foreign PEP, and Foreign PDP authentication, and Terminal

PEP, Foreign PEP and Terminal Home PDP authentication, and Terminal PEP, Foreign PEP, and User Home PDP authentication.

Presently, we are defining a new PIB (Policy Information Base) [12] to support mobile user policy objects structure.

5. I3 MODEL

The aim of I3 project is to put within the core network some intelligence to get easier its management. This intelligence will be introduced through agents in network equipments.

In I3 project, a policy-based architecture is considered. In addition, each flow associated to a client is characterised by a profile. The definition of a profile is quite similar to SLA notion, but goes beyond the classical vision characterising a traffic flow. Indeed, a profile, characterising the client's needs, contains a set of requirements: QoS parameters, security needs, mobility requirements, etc.

5.1 A Functional Architecture supporting QoS

In order to meet the main requirements (QoS) of I3 platform, we propose the functional architecture shown in Figure 7. There are four main parts in this architecture: Contracts Management, Network Management, Policy Management and Monitoring.

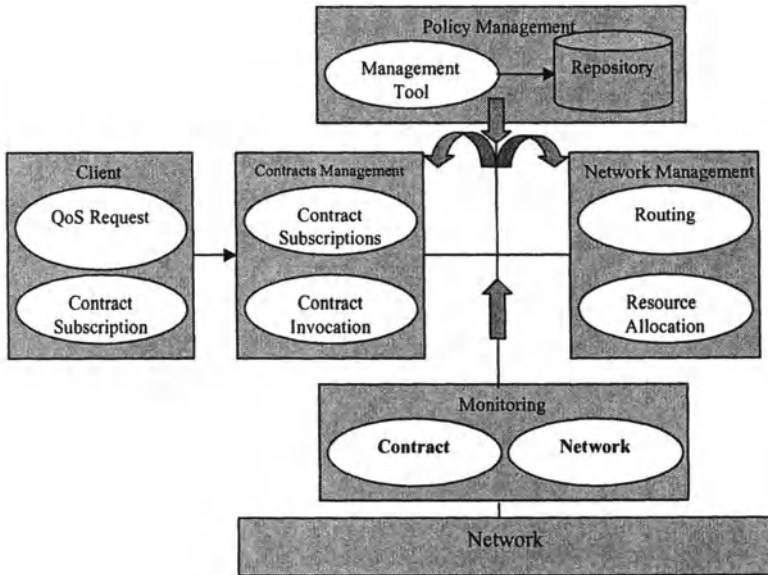


Figure 5. The functional architecture

Contracts Management is responsible for all contracts-related activities and includes two functions: Contract Subscription and Contract Invocation. Contract Subscription is responsible for customer registration and policy-based admission. The customer might either be an autonomous system, a PDP of another domain or a residential customer. Contract Invocation is in charge of admission control as requested by the user.

Network Management is responsible for mapping the traffic onto the physical network resources and configures the network according to the contract. Two sub-functions may be determined: routing and resource reservation.

Policy Management includes functions such as Policy Management Tool and policy Storing Service.

Finally, *Monitoring* provides information about the resource usage and the quality of offered services. Monitoring has not only a diagnostic role but also it has a pro-active/reactive operational role [13, 14].

5.2 I3 platform

The goal of I3 is to put the intelligence within the network by means of agents in order to ensure the functionalities described above.

The organization of activities of the defined agents is necessary. Madkit platform is the candidate platform we have chosen to implement I3 environment. Indeed, Madkit is based on AALAADIN model which provides simple and interesting abstractions allowing agent's activities and agent's interaction modeling.

5.2.1. AALAADIN MODEL AND MADKIT PLATFORM

AALAADIN is the organizational description model that structures MADKit platform [16]. This model is based on three main abstractions (Figure 5)

1. Agent

There are no constraints on the internal architecture of an agent. An agent is just specified as an active communicating entity, playing roles within groups. This definition is left general to allow agent designers to adopt the most accurate definition of "agenthood" according to their application. The agent designer is responsible for defining the most appropriate agent model he needs.

2. Group

Groups are defined as the smallest agent aggregation. An agent is part of one or several groups. The group is only a way to tag a set of agents. In a more developed form, in conjunction with the role definition, the group may represent any usual multi-agent system. An agent can be a member of several groups at the same time. Moreover, AALAADIN groups can freely overlap. Any agent can found a group, and an agent must request its admission to an existing group. Groups might be distributed among several machines.

3. Role

The role is an abstract representation of an agent function, service or identification within a group. Each agent can own several roles, and each role handled by an agent is local to a group. As with group admission, handling a role in a group must be requested by the candidate agent, and is not necessarily accepted.

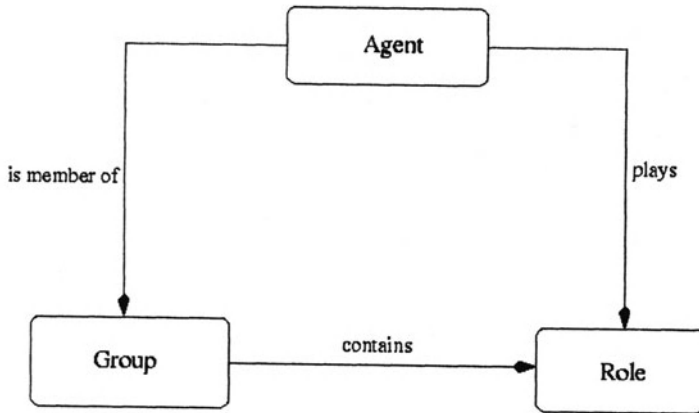


Figure 6. AALAADIN Model basic Abstraction

To communicate, two agents should own a role within the same group. But it is possible to communicate through agents, as shown in Figure 6. In this example, agents A and D cannot directly communicate. Agents A and B own the same group 1 and then are able to communicate directly. Agents B and C own the group 2, and agents C and D the group 3. Then, it is possible to make communicating agents A and D through agents B and C, according to the established hierarchy.

It is worth stressing that an agent can be a member of at least one group, a group contains at least one role, and an agent plays at least one role within one group.

MADKit stands for "Multi Agent Development Kit". It is an agent platform developed at LIRMM laboratory. Written in Java, this platform is easy to use on several system environments.

MADKit is based on AALAADIN organizational model to manage its agents. MADKit platform has been chosen to implement the prototype of I3 project for two main reasons. MADKit agents allow adding some intelligence within network elements and the organizational model makes easier the handling and management of network streams. An agent is a small program written in Java. MADKit platform is used through an API (Application Programming Interface) providing classes and methods allowing an organizational management of the agents. Abstractions as "group" and "role" help during the creation of the agents, giving them a logical role. Classes of MADKit API make easier the communication between agents. For example, it is possible to broadcast a message to all agents of a given group, or to all agents owning the same role within a given group. Other Java methods are defined to ensure the management of roles (add, remove, etc.).

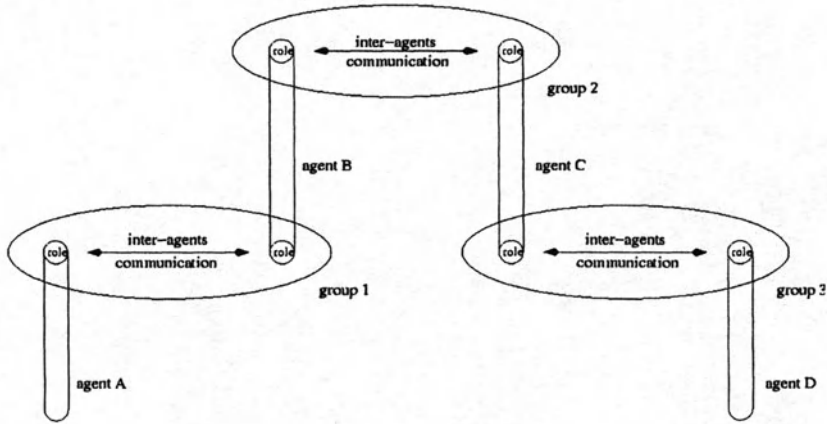


Figure 7. Inter-Agent Communication

5.2.2. I3 PLATFORM MODEL

a. The PEP Model

Figure 8 represents the AALADIN model of the I3 platform. Each "I3 router" is considered as a PEP. The PDP handles the management of network resources and the enforcement of decisions within the PEP so that the client (flow) requirements are satisfied. The I3 router model includes the following components:

The StreamAgent

The StreamAgent, associated with each flow stream, is defined to handle the flow during the communication live time. The main goal of StreamAgent is to provide the platform with information about flow requirements (QoS, security, mobility). For each requirement a group is defined. Thus, if a flow needs a given requirement the corresponding StreamAgent owns a given role within the corresponding group. For example, the StreamAgent could have a "maximum_security" role, in a security group, and a "low_mobility" role, in a mobility group. These roles characterize the stream, making easier its management. The set of flows having the same requirements are aggregated and managed by the same agent i.e., the ManagerAgent.

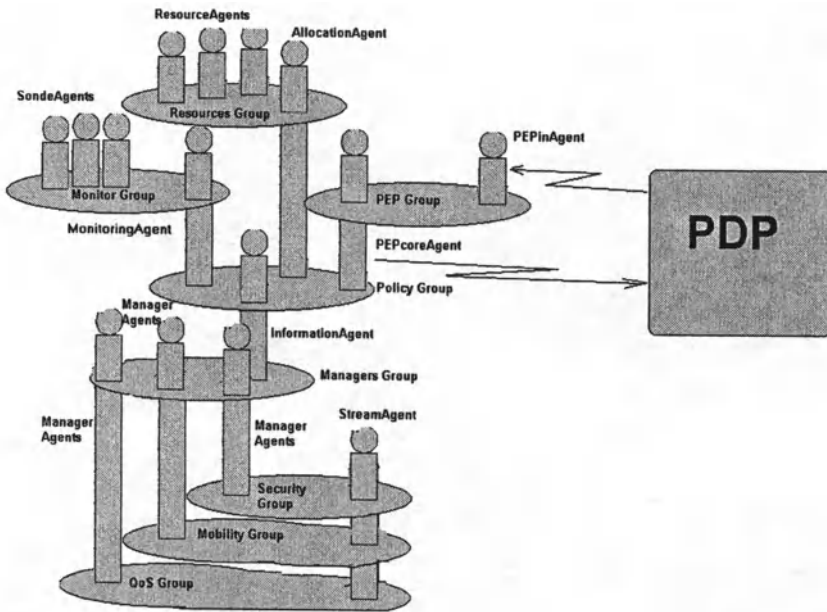


Figure 8. I3 Platform model

The InformationAgent

The InformationAgent has a global vision about all client's needs. It submits the service request to the PDP server via a "Policy" and "PEP" groups. The PDP Server decides on the policies to be enforced to satisfy the needs. In particular, the decisions may concern resource allocation. In this case, the request is taken in charge by the AllocationAgent and forwarded to the ResourceAgents, which represent the network resources.

The MonitoringAgent

The MonitoringAgent is the manager of the "monitor" group, which contains Agents for collecting indicators of QoS.

b. The PDP model

It is worth stressing that the PDP server is a remote entity. Thus, the communication between a PEP and a PDP does not take place in the same group. Intermediate agents, i.e., PEPcoreAgent and PEPInAgent, which communicate via COPS messages, are defined (see Figure 9).

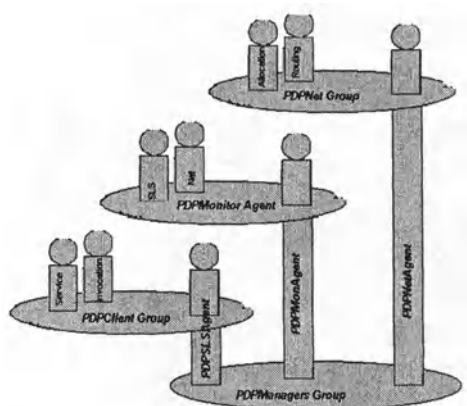


Figure 9. PDP model

c. The client model

In addition to the model defined for the PDP/PEP environment, we propose a client model (see Figure 10) for the choice of the provider. Indeed, the main problem encountered by the operators and providers is at the level of their access network that generally constitutes the bottleneck. We retained the protocol called "Contract Net" based on the concept of contracts between Providers (ISP) and consumers (Customer) of services. We have introduced a broker which is connected to all the providers and who plays the role of an edge router. Thus each customer knows only the address of the broker.

In this model, provider agents take suitable roles in the Provider-Group according to their competence, and attribute prices for their Services. An Agent Broker joins the Client-Group group and the Provider-Group. When a customer wants to make a request for service with a given QoS, a Client agent joins the Client-Group group with the role of member. It seeks for the Broker role, and asks him to find the best offers for the requested QoS (for example, a Gold, Silver or Bronze Assured Forwarding service in DiffServ). The broker agent interacts with the provider concerned with the Provider-Group, chooses the best offers and transmits information to the Client and to the selected Provider. Then, these two agents create their own Contract-Group group to sign the contract.

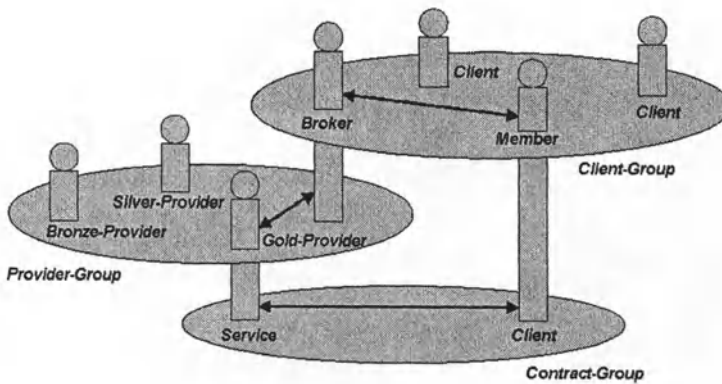


Figure 10. Agent model

6. CONCLUSIONS

This paper introduced new concepts to support future Internet generations. These concepts are based on the introduction of some intelligence in a policy-based networking environment.

We believe that the use of COPS and PDP/ PEP model offers a good way to achieve IP network management. In the case of user mobility, it could allow the deployment of universal personal computing policy control in IP networks by offering policy based network providers negotiation for user and terminal management, service portability, and QoS negotiation.

We proposed a policy-based networking environment supporting new protocols compatible with COPS and simplifying the control of the QoS and the mobility in the IP network. Then, we describe the I3 platform able to support some intelligence to help the new policy-based networking environment to react, manage and control future Internet networks.

Future works concern authentication and more generally the security to be provided in the future IP networks.

REFERENCES

- [1] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.
- [2] T.M.T. Nguyen, G. Pujolle, N. Boukhatem, "COPS Usage for SLS negotiation (COPS-SLS)", draft-nguyen-rap-cops-sls-00.txt, work in progress, June 2001.

- [3] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001.
- [4] S. Herzog, Ed., J. Boyle, R. Cohen, D. Durham, R. Rajan, A. Sastry, "COPS usage for RSVP", RFC 2749, January 2000.
- [5] M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, C. Bell, A. Smith, Francis Reichmeyer, "Differentiated Services Quality of Service Policy Information Base", draft-ietf-diffserv-pib-03.txt, work in progress, March 2001.
- [6] C. Jacquenet, "A COPS client-type for IP traffic engineering", draft-jacquenet-ip-te-cops-02.txt, work in progress, June 2001.
- [7] Man Li, David Arneson, Avri Doria, Jamie Jason, Cliff Wang, "IPSec Policy Information Base", draft-ietf-ipsip-ipsecpib-03.txt, work in progress, July 2001.
- [8] E. Koukoutsis, C. Kossidas, N. Polydorou, "User Aspects for Mobility", Acts Guideline SII-G8/0798.
- [9] A. Fasbender, F. Reichert, E. Gueulen, J. Hjelm, T. Wierelemann, "Any Network, Any Terminal, Anywhere", IEEE Personal Communications 1999.
- [10] C. Perkins, "IP mobility support", RFC 2002, October 1996.
- [11] L. Bos, S. Leroy, "Toward an All-IP-Based UMTS System Architecture", IEEE Network, January 2001.
- [12] B. Moore, et al, "Policy Core Information Model", RFC 3060, February 2001.
- [13] Tequila, "A Management and Control Architecture for Providing IP Differentiated Services in MPLS-Based Networks", IEEE Communications, May 2001.
- [14] Tequila, "A Monitoring and Measurement Architecture for Traffic Engineered IP Networks", IEEE Communications, May 2001.
- [15] D. Goderis, Y. T'joens, C. Jacquenet, G. Memenios, G. Pavlou, R. Egan, D. Griffin, P. Georgatsos, L. Georgiadis, P. V. Heuven, "Service Level Specification Semantics, Parameters and negotiation requirements", draft-tequila-sls-01.txt, work in progress, June 2001
- [16] O. Gutknecht, J. Ferber, "Madkit : Organizing heterogeneity with groups in a platform for multiple multi-agents systems", Technical Report 97188 LIRMM, Montpellier, 1997 <http://www.lirmm.fr/~gutkneco/aalaadin/rr97188.ps.gz>

Accessing CORBA Services from Bluetooth Mobile Terminals

Sami Merovuo, Ari Valtaoja, and Andrei Kotchanov

Lappeenranta University of Technology, P.O .Box 20, 53851 Lappeenranta, Finland

{sami.merovuo, ari.valtaoja, andrei.kotchanov}@lut.fi

Abstract The recent development of mobile communications and mobile devices has created a demand for running distributed applications on mobile devices. CORBA is currently the most widely used middleware solution in fixed networks, providing an open, widely applicable and platform-independent technology that hides most of the complexity in distributed systems. However, CORBA is designed for fixed networks and is not suited for networks containing wireless links. The standardization body of CORBA has recently released a new standard for wireless access and terminal mobility in CORBA which provides a framework for the use of CORBA on wireless networks, and the mechanisms to hide mobility. However, it does not deal with what kind of wireless technologies and networks the suggested framework can be used for and what kind of requirements it sets for them. In this paper, we discuss these requirements, and propose ways in which the framework can be bridged together with a real wireless network technology, Bluetooth. This paper also presents the problems of Bluetooth from the viewpoint of wireless CORBA and suggests some possible solutions for them. This paper proposes, as a result, the architecture for the use of wireless CORBA over Bluetooth.

Keywords: Bluetooth, CORBA, Wireless networks

1. INTRODUCTION

The recent development of mobile communications and mobile devices has created a demand for running distributed applications on mobile devices. In the future, users will even expect to be able to use the same services in both, stationary and mobile devices. Common Object Request Broker

Architecture (CORBA) is currently the most widely used middleware solution in fixed networks, providing an open, widely applicable and platform-independent technology that hides most of the complexity in distributed systems.

However, CORBA, as it exists today, is not perfectly suited for wireless networks because certain assumptions, which are not valid for networks containing wireless links, have been built into CORBA. The standardization body of CORBA, the Object Management Group (OMG), has recently released a standard, the *Wireless Access and Terminal Mobility in CORBA specification* [1], which provides a framework for the use of CORBA on wireless networks, and the mechanisms to hide mobility.

A new wireless technology, Bluetooth, has drawn attention in recent years and it is expected to be the most widely used short-range communication technology between mobile devices in the future. First Bluetooth cards have already entered the market, and the real breakthrough is expected to happen in the near future. The high probability that Bluetooth will be widely used technology, especially on access networks, means that there is a need to provide CORBA services through Bluetooth connections. More than that, making wireless CORBA and Bluetooth work together will also speed up the breakthrough of both technologies.

2. WIRELESS CORBA REQUIREMENTS FOR NETWORK TECHNOLOGY

The main component of CORBA is the Object Request Broker (ORB) which enables communication between client and server objects. Communication in CORBA is transparent, allowing a client to make an object request from a server without having to know where the server is in a distributed network. In a wireless world mobile terminals can access CORBA services from different locations, and perform connection establishment and disconnection at any time. This makes transparent communication more difficult to implement. The wireless CORBA specification provides a framework which enables this transparency by hiding the mobility of the mobile ORBs. The framework does not require any modifications to be made to non-mobile ORBs which means that ordinary ORBs on fixed network do not have to implement the wireless CORBA specification. However, the specification does not discuss on what kind of wireless technologies and networks the suggested framework can be used and what kind of requirements it sets. In order to bring wireless CORBA to some particular wireless network technology, these requirements

have to be defined, and fulfilled by the used technology. These requirements are listed below:

- the existence of mechanisms to detect transport end-points on the link, the network and the transport layers [1].
- the ability of a data terminal to connect to, and disconnect from the transport end-point in a reasonable time.
- the ability to detect connection loss in a reasonable time.

In addition, wireless CORBA contains some optional features that demand more from the underlying technology. The most important of these features is the handoff process, which allows a mobile device to move from one access point to another without losing the CORBA connection. In order to implement the handoff process, there should exist mechanisms in the network technology that inform the data terminal and the access point when the handoff process should be started. The indication can be obtained, for example, from the user, in the case of manual handoff, or from the network, in the case of automatic handoff. To fully implement handoff, the network technology should also allow the establishment of two simultaneous connections between the data terminal and the access point (see chapter 4.2).

3. WIRELESS CORBA CONNECTION

The General Inter-ORB Protocol (GIOP) is an element in CORBA which specifies standard transfer syntax and a set of message formats for communications between ORBs. The GIOP is just an abstract protocol and cannot, therefore, be used to communicate directly between ORBs, although a concrete implementation of GIOP in specialized mapping for some real transport protocol is required. In ordinary CORBA, there exists an end-to-end TCP/IP connection between the client and the server, and the GIOP messages are transmitted using the Internet Inter-ORB protocol (IIOP). In the Wireless CORBA approach, the GIOP connection between the client and server consists of two different connections: one between the mobile host and the access point (Access Bridge), and another between the access point and the other end of the CORBA connection that can be a host in a fixed network or a mobile host (see Figure 1). In order to maintain the interoperability between ordinary ORBs and mobile ORBs, the GIOP connection in the fixed network should be a normal IIOP connection over TCP/IP. Another connection over a one-hop wireless link (e.g. Bluetooth) does not necessarily have to be over TCP/IP which is not a very convenient solution for wireless links without any extensions [2]. Rather, a more optimized wireless link-specific protocol tuned for better performance can be used.

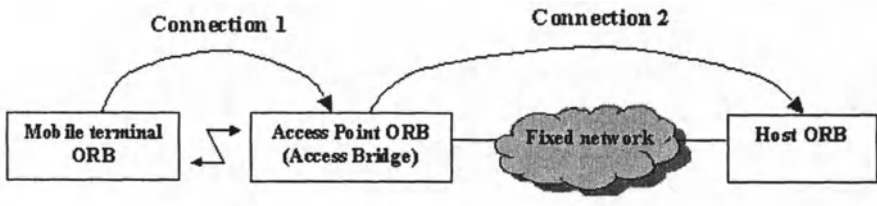


Figure 1. GIOP connections in wireless CORBA

Wireless CORBA does not provide mapping of GIOP messages to some concrete protocol like IIOP in ordinary CORBA. Instead, it specifies a tunneling protocol which encapsulates and decapsulates GIOP messages over the real transport protocol on the wireless media. The GIOP Tunneling Protocol (GTP) is an abstract, transport-independent protocol which defines the message formats for establishing, releasing, and re-establishing the tunnel, but also for transmitting GIOP messages. It also defines the messages for maintaining the GIOP connection through the Access Bridge. GTP is an abstract protocol and, for this reason, needs to be mapped onto some concrete protocol. GTP is designed in such a way that a concrete tunneling protocol is provided as an adaptation layer between GTP and the real transport layer protocol (see Figure 2). Thereby, in order to bring wireless CORBA to some particular wireless network, the adaptation layer between GTP and some protocol of this network has to be implemented.

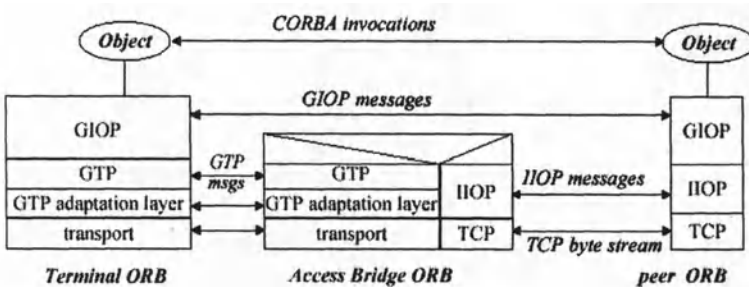


Figure 2. GIOP Tunneling Protocol Architecture [1]

4. BLUETOOTH AND WIRELESS CORBA

A Bluetooth device, which has an interface to some network and which acts as a gateway to that network, can act as an access point for other Bluetooth devices. Hence, it should be possible, at least in theory, to use Bluetooth access points as wireless CORBA access bridges and access CORBA services from Bluetooth data terminals. Bluetooth also has the

capability of forming other kinds of networks, ad hoc networks, that are not necessarily connected to other networks and that can be formed anywhere, anytime, using the proper devices. However, ad hoc networks are beyond the scope of this paper.

As was mentioned in chapter 2, the wireless network should fulfill certain requirements in order to enable wireless CORBA to function on it. This chapter discusses how well Bluetooth meets these requirements and what kind of problems exist when using wireless CORBA on Bluetooth access networks.

4.1 The Problem of Seven Active Slaves

Bluetooth is distinguished from other well-known wireless communication technologies (like Wireless LAN or GPRS) by the very strict limitations it imposes on the number of simultaneous connections it accepts. In a Bluetooth, a strict master-slave scheme is employed where slave devices can communicate with each other only through the master device. The master and slaves form a *piconet*, where the master can maintain up to seven *active slaves* connected. An active slave is a Bluetooth device which is able to communicate and transfer data with the master. More slaves may exist in a piconet but only if they are in a low-power mode where they are not able to transfer data with the master.

The most convenient architecture for Bluetooth access networks is one where the Bluetooth access point acts as a master and the mobile terminals accessing it act as slaves (see Figure 3). The seven active slaves restriction causes major problems in this kind of architecture whenever a new data terminal desires to connect to an access point which is already fully reserved, having seven slaves connected already. In this case, the Access Point refuses to serve the most recent Bluetooth device until one of the already connected data terminals releases its connection. The worst scenario is one in which the access point is full and slaves, which may not necessarily perform any data transfer, have already reserved the active places in the piconet.

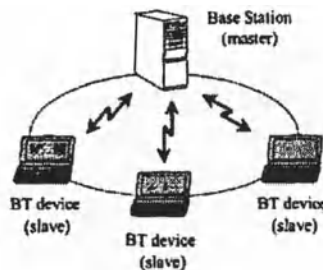


Figure 3. Bluetooth architecture [3]

It is obvious that the problem of the seven active slaves decreases the usability of certain access points. However, in practice, there always exists some kind of limit to how many clients an access point can serve. The access point shares its bandwidth among the connected slaves and thus, the number of connected slaves cannot be much greater than seven if the proper connection is to be guaranteed to all slaves. The problem is, therefore, not the small number of connected slaves but rather the existence of data terminals that are connected to the access point or that have reserved active places from the piconet without having a reason for doing so. The problem can be solved with the help of *park mode*, which is a power-save mode where a slave in a piconet remains synchronized to the master but is not allowed to transfer data. Thereby, at a particular moment, when there is no data traffic, the idle active slave should be switched to the park mode in order for an active slave place to be released from the piconet for devices that may really need it. Switching to the park mode and back again can be done at any time by a decision from the master. A slave can also leave a request to be switched to one or the other mode. Whenever a parked slave needs to transfer data again or if the master receives some data targeted to a parked slave, the slave can be switched back to the active mode. Of course, in order to switch back to the active mode, there has to be room in the piconet for one more active slave. If there is no room, some other active slave can be switched to the park mode. In a case where the piconet is full and more than seven slaves need to transfer data, some kind of prioritisation can be used for managing the switching of slaves to the park mode and back.

4.2 Wireless CORBA Handoff

Wireless CORBA specifies methods for handoff, moving from an access point to another without losing the CORBA connection. There exist two kinds of handoffs: 1) backward handoff is a method where the connection between the mobile terminal and the access points are not dropped, 2) forward handoff is a method where the connection is recovered after a sudden loss of connection. Generally, handoff can happen because of following reasons:

1. a mobile terminal physically moves and changes the current access point to another without losing the CORBA connection (backward handoff);
2. a mobile terminal is switched from the current access point to another, without losing the connection, by a server that distributes data traffic between access points in order to provide better bandwidth usage (backward handoff);
3. for some special reason, the user of the Bluetooth mobile terminal wants to change the current access point to another without losing the

connection (backward handoff, an example of which is discussed in chapter 4.5);

4. a mobile terminal unexpectedly loses the connection to the current access point and the connection is re-established (forward handoff).

Implementing forward handoff for Bluetooth should be quite easy because it involves simply re-establishing a connection that has been lost. The backward handoff process causes more problems. Wireless CORBA specifies that a connection to the new Access Bridge has to be established before the connection to the old access point can be released. However, if the terminal cannot maintain simultaneous transport connectivity to the old and new Access Bridge, the so-called “alternative handoff procedure” can be used. The alternative handoff procedure first releases the connection to the old Access Bridge and only afterwards establishes a connection to the new Access Bridge. It is obvious that the performance in the latter approach is much worse than in the former. The requirement of “real” backward handoff to have simultaneous connections to both the new and old access points makes its implementation impossible for existing Bluetooth devices. The reason for this is that they do not provide support for *scatternet* which allows Bluetooth slaves to be connected to more than one master at the same time. However, while the support for *scatternet* is awaited, the alternative implementation of backward handoff can be used without problems, but of course at the price of performance.

4.3 Starting the Backward Handoff Procedure

Another, much more difficult problem, is to recognize when the backward handoff process should take place. Bluetooth was originally designed to simply replace cables, and there was no intention to make it work as a real wireless network and provide mobility services etc. This is the most probable reason why efficient mechanisms have not been incorporated into Bluetooth in order to determine its location inside the piconet. However, for backward handoff, location information is essential because without this information, it is impossible to know when the device should start the process of changing the access point. The device may be anywhere inside the piconet - on the edge of the piconet or right beside the master device. The only information that can be obtained is that the connection has been lost which is, for instance, the case when the device moves out of the piconet.

In GSM, the decision of changing the base station is made upon strength of the radio signal. The Bluetooth specification does not define any methods for measuring the strength of the radio signal but there are some methods that can be used. To achieve a hint of the location of the terminal, the so-called inquiry process (explained in chapter 4.4) can be run periodically.

However, inquiry is a very heavy process that takes quite a long time and only provides information on what devices exist in the vicinity, not telling, for instance, the mobile terminal to which access point it should connect to or how far the device is from different access points. Another, and much more convenient possibility, would be to use the Bluetooth method “strength of link” (also known as “link quality”). This method does not measure the radio signal strength, but calculates a value on the basis of the number of access errors and retransmissions during data transfer. The greater the distance of the mobile terminal to the master, the greater the number of errors and retransmissions is and thereby, the link quality value can be used to detect when a terminal is moving outside the piconet and there is a need to initiate the handoff process.

The measurement of the strength of a link can be implemented in two different ways depending on whether the mobile terminal or the access point is performing the measurement process. However, measuring the strength of a link only gives a hint of the distance between the master and the slave, not providing accurate coordinates. It is impossible to know in which direction the device is moving and to which access point the device should try to connect. There may very well be another reason, other than the distance, for the increase in the number of errors (for example, the interference could be caused by physical obstacles or other devices). Nevertheless, “strength of link” seems to be the only feasible way of obtaining information on mobility if measuring the strength of the radio signal is not possible.

4.4 Connection Establishment to a New Access Point in Backward Handoff

Bluetooth incorporates special mechanisms for establishing connections with other devices. Inquiry is a process where a Bluetooth device searches for other Bluetooth devices in the area and obtains their Bluetooth addresses. Paging is a process where the connection to the device that was found in inquiry is established. In backward handoff, when there is a need to establish a connection between the new access point and the mobile terminal, the following procedure is followed:

1. the mobile terminal does not know if there exist any other access points in the area and hence, the inquiry process has to be run by the mobile terminal in order to find new access points and obtain their Bluetooth addresses;
2. after an appropriate access point is found and the corresponding address received, the mobile terminal can connect to it by running the paging process.

Alternatively, the access point can run the inquiry and paging processes and establish the connection to the mobile terminal. In any case, the problem in the Bluetooth procedure for connection establishment is the inquiry process, which is very heavy and thus consumes substantial amounts of time. As can be seen from Table 1, the inquiry and paging processes together take approximately between 4.28 and 6.28 seconds on average, but can, in the worst case, take up to tens of seconds. In backward handoff, where the mobile terminal moves between Bluetooth access points that are within a range of 10 meters, a delay of between 4 and 6 seconds for Bluetooth connection establishment is unacceptably long. In addition, the wireless CORBA handoff process will consume some time. At worst, the delay can be over 30 seconds, which is totally unacceptable.

Table 1. The time taken to complete the inquiry/paging procedures [4]

Operation type	Minimum Time	Average Time	Maximum Time
Inquiry	0.00125	3-5	10.24 - 30.72
Paging	0.0025	1.28	2.56
Total (Paging + inquiry)	0.00375	4.28 - 6.28	12.80 - 33.28

However, if the address of the new access point is somehow known in advance, there is no need to run the time-consuming inquiry process. If the address is known, only the paging process is required for establishing the connection, which takes only 1.28 seconds on average and, even at worst, only 2.56 seconds (Table 1). The address of the new access point can be known in advance if, for example, the mobile terminal receives the address information from the current access point. If the access points are responsible for establishing the connection, the current access point delivers the address of the mobile terminal to the next possible access points. These access points then run the paging procedures, and the connection is established between the mobile terminal and the access point in the area of which the mobile terminal happens to be. Once again, only paging is needed. The point of delivering the address information is that the inquiry process only has to be executed once when the connection between the first access point and the mobile terminal is established.

Another disadvantage of the inquiry process is the fact that during this process, devices that already have a Bluetooth connection to the inquiring device are not able to communicate with it [3]. This is the main reason why the inquiry process should not be run by any access points, and is in actual fact a major disadvantage for the whole Bluetooth technology. For example, let assume that there exists a mobile phone with a Bluetooth headset and a laptop that has a Bluetooth chip. Now, if the mobile phone and the headset are used for a telephone call, and during the call the mobile phone runs an

inquiry to find the laptop, there will be a break in communication between the mobile phone and the headset until the inquiry process is terminated.

4.5 Backward Handoff between Different Technologies

Backward handoff between different technologies may be even more important than normal backward handoff between Bluetooth access points. For example, a user may currently be using a GPRS connection but would like to switch to a Bluetooth connection when they arrive in the vicinity of a Bluetooth access point (perhaps because Bluetooth is cost-free). Vice versa, a user using a Bluetooth connection may want to switch to GPRS if they know that they will soon move out of the range of Bluetooth and risk losing the connection.

Implementing backward handoff between different technologies should be quite easy, at least if the user makes the decision to switch from one technology to another (manual handoff). In such a case, all problems related to automatic handoff can be forgotten (Chapter 4.3). The user is, most probably, really responsible for making the decision to switch between technologies, at least if switching means alternating between cost-free services and services that carry charges and vice-versa (for example, from Bluetooth to GPRS). The process can also be automated, although the user should always be aware of when the new connection will carry a charge and when it will not.

4.6 Connecting to the Correct Access Points

An area in which a device moves can contain several Bluetooth access points all of which may not necessarily provide access to wireless CORBA services and there must, therefore, be some way of recognizing the correct access points. Bluetooth provides the mechanisms to request information on what kinds of services exist on a specific Bluetooth device, although the problem is that the connection to the device first has to be established. This information cannot, therefore, be obtained through inquiry calls, for instance.

Connection establishment takes time and if the connected device does not provide the required services, even more time is wasted when connecting to another access point. Nevertheless, as was discussed in previous chapters, connection establishing can be done in order for the inquiry process to be carried out only once, when the first connection is established. Afterwards, the addresses of new access points are always obtained from the current access point, so that only the paging process needs to be run. This sort of an approach also solves the problem of connecting to the correct access points because it is possible to transfer only the addresses of those access points

that provide some particular services such as, in this case, access to wireless CORBA services.

5. CONCLUSIONS AND FURTHER WORK

The disadvantages of Bluetooth technology impede it from fully meeting the requirements set by wireless CORBA. Bluetooth, in its present form, enables the use of wireless CORBA at some level, while the proper use of CORBA is not possible. The usability of Bluetooth access points is low, connection establishment mechanisms are slow, correct access points cannot be recognized and the support for automatic handoff cannot be implemented. However, as the research carried out by the authors has shown, these problems can be avoided. Switching active idle slaves to the park mode solves the problem of the seven active slaves. Obtaining the address of the new access point from the current access point speeds up the connection establishment process and the time-consuming inquiry process need not be run. This approach also solves the problem of recognizing the correct access points. The strength of the link can be used to obtain the impulse for starting the backward handoff process and automatic handoff can be implemented. Thus, these kinds of extensions to Bluetooth allow the proper use of wireless CORBA over Bluetooth.

In our proximate work, we will study which protocol in the Bluetooth protocol stack can be used for tunneling and will specify the adaptation layer between that protocol and GTP. Furthermore, we carry out a prototype implementation of tunneling and test our work against real wireless CORBA implementation.

REFERENCES

- [1] Object Management Group, "Wireless Access and Terminal Mobility in CORBA specification", June 2001
- [2] E. Amir, H. Balakrishnan, S. Seshan, R. Katz, "Efficient TCP over Networks with Wireless Links", May 1995.
- [3] S. Baatz, M. Frank, R. Göppfarth, D. Kaastkine, P. Martini, M. Schetelig, A. Vilavaara. "Handoff Support for Mobility with IP over Bluetooth", 2000
- [4] Palowireless - Bluetooth Resource Center, "Time Taken to complete Inquiry/Paging Procedures".
- [5] <http://www.palowireless.com/infotooth/knowledge/baseband/99.asp> [Accessed 26 Sep 2001]

Comparison of Softswitch-Architectures Using Parlay as Application Interface

J. Stadler¹, I. Miladinovic¹, R. Pailer¹, and M. Vucic²

¹*Vienna University of Technology, Institute of Communication Networks, Favoritenstrasse 9/388, A-1040 Vienna, Austria; {Johannes.Stadler, Igor.Miladinovic, Rudolf.Pailer}@tuwien.ac.at*

²*SIEMENS AG Österreich, Gudrunstrasse 11, A-1100 Vienna, Austria*

Abstract In the last years the convergence between circuit switched and packet switched communication networks has become more and more important, not at least because of the growing Voice over IP or rather Multimedia over IP tendency. Currently, network providers from the classical circuit switched world have to think about enabling interconnection to the IP domain over appropriated gateways. Another important issue is the centralization of service logic to make it usable from different types of networks. If this is done over open interfaces, so called 3rd Party Service Providers could offer their services in a more flexible way. Hybrid Components, called softswitches, could fulfil both, gateway functionality and access to centralized applications using standardized interfaces. One possibility for such an interface could be the Parlay API. In this paper we propose three different softswitch architectures supporting connectivity to the PSTN as well as to the IP world. They differ in the placement of network specific components and in the way of gateway control. In each case Parlay is used between the networks specific Call Control and the independent application.

Keywords: Softswitch, Parlay, Gateway, Services, SIP, MGCP

1. INTRODUCTION

Due to the growing importance of packet switched technology in contemporary communication networks, there is a promising tendency of convergence between the world of telecommunication and that of data communication. As much as the transport of voice or more general multimedia data over packet switched networks provides an enormous opportunity, it also implies incalculable risks for current circuit switched network providers.

Beside their PSTN services as main business, they have to decide about the development of new packet switched networks, in many cases from the scratch. Huge investments would have to be made for hardware and software, new groups of users have to be addressed and new, more powerful services have to be invented to get back the high costs.

To reduce the financial effort, it would be a goal to utilize equipment, which is applicable for both domains. Furthermore, services should have to be developed only once. Obviously this reduces the cost of the creation of such services and moreover it enables a new class of applications, based on the interaction of different types of communication networks.

So it seems possible that for the years of migration, systems for hybrid network handling could be a meaningful alternative. Using current architectures, which we know from PSTN switches, this is not easily possible. In most cases such devices are designed in a more or less monolithic manner. The software is tightly coupled to the hardware mostly, interfaces are rarely standardized and applications are hardly reusable.

This implies two further problems. Because of the lack of an open service interface only the switch developers are able to develop applications for a certain piece of hardware. This affects the spectrum of available services and increases the time to market. The second issue is that it is impossible to offer network capabilities to so called third party service providers. So only network providers have the ability to act as service providers. Figure 1 shows this conventional architectural approach.

Above considerations motivate the redesign towards a more flexible and more open architecture. Figure 2 depicts such an approach, which is commonly called a softswitch. The International Softswitch Consortium (ISC), formed in 1999, offers the industry a global coordination and promotion of these issues.

A softswitch enables the combination of applications, call control and hardware from different vendors as well as an abstraction of the applied hardware to the call control layer and of the communication network to the applications, by the usage of open interfaces.

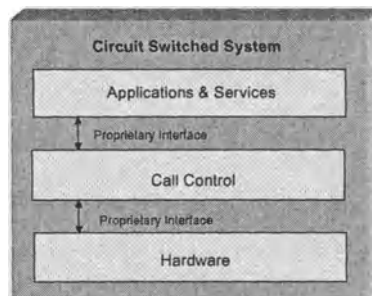


Figure 1. Classical Circuit Switched Architecture

Different realizations of such softswitches are imaginable. Due to a wide range of requirements, it is not possible to fulfil all of them without any shortcomings. In this paper we propose and compare three possible architectures of different softswitch realisations, which have in common that they are all based on the Parlay APIs [1], which are designed as an interface for the open separation between the service logic and the call control layer.

Because Parlay supports developments for the IN, network providers are enabled to reuse their legacy services, which seems to be one of their base requirements. Chapter 2.1 gives a brief introduction into Parlay, its functionality and its advantages for the usage in softswitch architectures.

The scenario, which we have chosen for explanation of the several approaches is a call setup between circuit switched and packet switched domain. Due to this reason, the usage of a media gateway, which is the entity sending, receiving and converting data in an appropriate way, is necessary. For the control of such media gateways, IETF'S Media Gateway Control Protocol (MGCP) [9], which we describe in chapter 2.2, is proposed.

For the purpose of call control in packet switched networks, different protocols are available. Nowadays it is not certainly clear which one will be able to survive or to become industrial standard, but it seems to be the Session Initiation Protocol (SIP) [7], also created by the IETF. 3rd Generation Partnership Project (3GPP) decided in favour of SIP as call control protocol in UMTS, starting with release five. Therefore, our scenarios are based on SIP, which makes them applicable to conversions between PSTN and the mobile domain without any changes. For a better understanding chapter 2.3 overviews the Session Initiation Protocol briefly.

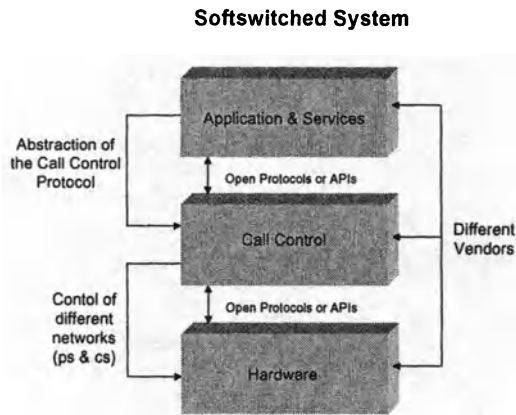


Figure 2. Softswitched Architecture Approach

In chapter 3 we describe three different design principles, which differ in the place where the entities for the generic call control reside, either common or distributed, and whether the application has to control the media gateway or if it works autonomously. Chapter 4 summarizes the conclusions.

2. RELATED PROTOCOLS AND TECHNOLOGIES

2.1 Parlay

Parlay [1] is a forum of many of the key players in telecommunications and computer industry with the goal to define object oriented APIs that allow third party application developers to access network resources and functionality in a generic and technology independent way.

The Parlay APIs have their origins in the Intelligent Network (IN) architecture [2] that is widely used in today's telephony networks. The IN service framework, however, is not an open platform for application development and IN service creation is a time consuming task and restricted to the network provider domain. The Parlay forum learned from the deficiencies of the IN architecture and separated the un-trusted service provider domain from the trusted network provider domain.

Furthermore, Parlay service capabilities are defined far beyond the scope of simple telephony and include the possibility to create powerful applications that combine the strengths of data networks with multimedia communications. A Parlay client application can be developed and run by Third Party Service Providers outside of the Network Providers core domain.

The Parlay APIs between those domains manage access to the network resources and are divided in the Framework Interfaces for security, service registration, service discovery, service subscription and management, and the Service Interfaces for the control of specific network capabilities like multimedia call control, mobility, messaging, QoS management or user interactions. Currently, the Parlay APIs are available in version 2.1 and interface definitions are published for Microsoft's DCOM and CORBA.

For the next phase of the Parlay API specification that is expected for the end of this year further extensions of the existing APIs and new network service APIs are planned like generic charging and billing, enhancement of user profile & subscription data handling, policy management, new framework interfaces, management support, new generic application interfaces and mobile E-pay.

2.1.1 Advantages of using the Parlay APIs in a Softswitch architecture

The Parlay APIs have initially been proposed for providing 3rd party service developers with a secure interface towards telecommunication resources of a network service provider. Towards the application program only a highly abstracted, object oriented and fully distributed call-processing model is visible. Existing solutions usually offer access to one specific switching resource (e.g. an IN Switching Control Point – SCP), where the calls and call legs are abstractions of the same network technology. A call that crosses networks boundaries is therefore only visible up to the gateway where the call leaves the Parlay controlled domain.

We argue that this classic Parlay usage model fits very well in the proposed Softswitch architecture where the call control layer is separated from the application and service logic layer by open and standardized interfaces. The Parlay APIs facilitate the implementation of Softswitch architectures by a mature security and management concept, by an object-oriented design and by the distributed nature of all of the interfaces.

We propose that apart from this classic 'network technology centric' usage model the Parlay APIs can be used to present a call model towards the application logic where a single instance of a call object can have call legs that use different access technologies. The Parlay Multi-Party call model defines the call leg as an abstraction of a signalling relationship between an end-point and a call object. Call legs are modelled as independent interfaces that are implemented by distributed objects (e.g. by using CORBA technology). This means that the Parlay specification allows call legs that belong to one call to be implemented by objects that run on different physical machines. A network resource that offers access through some

specific technology (e.g. circuit switched ISDN, analogue lines, SIP, H.323) can host a Parlay call leg that represents and controls this half of the call. The call legs are associated with a call that runs for example on a separate call-processing server. In this architecture a signalling gateway between different network technologies is no longer necessary as its function is implicitly carried out by a common Parlay call processing model. Thus, call set-up procedures and call control tasks are unified and simplified through a common and distributed call model. We have shown that a complete mapping of Parlay's Generic Call Control Service functionality towards Session Initiation Protocol (SIP) [7] messages is possible, so that Parlay can be used for control of a modern IP based call processing system [3], [4].

2.2 Media Gateway Control Protocol (MGCP)

Telephony gateways are network elements between circuit switched (e.g. Public Switched Telephone Network (PSTN)) and packet switched networks (e.g. Internet) that basically perform two types of tasks: signalling translation and media translation. The European Telecommunications Standards Institute (ETSI) project Telecommunications and Internet Protocol Harmonization over Networks (TIPHON) founded a decomposition model for telephony gateways, where the gateway is divided into three entities – a Signalling Gateway (SG), a Media Gateway (MGW) and a Media Gateway Controller (MGC). Nevertheless, the decomposed gateway appears to the outside as a single gateway. The gateway decomposition does not impose a physical breakdown, but rather a logical perspective. The goal of this logical perspective is to allow the signalling that enables services to be normalized into a media control interface, thereby separating services from the underlying media handling. The key advantages of this decomposition approach are the higher degree of scalability and maintainability the resulting gateway solutions provide and the opportunity to introduce new services more quickly. [5]

Each of the three entities of a decomposed gateway has to perform different tasks. The SG mediates between circuit switched (e.g. Signalling System 7 (SS7) or Integrated Services Digital Network (ISDN) [6]) and packet switched (e.g. Session Initiation Protocol (SIP) [7] or H.323 [8]) signalling protocols. The MGW focuses on the audio signal translation function, performing conversion between the audio signals carried on circuit switched networks and data packets carried over packet switched networks. Finally, the MGC acts as a manager within the telephony gateway and instructs the MGW according to the signalling information received by interfacing with the circuit switched network via the SG on the one side and

with the packet switched network via a signalling device such as a SIP server or an H.323 gatekeeper on the other side.

The Media Gateway Control Protocol (MGCP) [9] serves as an internal master/slave protocol between the MGC (master) and the MGW (slave) of an ETSI/TIPHON compliant decomposed telephony gateway. MGCP retains simplicity by utilizing an Application Programming Interface (API) that is made up of a fairly small set of eight commands, which are grouped into three basic command sets: device management, connection control and in-band signalling handling. Commands may take numerous arguments and are transmitted using plain American Standard Code for Information Interchange (ASCII) text. Like SIP and H.323, MGCP also relies on already established standards such as the User Datagram Protocol (UDP) [10] for transmitting messages between the MGC and the MGW (with the expectation that transmission reliability will be managed by the specific implementation), the Session Description Protocol (SDP) [11] for negotiating the media aspects of a call (e.g. ports and Internet Protocol (IP) [12] addresses, voice coding methods and other connection type parameters), the Real-time Transport Protocol (RTP) [13] used by the MGW for handling the transport of media streams, and the Internet Protocol Security (IPSEC) [14] protocol for providing secure connections.

The benefits of MGCP are manifold: the potential to control very large deployed systems, the opportunity of simple integration into SS7 networks, which gives greater control and throughput in handling calls and finally the fact that gateways utilizing MGCP can co-exist in networks with other SIP or H.323 compliant entities. One of the weaknesses of MGCP is the time-consuming implementation of the protocol for smaller applications.

MGCP owes its origin to the confluence of the Simple Gateway Control Protocol (SGCP) [16] and the Internet Protocol Device Control (IPDC) [17] protocol. Nevertheless, MGCP is not the final word on gateway control. The IETF combined forces with the ITU to endorse the MEGACO/H.248 [18] standard in late 2000. MEGACO/H.248 draws heavily from MGCP plus introduces several enhancements, as for instance: support for multimedia and multi-point conferencing enhanced services, improved syntax for more efficient semantic message processing, TCP [15] and UDP message transport options, text or binary encoding and an expanded definition of packages. Accordingly, MEGACO/H.248 can be considered as a richer approach to gateway control, nevertheless implementation is a lot more complex.

2.3 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) [7] is an application layer protocol, used for signalling in IP networks. It was developed by the Multiparty Multimedia Session Control (MMUSIC) working group of the IETF. SIP allows establishment, modification and termination of all types of sessions. It is a text-based, HTTP-like peer-to-peer protocol with two basic types of messages, requests and responses. SIP messages also carry information about the session. For VoIP applications and multimedia conferencing applications this information is usually placed in the Session Description Protocol (SDP) [11]. In that case the SDP packet is a part of the SIP message, called body. Another important IETF protocol for VoIP is the Real-time Transport Protocol (RTP) [13]. It is used for the transport of real-time media data (voice and video) in an established session.

There are two types of entities in SIP, SIP User Agents (UA) and SIP network servers. A SIP UA could be seen as end device and acts either as user terminal or as automated connection endpoint, for instance a call answering machine. Network servers are used for call routing and for maintaining call states, as well as they could be enabled to perform different kinds of applications. They are divided into three different types, the proxy server, the redirect server and the register server. A SIP proxy server is an application layer router that forwards SIP messages after address resolution. SIP redirect servers reply incoming SIP request with a new user location. After receiving a response from a SIP redirect server, a SIP UA will try to reach the called party at the new location.

Call setup in SIP is quite simple. Suppose there are two users (A and B) and user A wants to set up a call to user B. User A sends an INVITE request to B. If B wants to accept the call, he replies with OK response. The last step is the sending of an acknowledge (ACK) request by A, after the response arrives from B. As we can see, SIP uses a three-way handshake (INVITE, OK, ACK) for the call setup. If one party wants to tear down the call, it sends a BYE request to the opponent UA, which replies with an OK response and the call is terminated.

3. DIFFERENT SOFTSWITCH ARCHITECTURES USING PARLAY

In this chapter we propose and compare three possible softswitch architectures, which differ in some specific parts. The scenario we used to explain the functionality is the same for every approach. A user in the PSTN wants to call another one, who has configured a call routing application in

the way that the call is forwarded into the IP domain, more exactly to a certain SIP address.

The application resides in the service provider's domain and offers two access points. The first enables the user to configure his service via a HTTP interface, for instance to decide to which SIP UA a call to his PSTN number should be forwarded. The other, offering the Parlay API, is used by the call control components to access the application, e.g. for the purpose of address translation or for the interaction towards the media gateway controller.

Call control components are separated in two parts, the generic and the specific call control part. The generic call control part represents the call towards the application, but is independent of the underlying signalling network. It interacts with the specific call control, which represents a certain call leg towards the protocol stack of the corresponding network via an open interface.

The media gateway controller gets its tasks either by the application itself using the Parlay interface, or via the generic call control part. In that case it works autonomous, as described in chapter 3.3, performing the signalling towards the different networks by its own. It controls the media gateway, which actually does the media conversion between the involved networks and deals with the appropriated transport protocols or mechanisms.

As mentioned above, the call should be forwarded from the PSTN domain into the SIP domain. Due to this, ISUP and SIP protocol stacks take place beneath the specific call control parts.

We compare the architectures in the degree of complexity in the application versus that in the generic call control. We suggest that application programmers should not necessarily have to understand call or network specific parts. The required traffic over the open interface towards the service implementations is another relevant point, because such interaction is probably remote and so it deteriorates call setup time intensely. Moreover, it is reasonable that the application exists only once, whereas different call control facilities have to interact with it. So we can say it is important to have very low traffic here. Finally, it might be important that the single blocks could be independent, which means that the ideal case is being able to combine components of different vendors.

3.1 Common Generic Call Control Unit

The most important detail of this architecture is the necessity for only one Generic Call Control (GCC) unit. It communicates on the one hand with different protocol specific Call Control (CC) units and on the other hand with the application as shown in figure 3. The interface between the GCC unit and the application is Parlay. Note that the application makes usage only

For calls between different kinds of networks (e.g. circuit switched and packet networks) the usage of a Media Gateway (MGW) is necessary. It converts the media streams between two networks. Therefore, the GCC unit contacts a Media Gateway Controller (MGC), which tries to occupy a trunk between the MGW and the PSTN switch (5-8). If this does not succeed, a busy signal is sent to the PSTN user and the call setup ends. Otherwise the GCC unit contacts the SIP specific CC unit with the previously resolved destination address (9). The SIP specific CC unit creates the second call leg of this call and forwards the call request to the SIP stack.

The SIP stack sends a SIP INVITE request to the destination address (in this case it is the address of a SIP UA) (10). The SIP UA indicates an incoming call. Supposed the user wants to accept this call, the SIP UA sends an OK response to the SIP stack of the softswitch (11). This response is forwarded to the SIP specific CC, which notifies the GCC unit (12). The next step is to inform the MGW about the media settings of the callee via the MGC (13-16).

Afterwards, the GCC unit informs the ISUP specific CC unit that the call-setup was successful (17). This information is forwarded to the ISUP stack and further to the PSTN switch (18). Finally, the ISUP specific CC notifies the GCC unit (19) that appraises the application (20) and the SIP specific CC (21) that the call-setup succeeded. The SIP CC forwards this information to the SIP stack and a SIP ACK request is sent to the SIP UA (22). After completion of the call-setup, media data exchange (voice, video) starts (23).

The big benefit of this architecture is that the application communicates only with one entity (GCC unit). Therefore, the programmer of the application does not need to care about underlying network issues. He simply always delegates this duty to the common GCC unit. This abstraction simplifies service creation, which reduces time to market. Another important benefit is the low traffic over Parlay. We suppose that function calls over this API would be remote in most cases. Economizing this traffic enables better call setup times. As a drawback of this architecture we notice that the GCC unit, protocol specific CC, protocol stack and MGC must be obtained by the same vendor, because of proprietary interfaces, and a more or less distributed state-machine between the CC entities. It means that there is no possibility to combine protocol stacks of different vendors. Besides that, we should remember the complexity of the GCC unit.

3.2 Distributed Call Control

In this architecture (see figure 4) there are no proprietary interfaces, due to the usage only of the open Parlay API between all protocol specific

components and the application. Every protocol specific component contains a GCC unit and is therefore able to communicate with the application.

For the scenario mentioned in the above section, it is important that in this architecture the GCC unit of the ISUP stack (note that is not a protocol specific CC unit) contacts the application directly (2) over Parlay. The application contacts the MGC that orders the MGW to occupy a trunk to the PSTN switch (3-6). After that, the application performs a call-setup (7-10) into the SIP domain, contacts MGC to configure callee's media settings (11-14) and notifies the GCC of ISUP stack that the call-setup has succeed (15). A call representation is created in both GCC (ISUP and SIP) and in the application. Comparing this architecture to the one of chapter 3.1, we notice that protocol specific components can be obtained from different vendors, because there are no proprietary interfaces between them and every unit is enabled to have its own state machine.

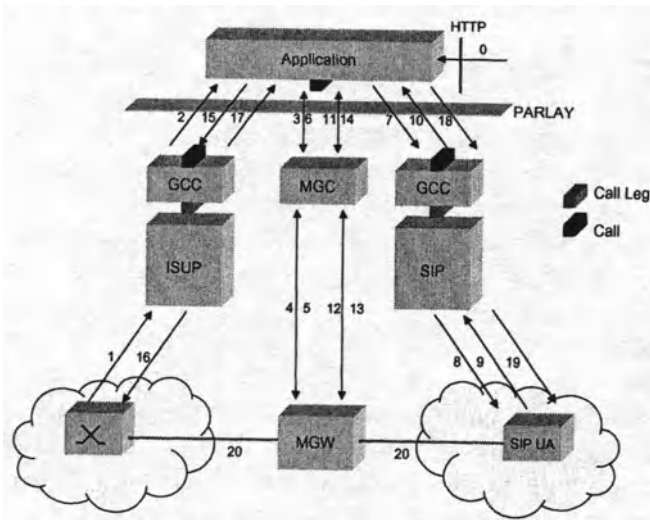


Figure 4. Call Setup Scenario with distributed Call Control

On the other hand, the traffic over Parlay is more than twice intensive, which increases call-setup times significantly. Another difference is that the complexity of the GCC has been moved to the application, which means that the application has to recognise the type of the call (e.g. IP-PSTN call) and to use the corresponding GCC unit. Furthermore, the application has to maintain states for different GCC units in a call. This complexity has to be implemented in every application (we assume independent applications), what considerably increases the developing time.

3.3 Distributed Call Control with external Gateway (Pseudo Softswitch)

The last architecture we propose is very similar to the architecture shown in the section above. The difference here is that there is no MGC component in the softswitch itself, which demands the usage of an external entity. Due to this reason we call this architecture pseudo softswitch. The call-setup is quite different in comparison to the two other architectures (see figure 5).

After receiving a call-setup request from the ISUP, the GCC of the ISUP side contacts the application (2). The application recognises that this call should be routed to the IP network and forwards it over ISUP stack to the MGC (3, 4). Note that a call is created in the application and in the GCC of the ISUP stack.

The MGC reserves resources from the MGW (5,6) and sends a SIP INVITE request to the SIP stack of the softswitch (7). Afterwards, the GCC of the SIP side notifies the application that determinates the address of the callee and contacts the SIP GCC (8, 9). Another call is created in the application and in the GCC of the SIP stack. The callee receives a SIP INVITE request and replies with an OK response (10,11). Now the SIP stack informs the MGW over the MGC (12-15) and notifies the application and the callee that the call-setup has succeeded (16,17). Finally, the MGC reports the call-setup results to ISUP stack, which forwards this report to the application and to the PSTN switch (18-20).

In this architecture we have tried to avoid drawbacks of the above section by reducing the traffic over Parlay and by making the application less complex in terms of call control. There are two calls in the application now, which can be maintained separately. This makes the application more flexible and reduces the developing time. Furthermore, the application does not need to communicate with the MGC. Protocol specific components are independent from each other and so they could be obtained from different vendors. One drawback is the usage of an external MGC component. It means that this softswitch does not support MGCP and therefore it cannot route calls between different networks without an external MGC component. Furthermore, the application has to perform the mapping between gateway addresses and user addresses.

distributed call control instead of the pseudo softswitch. Both architectures have in common that here the legacy systems could mostly be reused. With the usage of Parlay, existing IN applications are utilizable, and, except for the integration of Parlay, no changes in call control are necessary. Support of other networks and extension of functionality could be added step by step in module manner. So we call this approach a moderate architecture in terms of convergence.

REFERENCES

- [1] The Parlay Group, "Parlay Specification 2.1", <http://www.parlay.org>
- [2] ITU-T, "Q12xx, Intelligent Network", ITU-T Standards
- [3] S. Bessler, A.V. Nisanyan, K. Peterbauer, R. Pailer, J. Stadler, "A Service Platform for Internet-Telecom Services using SIP", 6th IFIP Conference on Intelligence in Networks, SMARTNET 2000, September 2000.
- [4] R. Pailer, J. Stadler, "A Service Framework for Carrier Grade Multimedia Services using Parlay APIs over a SIP System", 1st ACM Workshop on Wireless Mobile Internet, WMI 2001, July 2000.
- [5] Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON): Network Architecture and Reference Configurations; Phase II: Scenario 1 + Scenario 2. TS 101 313 V0.4.2. February 1999.
- [6] Telecommunication Standardization Sector of the International Telecommunication Union: Integrated Services Digital Networks (ISDNs). Recommendation I.120. March 1993.
- [7] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg: SIP: Session Initiation Protocol. IETF RFC 2543. March 1999.
- [8] Telecommunication Standardization Sector of the International Telecommunication Union: Packet-Based Multimedia Communications Systems. Recommendation H.323. November 2000.
- [9] M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett: Media Gateway Control Protocol (MGCP) Version 1.0. IETF RFC 2705. October 1999.
- [10] J. Postel: User Datagram Protocol. IETF RFC 768. August 1980.
- [11] M. Handley, V. Jacobson: SDP: Session Description Protocol. IETF RFC 2327. April 1998.
- [12] J. Postel: Internet Protocol. IETF RFC 791. September 1981.
- [13] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: A Transport Protocol for Real-Time Applications. IETF RFC 1889. January 1996.

- [14] S. Kent, R. Atkinson: Security Architecture for the Internet Protocol. IETF RFC 2401. November 1998.
- [15] J. Postel: Transmission Control Protocol. IETF RFC 793. September 1981.
- [16] M. Arango, C. Huitema: Simple Gateway Control Protocol (SGCP) Version 1.1 Draft. IETF Internet Draft. July 1998.
- [17] I. Elliott: IPDC: Media Control Protocol. IETF Internet Draft. August 1998.
- [18] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, J. Segers: Megaco Protocol Version 1.0. IETF RFC 3015. November 2000.

Mobility Management in a Plug and Play Architecture

Mazen Malek and Finn Arve Aagesen

Department of Telematics, Norwegian University of Science and Technology, Trondheim, Norway

malek@item.ntnu.no, aagesen@item.ntnu.no

Abstract Mobility Management functionality in the Plug-and-Play (PaP) architecture is presented comprising different approaches for *Actor*, *Terminal*, *User* and *Session* Mobility Management. We explore a few issues related to implementation design and propose a set of components to facilitate the deployment of this platform in the available PaP applications. The PaP architecture is briefly introduced and subsequent definitions and terms are dealt with. The architecture itself is based on a theatre metaphor, in which plays define the functionality of the system. PaP components are realised by actors playing roles defined by manuscripts. An actor's capabilities define his possibilities for playing various roles. The mobility management is introduced to add a capability to handle any move of any constituent of the system. This could range from moving functionality, or actors, to moving users of the system. Throughout the paper we try to give a survey on the various mobility cases and make an effort to demonstrate an early set of mobility management algorithms or methods.

Keywords: Plug and Play, mobility management, teleservice architecture, active networks.

1. INTRODUCTION

Grade of network intelligence is defined as *the efficient flexibility* in the introduction of new teleservices and the *efficient flexibility* in the execution of teleservices. IN (Intelligent Networks) [ITU92], TINA (Telecommunication Information Networking Architecture) [TINA95],

Mobile Agents and Active Networks ([Bies97], [Bies98], [Raza99], [Tenn97]) are all solutions aimed to improve the network intelligence.

Plug-and-play (PaP) for telecommunications means that the hardware and software parts have the ability to configure themselves when installed into a network and then to provide services according to their own capabilities, the service repertoire and the operating policies of the system. Plug-and-play functionality means uttery increase of network intelligence. The concept PaP stems from the personal computing area. PaP simply means that you plug-in and then the system works. In these systems, the plugged in component as well as the framework has a *predefined* functionality. We denote this *static* PaP. A more general kind of PaP is when the plugged-in unit has a set of basic capabilities, but its functionality is defined as a part of the plug-in procedure and it can be changed dynamically. We denote this as *dynamic* PaP. An example is a cellular phone, which obtains the services it provides depending on its inherent capabilities, which user that logs on, and which network it is attached to. *The focus of this paper is on dynamic PaP, and from now on PaP means dynamic PaP.* For a detailed description of the PaP architecture and the ongoing research activities around this project please refer to: (<http://www.item.ntnu.no/~plugandplay>).

For purposes of more flexible computational capabilities, utilisation and adaptation of wider range of network services and possible introduction of network management tasks, our PaP model should be enhanced with various mobility management schemes. This comprises *Actor Mobility*, *Terminal Mobility*, *User and Session Mobility*.

The very basic requirement in this regard is the ability to move already instantiated functionality among nodes or components. In this paper we refer to this by *Actor Mobility*. Actor is one of the core concepts of the PaP architecture. An Actor constitutes a Role-Figure by behaving according to a defined role. An Actor has the capability to play different roles and then to constitute various Role-Figures. An Actor can make up different roles during its lifetime. Movement of functionality to the generic Actor by the time when the Role-Figure is instantiated or changed is already a core mechanism of the PaP architecture. Referring to the concept model in Figure 1, the *Actor Mobility is the physical movement of Role Figures*.

Now what is the difference between this PaP architecture, Active networks and Mobile Agents [Bies97]. The PaP architecture is based on Role-Figures, which have the power of EFSM (Extended Finite State Machines). All network and teleservice behaviour are assumed to be based on the same PaP architecture, in which Role-Figures get their behaviour based on Code-on-Demand. Actor mobility in this architecture is different from Active Networks and Mobile Agents in the sense that movement conserves state and relationships to already ongoing sessions with other Role-Figures. The picture is not the almost independent piece of program that travels around with licence to do more or less independent actions. The picture is, instead, the movement of a program with well-defined behaviour,

and which already has a lot of bindings that need to be conserved. In our case the actors are just a representation of the functionality at a given node. When actors move the execution of the functionality is moved to a different node. Section 3 provides a clearer view of *Actor Mobility*.

Nowadays telecommunication networks provide the users with ever improving and flexible mobility in both wired and wireless technologies. Users could move their terminals more and more freely, and yet be able to access the same range of services. This ability adds a great challenge in terms of managing terminal mobility. *Terminal Mobility* is handled in Section 4. For the remaining types of mobility, the *User and Session mobility*, sections 5 and 6 give an overview of the proposed methods. Finally we give *summary and conclusions that include our further interests and future work*.

2. A PAP REFERENCE ARCHITECTURE

In this section we give a brief description of the PaP architecture, how does it define functionality and what are its basic constituents. The PaP was introduced in [Age99]. Important definitions are highlighted and the basic procedures are mentioned. Some figures illustrating the different object models and view points are also included to add clarity.

2.1 PaP functional object model

PaP components: are real-world active hardware and/or software modules. These can be *combined hardware/software* modules with one or more external hardware interfaces, or *pure software modules*. These must interface with a software platform capable of running PaP application software. PaP components are composed from (one or more) interacting instances of PaP functional objects, where each instance is defined by reference to an object type. This means that the PaP *component* functionality is defined by a *functional object model* consisting of *functional PaP objects*. ISO's reference model for Open Distributed Processing (ODP) [Duts96] defines the enterprise, computational, information, engineering and technical viewpoints. The viewpoints of *primary interest* with respect to PaP are the *computational* and the *engineering* viewpoints. The PaP components are basically engineering viewpoint objects. The relationship between important PaP concepts is shown in Figure 1.

PaP Actor: An actor is a generic object with a generic behaviour. Actors are able to behave according to a *manuscript*. The *repertoire* consists of *plays*, which are defined by *roles* and the role is formalised by a *manuscript*. These concepts have meaning similar to as those that are used in the theatre context. The manuscript is the functional PaP object type definition. An instance of a PaP functional object, also here denoted as a *role-figure* is realised by an actor, which is executing the manuscript.

PaP Manuscript: defines the *entire* behaviour of an actor. *Role-session* is a projection of the behaviour of the actor with respect to one of its interacting actors. The entire role as well as the role-sessions are EFSMs (Extended Finite State Machines). An actor also has a defined set of *capabilities*, which is the ability or power to do something. The capabilities are the result of the available hardware functionality connected to the hardware executing the actor software behaviour, but also the quantitative aspects such as processing capacity [Aage99].

PaP Play: is a defined autonomous functionality. The play defines the context for relationships between PaP objects as well as their behaviour. One important PaP object instance necessary to initialise any play is the *director role-function*. Director behaviour is also defined by an instance of a play. An actor has three distinct behaviour phases: 1) the plug-in phase, 2) the play phase and 3) the plug-out phase.

PaP support functionality: The following functions are needed: PlayPlugIn, PlayChangesPlugIn, PlayPlugOut, ActorPlugIn, ActorPlugOut, ActorBehaviourPlugIn, ActorPlay, RoleSessionAction, Subscribe, ActorChangeBehaviour, ActorBehaviourPlugOut and ChangeActorCapabilities. For details see ([Aage99], [Joha99]). The functions: ActorBehaviourPlugIn, ActorPlay and ActorBehaviourPlugOut comprise the initialisation of a generic actor pending for a manuscript, performing the manuscript, and finally making the actor pending for a new manuscript. This functionality with the addition of ActorChangeBehaviour is denoted as the basic PaP functionality. The actor is initialised by first activating its director. An actor negotiates with a director role-figure in order to obtain its behaviour. The director role-figure will create an instance of a manuscript with all necessary parameters bound particularly for the actor. The director role-figure also acts as a binding object, which helps to establish communication or interactions among actors. After receiving a manuscript from the director role-figure, an actor will start acting according to the specification described in the manuscript. From this point on in time, the actor becomes autonomous and independent of the director role-figure and constitutes an application role-figure until it terminates or wants to change its behaviour.

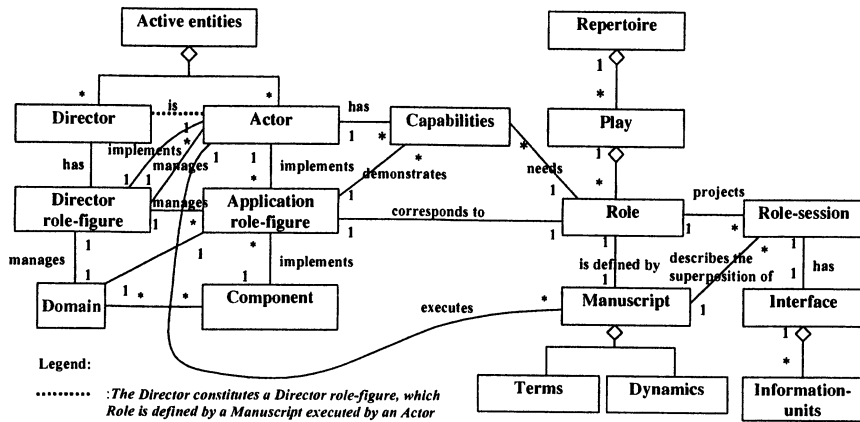


Figure 1. PaP concepts

2.2 PaP Engineering Model

As illustrated in figure 1, there are two types of role-figures, the *application role-figure* and the *director role-figure*. A relation between an application role-figure and a director role-figure must always exist. Relations between director role-figures will give a possibility to obtain a distributed solution for the director role. A PaP system with more than one director needs *administrative domains* to manage the federation of responsibility between director role-figures. Figure 2 shows the structuring of the needed functionality into five layers. All layers in except for *PaP specific applications* and *non-PaP applications* are completely independent of the applications themselves. The PaP functionality will have to interface to some infrastructure technology at the bottom layer, and may interface with any type of non PaP application through the top layer.

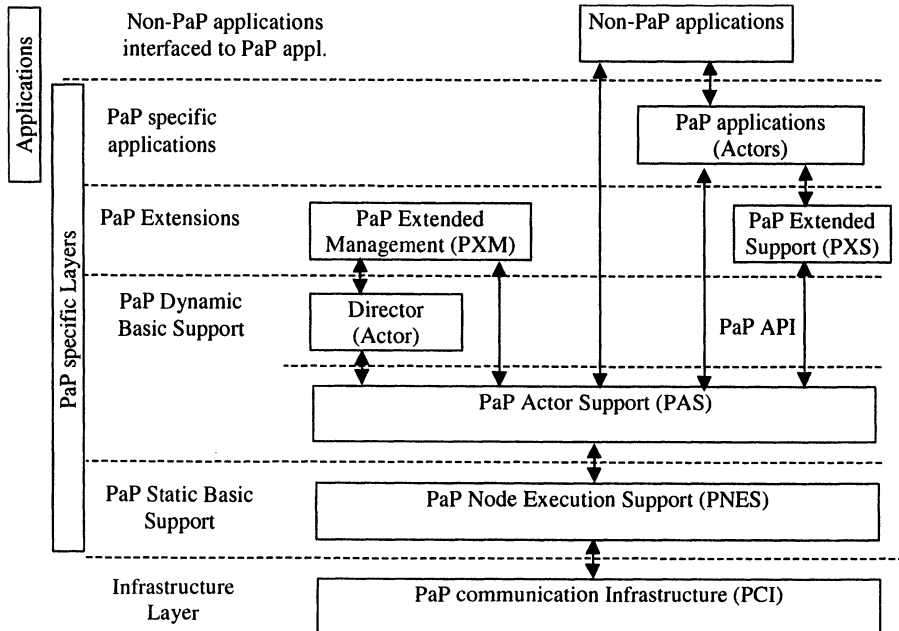


Figure 2. PaP layered model

A *PaP communication infrastructure (PCI)* architecture based on standard solutions, will usually consist of three layers with the operating system functionality (e.g. Unix or Windows) at the bottom, the network communication functionality (e.g. TCP/IP) in the middle, and some distributed system solution (e.g. CORBA ORB or Java RMI) at the top. The PCI top layer may be omitted, but that will require a more complex implementation of the interfacing module PNES if the PaP functionality require a distributed system solution.

PaP Node Execution Support (PNES) makes it possible to run PaP software on a node, and for PaP functionality (i.e. executed by actors) on different nodes to interact with each other. PNES is able to receive requests from other PNEses, interpret these requests and take proper actions. PNES will also do start-up and initialisation of PASEs or PCIs if that is required. PNES implements the PaP functionality that is termed the *PaP Static Basic Support* in the model. Static in this sense means that changes/extensions of the PNES functionality must be backward compatible with earlier versions because this functionality represents the “bootstrap” that is necessary to be able to run PaP applications. Only this functionality must be manually installed at a node before PaP applications can be installed and activated.

PaP Actor Support (PAS) makes it possible to create actors within the context of an operating system process/thread, to give these actors behaviour, and to communicate information between these actors and their environments.

Director is both responsible for the management of the PaP application definitions, i.e. its part of the repertoire- and manuscript-bases, and for the management of information concerning actors, i.e. its playing-base. A director is involved in many of the functions related to the services provided by PAS.

3. ACTOR MOBILITY

Actor mobility here stands for the movement of instantiated functionality at a node, or the Role-Figure executed by an actor. Actor mobility could be achieved by extending actors capabilities with movement ability, and by keeping track of this movement. In other words, the actor location specific information should be updated whenever an actor moves. In the PaP terminology we use Global Actor Identifier (GAI) as an object that maintains actor specific information regarding which PNES and PAS instances the actor executes in. Definitely, the association between actors, directors, actor-actor relations and capabilities should be dealt with carefully.

Actors move due to requirements on resource limitation, configuration change, functionality change, etc. In order to support actor mobility, the system should be able to deactivate and reactivate moving actors transparently. Therefore moved actors should be able to continue their behaviour from the point when they started to move. For this we need to define the concept of *actor state* that defines actor's state of a play. This aims at defining different phases of actor's behaviour and consequently the movement process will be reduced into an extended request of ActorPlugIn. As far as the implementation design is concerned, actor manuscripts should define actor's behaviour and interactions with other actors using state-based objects. On the other hand, the movement (transition) of actors should be handled using a proper method. In the following we will introduce three method case studies that are based on replicating moved actors, using proxy to forward requests and applying a centralised agent. But before that we need to figure out what are the possibilities of actor mobility.

When actors need to move from PAS to PAS or from node to node (a node is equivalent to a PNES instance in the PaP terminology), full update of actor specific information should be applied at all instances that are related to this actor. Whenever an actor is activated it is associated to a PNES, PAS and a home director instances. PNES holds the GAI that is needed to route requests coming to this actor. PAS is the application (process or thread) in which actor objects are executing. Directors are related to actors through the definition of actors "home interface". This definition is crucial for directors to keep track of actors' location. Actor movement could be classified into three different scenarios:

1. Actor movement from PAS to PAS - in the same PNES instance
2. Actor movement from PNES to PNES – having the same director
3. Actor movement from PNES to PNES – having different directors

3.1 Actor Mobility Management methods

AMM method 1: The first method to be utilized is based on employing a single copy of a moved actor at the new location and keeping all information bases as they are. All active role sessions of that actor should be moved to the new location. Routing of requests will be done as if there was no movement at all, except for that the old actor will forward all arriving requests to the new location. This method is restricted for certain types of movement scenarios, namely 1 and 2 as described in the previous section. The transition period seems to be short and simple, however further movements of the same actor should be carried out with care. Thus instead of forwarding move requests to the new location, an *ActorPlugOut* request and a new *ActorPlugIn* request should be initiated. Figure 3 illustrate a case where an actor moves to a new location, and then moves to another location. In this figure we apply a general type of request to the actors and denote them by request1 and request2 for simplicity. Also for the other requests, e.g. *ActorPlugIn* and *ActorMove*, we don't use parameters, as well as we avoid detailed interactions with directors.

AMM Method 2: This is a more general method that allows actors to move freely and change “*home interface*” or associated director. Actors as objects will be created at different locations and will not be kept at their original locations. A full update of the system is required here, in which the actor identity should remain the same except for its location that need to be changed. This method is more flexible from the performance point of view, but more complex in the implementation. An updated actor location will be always available at the entity registries of the middleware platform, such as RMI technology. Figure 4 demonstrates how this method works. Note that request2 is directed to the new location by the request sender.

AMM Method 3: This method is based on centralised agent that acts as a mobility manager within the context of PaP architecture, as in figure 5. The whole PaP functionality should be modified in a way that it could include a mobility service that actors would broadcast to or consult before they move or send requests respectively. “*LocationUpdate*” procedure should be used prior to movements and “*ActorDiscovery*” procedure should be used prior to request routing. This method would be the most flexible and robust to comply with movements of actors.

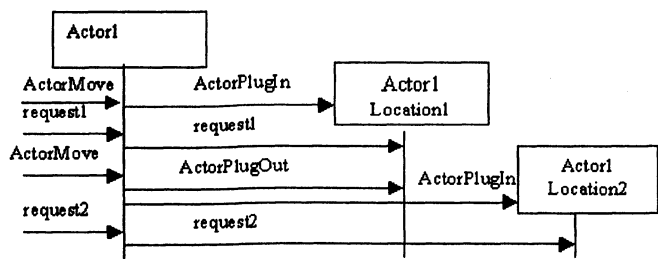


Figure 3. AMM method 1.

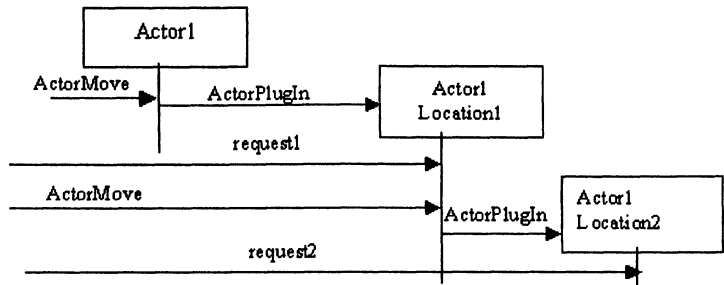


Figure 4. AMM method 2

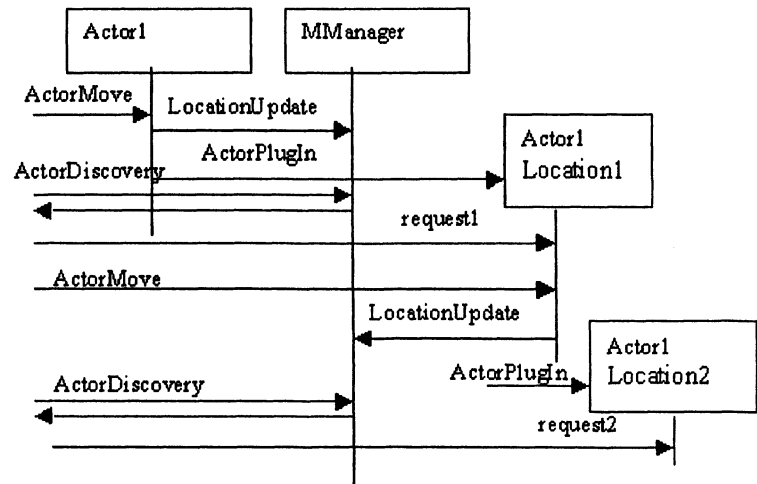


Figure 5. AMM method 3

4. TERMINAL MOBILITY

In the PaP architectural concept there is no special definition for terminals or nodes, instead PaP components are defined to be the real world active modules that take part in the PaP functionality. A terminal could be a network node that realizes the interface towards the end user. The mobility of a terminal or a node is defined by the change of its physical address. This is clearly seen in the engineering view of the PaP system, where at each node there is a PNES instance running at a distinct network address. To achieve mobility management for these moving nodes we need to keep track of their movements. So a manager should be responsible for updating the locations of all nodes that participate in a possible PaP application related to a movement. The knowledge of this central agent will be imbedded in the PNES objects, and whenever it moves all PNES objects are updated. This is better demonstrated by an example. Assume first a URL address as the initial access point where the PaP support software and application are available. Assume also a terminal, possibly a laptop, which would like to access this application. The first step is to download the PaP bootstrap and modify some configuration file stating the webserver from which the application will be downloaded and a default director interface, which used to be the localhost. Once this initial phase is ready the PNES could be started from which a play will be loaded and some actors could be plugged in to perform some role figures and interact with each other. Assume now the terminal will change location, e.g. connect to the network at different place using DHCP. Note that the terminal will not be switched off. The terminal will try to resume communicating according to the behaviour of its actors. On the other hand, the rest of the communicating nodes or terminals should be able to discover this move before, while and after it is being performed.

The proposed solution is based on mobility manager or MManager, which runs at an address known to all other nodes. MManager will run when the first PNES be started, and its network location should be part of the configuration file. Secondly, at each communicating node there should be a special agent performing location update and location query procedures. This agent could be called MAgent that stands for mobility agent, as in figure 6. MAgents are responsible for notifying MManager if there is a change in the node's address. On the other hand, they should send location query on every communication event with other nodes. The MManager responds to location updates by updating the node's location, and replies with the current location of the requested node upon receiving a location query. All in all, terminal mobility management is a kind of play that runs in the background of any PaP system that supports terminal mobility.

To obtain a clearer view of how nodes interact figure 7 shows a possible node-to-node interaction and their corresponding actors. This figure corresponds to moving node 1 in figure 6. The Actor-environment-

execution-module (AEEM) is a process/thread that executes a collection of actors with associated PaP Actor Support (PAS).

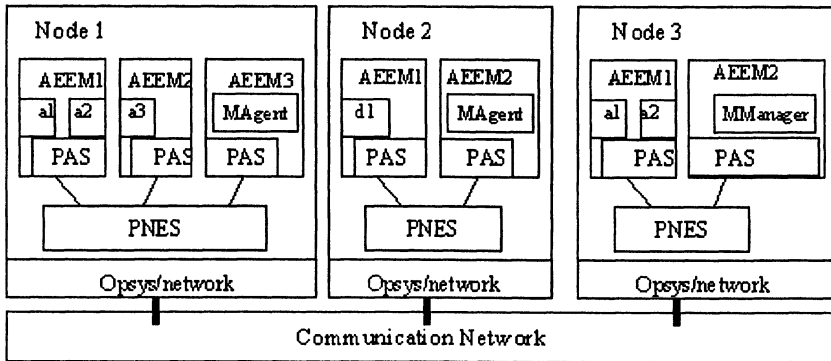


Figure 6. Engineering view of three PaP nodes with terminal mobility management.

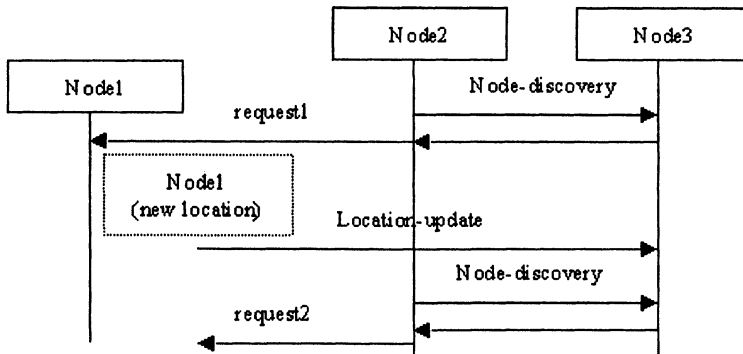


Figure 7. Node interaction before and after node moves

5. USER MOBILITY

This is also known as personal mobility, which is the utilization of services that are personalized with end users' preferences and identities independently of physical location and specific equipment. In this context we need to define the user and the services in a candidate PaP system. Services are carried out by service components, which are actors. In order to achieve user mobility we need to dedicate a special actor to behave as a provider agent [Axel98]. In our proposal we suggest to assign an agent to

each user, which is an actor or a set of actors carrying out system interface towards the user. This will be called User Agent (UAgent).

When we discuss user mobility we are actually talking about users and their existence in a certain domain. So services that are utilizable for a user at certain domain are not necessarily available at another domain. If a user could access all services provided to him by his home domain through another domain then these domains support user mobility. In the following we suggest a solution to achieve this. When a user moves from his home domain into a visitor domain he will first access the PaP service provided by the visitor domain's director, or director 2 in figure 8 and 9. This service should be able to grant some users visitor status (a guest type access to basic functionality). After some initial interaction with this domain, this user will be offered to access his home domain. Meanwhile, the user is a user of the visitor domain, so he requires a UAgent. However, as this user chooses to access the services provided by his home domain then a Visitor Agent (VAgent) should be assigned to him that is controlled by the home director, director 1. VAgent has similar definition to UAgent.

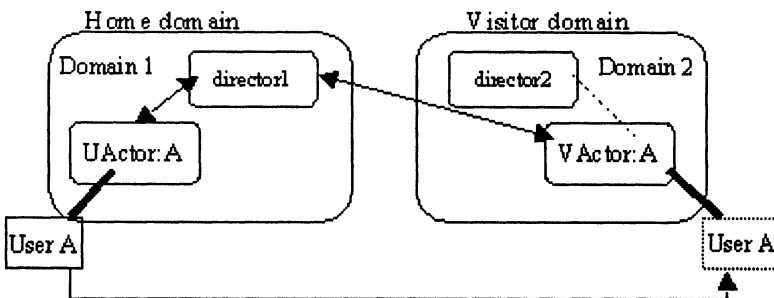


Figure 8. A proposal to handle user mobility

To realize personal mobility service, personalized environment is constructed according to the user profile of the end user. According to the service architecture of TINA-C this is defined as an aggregation of four informational objects: usage context, service profile, session description and user registration. The user profile object should be moved to the visited domain in order to construct the user's environment. In our architecture this issue is less complex as we associate the VAgent to the home domain, or home director. By this way, any needed objects, e.g. the user's profile, could be accessed and downloaded from within the VAgent without any interaction with the visited domain. The main issue here is, however, at the beginning of the registration process. We should note that these domains are not business administration domains, they are instances of executed or performed plays controlled by directors.

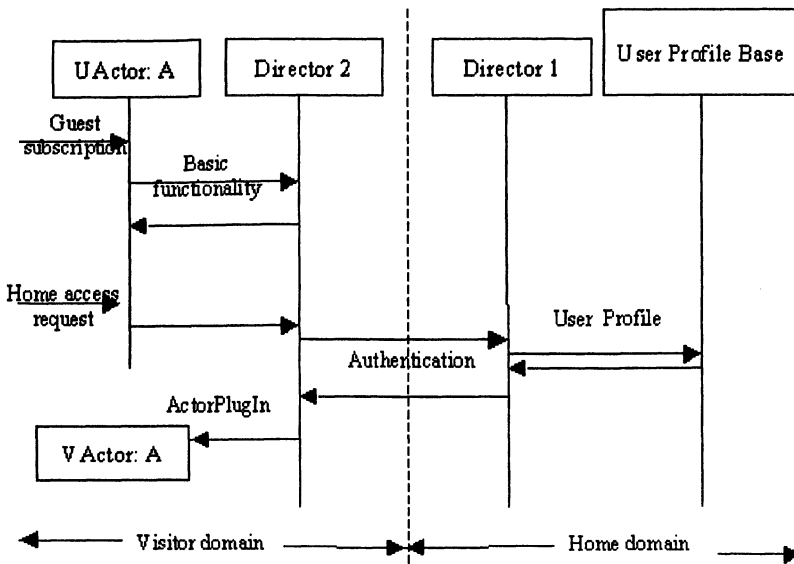


Figure 9. User mobility support: subscription, authentication and configuration.

6. SESSION MOBILITY

Session is an interaction collection who's goal is to satisfy the goal of a service by performing activities during a specific period of time. Session is also associated with the allocation of resources that are necessary to execute the aimed service. There are two types of sessions: User/Access and Service/Communication sessions.

Here we should recall the UAgent from the previous section and give it a broader definition: a computational object that manages user's preferences and performs all the operations and tasks required by the user. We should recall also that PNES object carries enough information about the node where the user is seeking access.

For the first type of session the user actor concept seems to be capable of providing this mobility. A user is granted an access to the system by assigning a UAgent. When it demands the same service from different domain a VAgent is assigned. To support the second type of session mobility we need to introduce a new role to the PaP architecture, this is the *session manager*.

We shall consider two requests to serve as a basis for session mobility: *suspend* and *resume*. A session is always initiated whenever a user is subscribed to the system and demands one of its services. The session manager is invoked and asked for a session ID. Now on, this session ID will be addressed at any future operation for session mobility. Since there is

always a special actor associated with users behaviour, UAgent or VAgent, and since these actors behave according to some state transition scheme, then resuming of any suspended session is achieved by recapturing the actor's behaviour at the state where the session was stopped.

7. SUMMARY AND CONCLUSIONS

In this paper a mobility management functionality based on a PaP architecture was introduced. Support for Actor, Terminal, User and Session mobility is the main part of a platform realising this functionality. Some parts of the proposed platform are being integrated to the available demonstrator application, the tele-school application, which is based on Java RMI technology. The Actor mobility Management methods are the first parts to be integrated, as they don't require a major shift to the basic functionality. The AMM1 method in particular provides basic and yet powerful actor mobility to the PaP architecture. The idea is basically an actor proxy scheme for moving actors.

The paper dealt also with terminal mobility that is an introduction of mobile nodes. Some implementation specific details were elaborated, and various aspects of the modified architecture were mentioned. New requests and procedures will be needed to cope with this added capability. However, this addition is totally inline with the overall terminology, as we outline it as a PaP play (or service) running in the background of a PaP application. At last, we introduced a support for user and session mobility. This is mainly based on TINA definitions and its concept of a user and session. We believe our proposal for such mobility types benefits from the simplicity of mapping actors to directors in our architecture.

While in this article we laid out a platform for the integrated provision of different mobility types, there are several issues that need further investigation. On the one hand, we need to break up the overall PaP architecture into subsystems to provide selected types of mobility, which for some will be in the support and for others in the application layers. On the other hand, we need to find efficient ways to map our methods and procedures into proper applications and communicating platforms. We expect the approaches presented here to play an important role in bringing the plug and play concept to telecommunications in a more practical manner.

REFERENCES

- [1] [ITU92] ITU-T, Principles of intelligent network architecture, October 1992.
- [2] [TINA95] TINA Consortium, TINA-C Deliverable: Overall Concepts and Principles of TINA V1.0, February 1995.

- [3] [Bies97] Andrzej Bieszczad and Bernard Pagurek, Towards plug- and play networks with mobile code, proceedings of ICC97, November 1997.
- [4] [Bies98] Andrzej Bieszczad and Bernard Pagurek and Tony White, Mobile Agents for Network Management, IEEE Communications Surveys, volume 1 number 1, 1998.
- [5] [Raza99] S. K. Raza and Andrzej Bieszczad, Network Configuration with Plug and Play Components, The Sixth IFIP/IEEE International Symposium on Integrated Network Management
- [6] [Tenn97] David L. Tennenhouse, Jonathan M. Smith, David Sincoskie, David J. Wetherall and Gary J. Minden, A Survey of Active Network Research, IEEE Communications Magazine, Volume 35 no 1, 1997, pages 80-86.
- [7] [Aage99] Finn Arve Aagesen, Bjarne E. Helvik, Vilas Wuwongse, Hein Meling, Rolv Braek and Ulrik Johansen, Towards A Plug and Play Architecture for Telecommunications, Proceedings of IFIP SMARTNET'99, Bangkok, November 1999.
- [8] [Duts96] Joubine Dutszadeh and Elie Najm, Formal Support for ODP and Teleservices, Proceedings of the IFIP/ICCC conference on Information Network and Data Communication, June 1996.
- [9] [Joha99] Ulrik Johansen, Finn Arve Aagesen, Bjarne E. Helvik and Hein Meling, Design Specification of the PaP Support Functionality, Plug-and-Play Technical Report, Department of Telematics, NTNU, 1999-12-10, ISSN 1500-3868
- [10] [Axe198] Axel Kupper, User Agent – An approach for Service and User Management in 3rd Generation Mobile Networks, International Conference on Telecommunications 1998.

II

SERVICES

Inside iTV- Interactive Commerce Services for Broadcast Digital Television

R. Chernock, P. Dettori, F. Schaffa, and D. Seidman

IBM T.J. Watson Research Center

Abstract Interactive television (iTV), i.e. the enhancement of the television viewing experience by the addition of multimedia content and services beyond the basic video and audio playout of a program, has a great potential for commerce, education and entertainment. By enabling the delivery to the home of rich multimedia content in coordination with broadcast television content, iTV can serve as a powerful platform for widespread home shopping and distance learning, among many other applications. The advent of high-bandwidth channels into the home, such as satellite, terrestrial, two-way digital cable and DSL, enhances iTV's power and the prospects for its acceptance. iTV has the potential to move from a curiosity to a common television enhancement, possibly with the ubiquity of the VCR.

Despite this, iTV has failed, to date, to fulfil its promise. While many fully mature technologies have taken years to find wide acceptance, in the case of iTV some issues can be isolated which can explain its slow approach. Before commercial iTV applications can gain wide acceptance, elements of the iTV network infrastructure must mature from their current state. Among these elements are those relating to the adoption of open standards and the enablement of purchase security and purchase tracking capabilities. For it to become commercially viable for network operators to offer iTV purchasing services to their subscribers, the capability must exist for every purchase transacted by a user to be fully secure and private, to be accounted for, and to be traceable. TV-based systems must be robust, user-friendly, and have reasonable response times.

In this paper, we present an architecture and a suite of commerce-related services for iTV networks designed to fill in some of the gaps in the current iTV software infrastructure, based on our experience in prototyping iTV systems. These services are of three types: operator services, advertiser services, and user services. After a brief review of current iTV commerce

systems, our proposed approach is described in detail. Finally we relate the details of one implementation of this architecture and these services in a prototype iTV environment. This successful proof of concept instantiated an iTV system with advanced commerce services as developed in the proposed architecture.

Keywords: iTV, Commerce Services, Controller, STB, Broadcast, MPEG-2, Transactions.

1. INTRODUCTION

The broadcasting industry is in the middle of a transition from analog to digital television. In addition to better video and audio quality, digital television allows carrying more standard definition programs in the same bandwidth spectrum, and provides the basic infrastructure for the deployment of new interactive services. There are a number of digital television standards that apply to different geographies, such as DVB [5] [18] in Europe, and ATSC [3] [4] and OpenCable in USA. A common feature among all the different instances of digital television systems is the capability to transport basic audio, video and data services and signalling information on the MPEG-2 Transport protocol [6]. Data services are the common layer which iTV services are built upon.

There are several embodiments of iTV services that require different system capabilities and that can be classified as broadcast-only services or services requiring a two-way communication channel. In the former case, interactive information is injected into the digital broadcast stream, detected and decoded by the Set-Top Box (STB), allowing the user to interact with content in the broadcast. The latter case adds the capability to perform transactions over a two-way network with back end servers, such as retrieving bank account information or sending back a purchase order.

While the deployment of basic digital television services is progressing quite smoothly, the deployment of iTV services has been slowed by a multiplicity of factors. In this paper we will focus on issues related to iTV commerce services, propose an architecture that could accelerate their deployment and present a prototype implementation.

2. ITV COMMERCE SYSTEMS

The iTV commerce infrastructure is built upon a set of building blocks, some of which are defined by standards and some others which are still proprietary, but migrating to open standards. A digital TV head end supports

the insertion of data into the MPEG-2 transport stream. The broadcast network carries the data to a digital STB, which in turn decodes the data and provides users with the capability to interact with the content. Transactions are routed by the network to back end servers, which will process information and will return data either via the broadcast channel or a two-way communication channel (phone line or two-way cable.)

In the past few years several iTV systems relying on proprietary back end systems have been deployed, producing expensive systems that are difficult to maintain and not open to multiple content and service providers. Today the focus is shifting to leveraging as much as possible the Internet infrastructure for interactive and commerce related services, bringing benefits that range from lower costs to more openness and interoperability to better capability to attract merchants and content providers.

The building blocks of e-commerce are the technologies of the World Wide Web, its protocols, servers and clients, along with the application development structures such as Java and HTML. Although the introduction of Internet standards and technologies has brought great advances to iTV, a number of issues are still open and slowing the adoption of iTV services. We will discuss some of these issues and proposed solutions in the following sections.

2.1 Current Scenario

The creation of iTV content today is still an expensive and time consuming task, and the general philosophy that should inspire it is "create once, re-use everywhere and many times". The creation of iTV content involves the creation of graphics assets, and the embedding of links either to self-contained content (e.g. content sent also in the broadcast as part of the ad) or to online servers (e.g. commerce servers.) The basic stages for content creation and distribution in an iTV system are authoring, embedding and distribution, ultimately to the viewer with a STB.

Once created, the ad will be delivered to the Broadcaster, together with a "metafile" describing the association of the interactive elements with the video/audio content, where it will be embedded according to a standard (for example, see the SMPTE standard for TV triggers [20]) or a proprietary mechanism. The resulting digital broadcast stream emitted by the broadcaster will include audio, video and data components that will contain the ad's interactive elements, with proper timing relationships with respect to the audio and video components. The STB will receive the regular video and audio program elements plus the interactive data. A STB with proper hardware and software capability will be able to process the data properly and present it in coordination with the video on the TV screen.

The user will interact with the STB to display text and graphics on the screen by means of a remote control. After receiving the initial data from the broadcast channel, the interaction may require loading more data either from the broadcast stream or from a two-way channel. If the user decides to purchase some item presented on TV, he or she will just follow a simple interaction path that typically would require some form of authentication by means of a numeric ID and a PIN or a smart card and a PIN. After authentication, the user will either enter payment and address information (i.e. credit card number) or will access information stored in the STB, in the smart card or in a remote wallet server to finalize the transaction. Finally the purchase request will be sent by the STB to the commerce server, that will process it and will send a confirmation number back to the STB. In some cases an e-mail confirmation with order details can also be sent, either directly to the STB or to some other user's e-mail account. The same flow applies both to physical and digital goods delivery (e.g. purchase of a music CD), but in the latter case the fulfilment happens over the network, by securely delivering the digital content to the STB hard-drive or to some other device (such a MP3 player) directly connected to the STB.

Figure 1 illustrates the current scenario for iTV commerce.

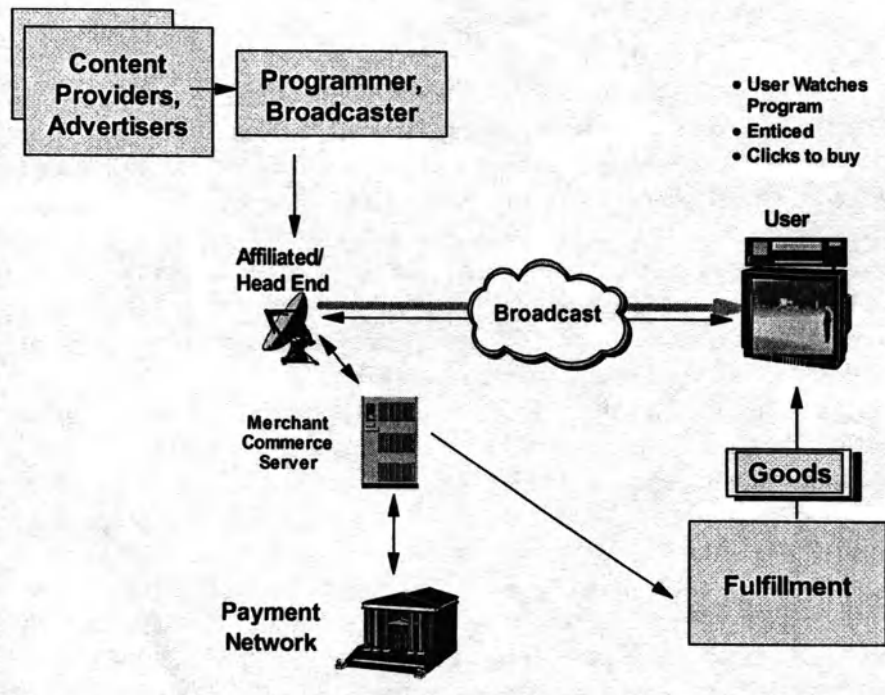


Figure 1. Current scenario for iTV commerce

As mentioned earlier, although this scenario is currently being deployed in some geographies, there are a number of issues that are slowing down its adoption in the market. We will outline these issues in the next section.

2.2 Issues

Network operators need to develop business models and agreements with advertisers and merchants using their networks for iTV commerce services. There is no mechanism today to enforce business rules among the different entities involved in the iTV commerce chain.

iTV content authoring is still an expensive process and ideally the same ad campaign should be re-used at different times in different markets. The same ad campaign may need to be linked to different merchants on different geographies. In the current scenario, users are typically directed to a specific commerce server that does not change over time or with the geography. This would require broadcasters or cable operator to have some mechanism for redirecting commerce related traffic to local merchants that would fulfil the order, or to dynamically update the interactive content related to an ad.

A chief component of the iTV revenue stream comes from advertising. Advertisers are moving from traditional TV ads to more sophisticated ads, and they require ways to track the popularity of an ad. Furthermore, methods to tune interactive ads would be also required. Again, there are no standard mechanisms today to track the lifecycle of an interactive ad and gather statistics on how and how many users interacted with it.

Statistics on commercial transactions need also to be gathered by the merchants and cable operators and reconciled in such a way that each party get its share of the revenue stream.

Finally, there are a number of user requirements to be satisfied, such as providing the capability to securely store information on the network, such as profile information, links to iTV ads and commerce offers. Typical STBs have very limited storage capability, except for those few STBs with hard-drives, making it difficult to utilize local storage for this purpose. Furthermore, links must be maintained and accurately updated. Everyone would agree that a TV user would not accept being presented with a page containing some arcane HTTP code for "page not found", as happens today for PC users on the Internet. Thus a mechanism to store and check consistency of links must be provided in the network.

2.3 Proposed Architecture

We have designed an architecture that capitalizes on the current iTV infrastructure and plugs in a new element that will allow a smooth

introduction of iTV services. A controller is introduced for the interactive television commerce environment, acting as a central point of control and information for enabling and facilitating information distribution and transactions among the entities involved in the iTV commerce value chain. The controller offers a low-cost solution for the introduction of iTV commerce services, in contrast with alternative systems, which involve the redesign of the network's IT infrastructure. It is designed to be retrofitted into an existing network.

The controller allows network operators to keep track of all flows in the network transactions that take place over their networks between clients and any servers.

The controller manages, adds to, and modifies the content of interactive ads and programs that are broadcast to their viewing audiences. It supports redirection of network traffic to the appropriate servers for local fulfilment of goods and services which are advertised in nationally broadcast ads, and which are purchased through interactivity with those ads.

The controller reports to network operators, advertisers, and viewers, on purchasing activities that are conducted on iTV networks, based on consensual information gathering and analysis of this data.

Figure 2 illustrates our approach. The controller is inserted in the digital TV network, and it is typically located in the cable operator's head end.

An additional element, the Service Control Point (SCP) communicates with multiple controllers to centrally manage them. The SCP updates database tables in the controllers, gather aggregate statistics and enables a single point of control and interface for distributed controllers.

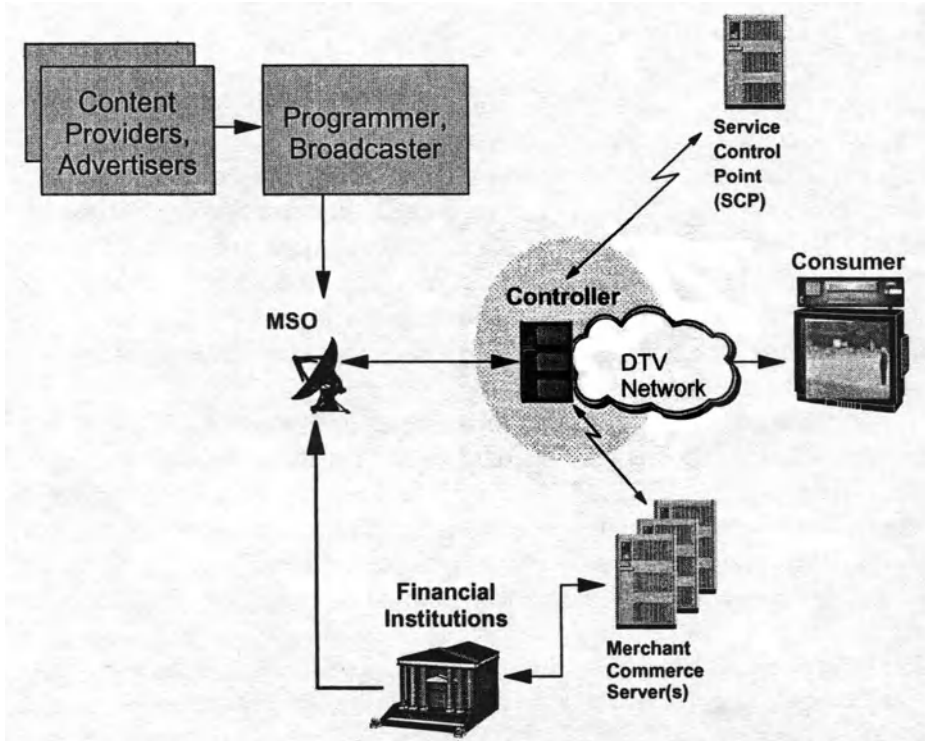


Figure 2. Proposed scenario for iTV commerce

3. SUITE OF ITV COMMERCE SERVICES

We will describe in this section the benefits and services supported by the architecture we discussed in section 2.3.

3.1 Enablement of Interactivity

Interactive content, in the typical scenario considered by this paper, is added to the basic video and audio content prior to broadcast or at broadcast time. However, it may be useful to selectively enable the utilization of interactive content by client applications, for example if an interactive commercial is aired after the time period in which the added content is valid, such as a limited-time offer. This selective enablement can be located at the controller (see section 2.3) or at each individual set-top box.

3.2 Transaction Capture

Typically, some display of multimedia content to the viewer during a program or commercial begins (and motivates) the interaction process.

If (as is the case with purchases) the interactivity proceeds to the point where connection to a server is necessary, this connection is facilitated by use of URIs (Universal Resource Identifiers) [17] that are included in the added content. At this point, the interaction could proceed between the STB and the server; however, this would leave the network operator out of the loop, not being aware of the type and amount of commerce traffic being handled on the network.

It is necessary for the controller to intermediate between the STB and the server in order for the network operator to be kept in this loop. Information on the items being purchased, their quantity, price, etc., can be extracted from the HTTP messages moving between the STB and the server, subject to privacy and security considerations (see sections 3.7 and 3.8.) Each transaction conducted over the network should be logged by the controller and made accessible (in the aggregate) to the network operator, who may receive a fraction of the profit. Statistics may be generated for the individual or in the aggregate, regarding purchase patterns, interests, etc. Again, this is subject to the approval of the clients, and must align with their privacy and security concerns.

3.3 URI Translation

As described above, an interactive TV advertisement is created by a content creator, typically the same entity which created the audio and video content. Alternatively, the interactive content may be added to a broadcast program. The enriched content (audio, video, and data) is broadcast and received by the client's STB that is running an application to detect and extract this data. Typical data includes URIs, text describing sale items, graphics such as icons, and "metadata", data that describes the data that has been added to the broadcast. The advertisement may be broadcast nationally or locally.

A service control point (SCP) (see section 2.3) has prepared the controllers, which reside in the head ends of network operators offering TV e-commerce services, to prepare for the broadcast. "Redirect URIs" are specified by the SCP and stored in the controller, in order to map the URI from the broadcast ad to the appropriate local server. When the user interacts with the advertisement (typically by pressing a key on the remote control), the controller recognizes the advertisement (by use of a unique ad identifier) and maps it to a local server, as specified by the SCP.

The transaction then proceeds between the client and the local server, with the controller intermediating to record user activity and transaction details. This mechanism allows localization of interactive content for different markets and spreading the transaction loads between different servers.

3.4 Activity Tracking

At any time, the client may request a review of his/her purchase activity. The controller maintains records for all clients' purchases in the database, and responds with a display of all purchases for this client that were transacted over this network. The client may also request the status of a given purchase. The controller then assists in the tracking of a purchase, based on the confirmation number, which it maintains for each purchase, and the purchase tracking mechanisms available for the item.

3.5 Bookmaking Purchase Opportunities

Because of the dynamic nature of television, purchase opportunities may be seen by clients who wish to interact with them but are unable to do so in the time allotted. The controller enables the client to set a "bookmark" during an iTV advertisement to indicate an interest in the advertised item. Each client's bookmarks are stored at the STB or at the head end. The client may request to be reminded, periodically, of the bookmarks he/she has set by way of an icon or audio beep. The controller takes care of purging bookmarks when the purchase opportunities they reference expire.

3.6 Offline Purchasing

At any time, the client may request to view the "channel" of purchase opportunities. This may be in response to an interactive ad which the client saw but did not interact with, or for items which the client did not see. The controller maintains a database of all interactive ads and displays the icons related to these ads to the client. Ads are characterized by a number of keywords, and these keywords are used to display the purchase opportunities in categories such as Food, Sports, Clothes, etc.

3.7 Data Mining

Subject to the consent of the viewers, information collected by the controller (see section 3.2) can be used to generate statistics on purchasing and viewing habits, in the aggregate or for individuals. The latter can be

useful in the targeting of specific opportunities to viewers with particular interests. Commercial packages like IBM's Business Intelligence can be utilized to generate statistics from the raw data thus collected.

Among the statistics of interest can be the effectiveness of an iTV ad for a particular demographic, at a particular time, on a particular channel or during a particular program. Statistics can reflect the number of "clicks" (i.e. remote control keys pressed in response to offered iTV content), whether or not a purchase was completed. Advertisers may be interested in the interest generated by an iTV ad and how highly the viewers' response to interactivity is correlated to purchases at that time or in the future.

3.8 Privacy

One issue, which may slow the adoption of iTV purchasing, is concern over privacy. Viewers do not want anyone monitoring the channels that they choose to watch. All information which is being gathered on viewer activity (purchases, channel changes, etc.) must be collected strictly with the consent and prior approval of the viewer.

The controller, intermediating between the STB and the servers, needs to "snoop" on only a very limited amount of the traffic between the two, e.g. total price of items to be purchased, amount of sales tax, etc. The remaining traffic can pass through the controller in encrypted form.

Information on individual activity can be discarded, and retained only in the aggregate. Incentives can be offered to viewers who wish to volunteer more individual information.

3.9 Security

The controller represents a vulnerable node in the guarantee to the viewer of secure transactions. It is of paramount importance that every precaution be taken to ensure security of information and funds. Since the STB may not have the capability of reading a smart card and the entering of detailed viewer information (such as credit card number, address, etc.) may be difficult, the controller may need to know this information in advance and provide access to it as a service. Access to this information for purchases can be secured via a PIN number, dedicated STB software or hardware, or other means.

3.10 Hand-offs

It is to be expected that while viewers might be interested in and attracted by the dynamic nature of multimedia iTV purchase offers, many will pursue

such offers only to the point of getting more information, and stop short of making a purchase, for several reasons. Despite the best security technology, many users (especially those who are not computer-savvy) will be reluctant to make a purchase using their television as the medium, for fear of privacy and security breaches. In order to coordinate the distance and in-person shopping experiences, "hand-offs" may be used, i.e. the initiation of a purchase (or purchase inquiry) in one medium which is followed by its continuation in another medium.

The controller would facilitate the management of shopping across media boundaries, by maintaining status information for each user on each initiated transaction, and enabling transfer of shopping data between user devices.

4. IMPLEMENTATION

Our research team has implemented a prototype controller system demonstrating several of the features described above. The system consists of three main components:

1. Transaction Capture Engine
2. Application Server
3. Persistent Storage Server.

The controller is inserted in the iTV network to enable advanced interactive services. It provides services to the iTV users, captures transactions and redirects commerce-related traffic to external commerce servers.

We have implemented and tested the controller in a pilot iTV commerce system built in our lab, consisting of a scaled-down cable head end, an iTV authoring suite, a commercial Internet commerce server, and an off-the-shelf STB, demonstrating the viability of our architecture as enhancement for current iTV systems.

In this section we briefly describe the building blocks for the prototype controller, and the test system we have adopted for functional and interoperability testing.

4.1 Transaction Capture Engine

The Transaction Capture Engine is a programmable proxy responsible for capturing HTTP transactions. This data is used for later analysis of system traffic and user activity. The Transaction Capture Engine is built on top of the IBM WeB Intermediaries (WBI) [8]-[10]. The WBI Development Kit provides a convenient and flexible API for programming intermediaries on the Web. In simple terms, an intermediary is a programmable proxy. More

precisely, intermediaries are computational entities that lie along the HTTP stream and are programmed to tailor, customize, personalize, or otherwise enhance data as it flows along the stream.

From the perspective of an HTTP client, WBI acts as an HTTP request processor, receiving a request and sending a response. The processing that happens in between is completely controlled by the functions WBI is set up to provide. The WBI architecture provides a simple and powerful programming framework for developing any such Web applications.

A WBI transaction flows through three basic stages: request editors, generators, and document editors. Request editors receive a request and may modify it before passing it along. Generators receive a request and produce a corresponding response (i.e., a document.) Document editors receive a response and may modify it before passing it along. When all steps are completed, the response is sent to the originating client. A fourth type of processing element, a monitor, can be designated to receive a copy of the request and response but cannot otherwise modify the data flow. Collectively, we refer to Monitors, Editors, and Generators as MEGs.

In our implementation with WBI, a special monitor has been built to gather data from the HTTP session and store it in a data storage facility for later processing and analysis (see sections 3.2, 3.4 and 3.7.)

4.2 Application Server

The Application Server runs the applications that support services such as those described in section 3. The Application Server interoperates with the Persistent Storage Server to maintain metadata related to the TV programming and user-related data. In our implementation the Application Server has been built on top of the IBM's WebSphere Application Server [14].

WebSphere Application Server provides an environment for open distributed computing. Users and processes on a wide variety of platforms can interact by using the facilities provided by WebSphere. The WebSphere software platform delivers the first commercial server that provides J2EE integrated with tools that support Web services standards, such as Simple Object Access Protocol (SOAP) [13] and Universal Description, Discovery and Integration (UDDI) [11]. By implementing these open standards in ways that preserve and enhance the server's scalability and reliability, the WebSphere software platform delivers a highly productive environment for developing, deploying and integrating e-business applications. The incorporation of Web services opens a new door to building WebSphere software applications that dynamically interact with Web services using

SOAP, UDDI, Web Services Description Language (WSDL) [12] and XML [19].

The Persistent Storage is provided by Enterprise Java Beans (EJB) [15] Entity Beans, which provide the Object mapping to the database tables maintained by the Persistent Storage Server. The database tables store information such as user profiles, re-mapping of URIs to local servers, bookmarks, etc.

Business logic has been implemented with EJB Session Beans. One of the advantages of using EJBs for transaction-based services is the capability to provide ACID (Atomic, Consistent, Isolated, Durable) transactions. This supports the requirements of high reliability and consistency that are necessary to run transactional services.

User interface and management functions have been implemented using Servlets and Java Server Page (JSP) technology. JSP and Servlets interact with the EJB Session and Entity Beans to run the applications that support the services listed in section 3.

4.3 Persistent Storage Server

The Persistent Storage Server maintains all the tables used by the controller for its operation, such as metadata related to the TV programming and user-related data. It is based on the IBM's DB2 Relational Database [16].

4.4 Test System

The test system for the prototype controller is illustrated in Figure 3. iTV commerce content is created with the IBM HotMedia TV authoring package [21], and then is transferred to a broadcast server that sends the MPEG-2 Transport multiplex composed of video, audio and interactive data to a Scientific Atlanta QAM modulator.

The digital broadcast signal is received by a Scientific Atlanta Explorer 2000 STB, which runs a PowerTV HTML engine. A user can interact with iTV-enhanced content, and purchase items through the STB. When a commerce transaction is started, the controller tracks it and routes it to the IBM WebSphere Commerce Suite (WCS) server [22]. Digital media files can be purchased from the STB and transferred to a PDA. A subset of the services described in section 3 have been deployed and tested in this environment.

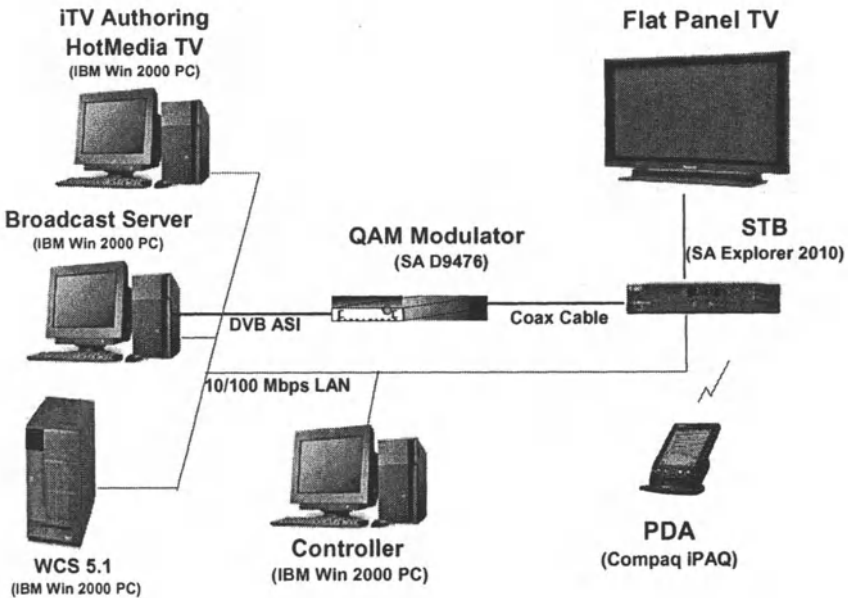


Figure 3. iTV commerce test system

5. CONCLUSIONS

The deployment of open iTV commerce systems has been slowed down by a multiplicity of factors. Business rules need to be defined and enforced among the entities involved in the commerce scenario. Furthermore, a number of operators, advertiser and user services need to be supported. In order to overcome some of these issues, we have designed an architecture that leverages the current iTV infrastructure by introducing a new plug-in element, the controller. The controller enforces business agreements between the parties involved in the iTV commerce flow, acts as a central point of control and information for data exchange, and provides the platform for the introduction of operator, advertiser and user services. The concept of controller as a service platform enables the introduction of new services beyond those described in this paper.

Our team has implemented a successful prototype demonstration system that embodies the concepts discussed above and interoperates with off-the-shelf iTV components.

REFERENCES

- [1] J. Paraszczak, F. Schaffa, D. Seidman, R. Chernock, P. Dettori and S. Depp. The Convergence of TV and PC - An Arranged Marriage. In 1999 Broadcast Engineering Conference Proceedings, NAB, 506-515, April 17-22, 1999.
- [2] R. Chernock, P. Dettori, X. Dong, J. Paraszczak and F. Schaffa, D. Seidman. Approaches in Enabling Electronic Commerce Services over Digital Television. In Proceedings of the 2nd International Conference on Telecommunications and Electronic Commerce, Nashville, TN, Oct. 6, 1999.
- [3] ATSC Digital Television Standard. In ATSC Standard, ATSC A/53 September 1995.
- [4] ATSC Data Broadcast Standard. In ATSC Standard, ATSC A/90, July 26, 2000.
- [5] DVB - Specification for Data Broadcast. EN 301 192, Edition 1.2.1.
- [6] ISO / IEC 13818-1 Generic Coding of Moving Pictures and Associated Audio: Systems.
- [7] ISO / IEC 13818-6 Generic Coding of Moving Pictures and Associated Audio: Digital Storage Media Command and Control.
- [8] IBM WBI Toolkit, available at <http://www.alphaworks.ibm.com/tech/wbidk>
- [9] Barrett, R. & Maglio, P. P. (1999.) Intermediaries: An approach to manipulating information streams. IBM Systems Journal, 38, 629-641.
- [10] Barrett, R., Maglio, P. P., & Kellem, D. C. (1997.) How to personalize the web . In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '97), Atlanta, GA
- [11] "UDDI Version 2.0 Data Structure Reference", June 8, 2001. Available at <http://www.uddi.org/pubs/DataStructure-V2.00-Open-20010608.pdf>.
- [12] Web Services Description Language (WSDL) 1.1, March 15, 2000. Available at <http://www.w3.org/TR/wsdl>
- [13] SOAP Version 1.2, W3C Working Draft 9, July 2001. Available at <http://www.w3.org/TR/soap12>.
- [14] IBM WebSphere Application Server. See <http://www-4.ibm.com/software/web servers/appserv/>
- [15] Enterprise Java Bean 2.0 Specifications. Available at <http://java.sun.com/products/ejb/docs.html>
- [16] IBM DB2 Relational Database: <http://www-4.ibm.com/software/data/db2/>
- [17] Uniform Resource Identifiers (URI): Generic Syntax", IETF Draft Standard, August 1998. Available at <http://www.ietf.org/rfc/rfc2396.txt>
- [18] DVB - Multimedia Home Platform, Revision 1.5, February 2000, SB280007.

- [19]XML - Extensible Mark Up Language. See <http://www.w3.org/TR/PR-xml-971208>.
- [20]Declarative Data Essence, Content Level 1, Interactive Television Triggers. SMPTE Draft 06/11/01
- [21]"Synchronized Data Services over MPEG-2 Digital Television Broadcast Systems", J. Brunheroto, R. Chernock, P. Dettori, X. Dong, J. Paraszczak, F. Schaffa, D. Seidman, European Transactions In Telecommunications Journal, Vol. 12, Issue No. 3 (2001.)
- [22]IBM WebSphere Commerce Suite. See <http://www-4.ibm.com/software/webservers/commerce>

GCP

A Group Communication Protocol for Supporting Closed Groups in the Internet

Mario Zuehlke and Hartmut Koenig

Brandenburg University of Technology at Cottbus

Department of Computer Science

email: {mz,koenig}@informatik.tu-cottbus.de

Abstract Video conferences have become a popular application in modern high performance networks. Advanced technologies allow their installation on desktop computers. Currently videoconferences are mainly deployed for large open group conferences in the Internet to transmit lectures, conferences, talks, and discussions. Multiparty video conference tools for closed group meetings are less broadly used. On the other hand, most meetings in every-day life are meetings of closed groups with a more or less confidential character. Unlike open group conferences closed group meetings require a sender related transmission paradigm, i.e. the sender must know the receivers of the data streams. This is usually solved by using a centralized group server for the group and the QoS management. The shortages of this approach are server failures and performance problems. The decentralized approach is more flexible, but it requires more sophisticated mechanisms to preserve the consistency of the management data at each participant. In the paper we present the signalization protocol GCP to support decentralized closed group video conferences. It provides an atomic, ordered, and reliable data delivery service and supports a dynamic join and leave of participants. We present two versions of the protocol: one for ATM networks using overlaid multicast connections and one for IP-based networks.

Keywords: Multiparty video conferences, closed groups, group communication, distributed group management, ATM, IP

1. MOTIVATION

Video conferences have become a popular application in modern high performance networks. Due to improved network, processor, and compression technologies, and reduced equipment and network cost as well video conferences do not any more depend on specialized studio equipment. Video conference systems are available on desktop computers meanwhile. Despite of years of development, however, video conference services are not that broadly used like services such as e-mail or WWW.

Currently videoconferences are mainly deployed for large video conferences for transmitting lectures, talks and discussions in the Internet [1]. Such video conferences are open group conferences. The sending paradigm is receiver related multicast. Everyone who subscribes to the multicast address can join the conference. The sender does not know who the actual participants of the conference are.

Multiparty video conferences for closed groups, e.g. project meetings, teleseminars, medical consultations, are less deployed. Most meetings in every-day life, however, are closed group meetings with a more or less confidential character. Under a closed group we understand here a tightly coupled group with a sender related multicast paradigm, i.e. the sender always knows the receivers of the data streams. The participation in the meeting is by invitation. Depending on the agreed confidence of the meeting authentication as well as encryption of the data streams are required. Note that not all closed group video conferences, e.g. teleseminars, require this just as not every e-mail must be encrypted. Videoconference systems for closed group meetings are mostly based either on circuit-switching networks like ISDN using the H.320 standard or on H.323 based systems in packet-switching networks. Examples are CU-SeeMe and Netmeeting.

In wide area networks it is still difficult to ensure a good Quality of Service (QoS) at the participants using these standards. Many video conference applications over the Internet use instead the Multicast Backbone (MBone) running the MBone video conference tools or the USMInT service. These applications support open group conferences. It is possible to set up closed group meetings by encrypting the media streams, but the key management and the speed of encryption are insufficient. Even more advanced encryption functions do not provide a sufficient confidentiality. It remains the possibility for an attacker to secretly attach to a conference if the multicast address is known to record the (encrypted) conference, to disturb or attack it. On the other hand, it can be expected that a convenient access to closed group video conference services will lead to a rapid growth of their use. This will exceed the bandwidth capacity of the MBone routers. For

setting up closed group video conferences, it is therefore more appropriate to support the sender related multicast paradigm.

Closed group video conferences require a group management to supervise the composition of the group, to control the access to shared resources such as the audio channel and the white board (floor control), and to tune QoS parameters. For closed groups, this is usually solved by a group server which centrally manages the group and takes all decisions. The main shortage of the centralized approach are server failures that terminate the conference. The server may also become a performance bottleneck due to many server accesses. Decentralized solutions are more flexible in this case. They avoid the performance problem and do not terminate the conference in case of a participants' failure. The distributed approach, however, requires a more sophisticated design. So far only a few distributively managed video conference systems are reported [1], [2].

In this paper we present the signalization protocol GCP which has been designed to support a distributed group management in closed video conference systems. It provides an ordered, reliable, and atomic data delivery service. We present two versions of the protocol: GCP_{ATM} developed for ATM based video conference systems and GCP_{IP} designed for IP-based network environments. The paper is organized as follows. In Section 2 we discuss the requirements to the group management of distributed video conference systems and give a short overview of related work. In Section 3 we first show how an ordered, reliable, and atomic data delivery service can be provided in ATM networks using overlaid multicast connections. Section 4 presents a solution for this service for IP networks using point-to-point connections. The final remarks give an outlook on next research steps.

2. REQUIREMENTS TO THE GROUP MANAGEMENT

The group management supervises the composition of the conference. It stores all information about the participants and handles the requests to join and leave the conference. The group management is closely related to the QoS management of the exchanged data streams. It is also connected with the applied floor control strategy.

The group management depends on the nature of the conference. A strongly coupled conference, e.g. a meeting, has other requirements to the group management than a loosely coupled one like a chat. A conference that requires access permission (invitation, authentication) needs another group management than a large public conference.

Another essential factor for the design of the group management is the kind of its organization. Most existing video conference systems use a centralized group server. The server controls the whole conference. Usually, the initiator of the conference provides this function. When he/she leaves the conference mostly has to be terminated. Centralized solutions are easy to implement because all management functions are concentrated in one system. This is especially advantageous if sophisticated control algorithms are applied, e.g. for scaling video streams to the actual load of the end systems. The shortages of the approach are as discussed above server failure and performance bottlenecks. The alternative decentralized approach avoids these problems. It is more flexible, especially in larger networks like the Internet, but it is more complicated to design. One of the main issues of the distributed approach is to preserve the consistency of the group and QoS data at each participants' host to ensure a unique view on the conference state for all decisions. It further requires mechanisms for a distributed calculation scheme for scaling the QoS parameters and for assigning the floor.

In order to ensure the consistency of the group and QoS data at the participants a distributed conference management has to support the following features:

- **Ordered delivery:** The messages are delivered in the order they are sent. This ensures that events like joining or leaving of a conference, or assigning the floor are indicated to all participants in the order they occur. Depending on the application different levels of ordering (causal order or total order) can be applied.
- **Reliability:** In contrast to the transmission of continuous data, where some frames may be lost, distorted, or discarded, signalization messages have to be delivered reliably.
- **Atomicity:** It must be ensured that the messages are either delivered to all participants or to none of them. This property ensures that all participants are equally updated.

Furthermore, it is desirable to have a dynamic group management which allows the participants to join and leave the conference at any time, and to exclude participants due to system failure. The protocol should be simple to limit the signalization overhead.

The mostly used transport protocol for multimedia data transmission in the Internet, the RTP/RTCP [14], does not fulfill these requirements (and it also was not its intention). The associated *Real time Transport Control Protocol* RTCP provides a feedback by periodically sending *Receiver Report* messages between the communication partners which inform about group membership and the quality of the data transmission. Thus RTCP only provides approximate information on the system and group state [1]. A

group member never exactly knows who is in the conference at a given moment. Moreover the leaving of the conference by a BYE message can cause error situations. When a participant has a system failure, it nevertheless remains registered in the session. This causes a critical situation when this partner rejoins the conference.

There are a lot of multicast protocol which support at least one of the above discussed features, e. g. RMP [9], SRM [7], RMTP [8], and others, but there are only a few protocols which support all three features: the Reliable Multicast Protocol RMP and the Totem Protocol [6]. Both protocols use a token mechanism and different ordering schemes. The protocols are relatively complex and assume that larger amounts of data are continuously exchanged. In small closed video conferences signalization messages are rather rarely exchanged, only when the speaker and the group composition change. Therefore a simpler protocol can be applied.

3. GCP_{ATM}

In the next two sections we present the signalization protocol GCP (*Group Communication Protocol*) that meets the requirements discussed in the previous section. GCP was first designed for the multiparty video conference system COVIS (*Cottbus Video conference System*) which followed from the experimental video conference system GCSVA (*Group Communication and Scalability in Video conferences over ATM*) [2]. COVIS was designed for ATM. Recently a new version of GCP for IP- based networks has been designed and implemented. It is deployed in the video conference system OCTOPUS [5] and in the IP-version of COVIS which is currently being implemented. In order to distinguish the two variants of GCP we denote them GCP_{ATM} and GCP_{IP} in the sequel. In this section we introduce the basic procedures of GCP and describe shortly how they can be provided over ATM. Details are given in [4].

3.1 The video conference system COVIS

COVIS was designed to support small closed discussion groups up to 15 participants. Typical applications are project meetings or problem discussions in which several persons collaborate to discuss a design or to jointly solve a problem. Other potential applications are teleseminars in which a tutor discusses problems with students at different remote places. The main features of COVIS are a strong floor control, a dynamic scalability of video streams according to the number and the performance of the

connected hosts, and a distributed group and QoS management. More detailed descriptions of COVIS are given in [2], [3].

3.2 Protocol principles

GCP is a signalization protocol for a distributed conference management. It is used to exchange management data among the participants' systems. GCP provides an ordered, reliable, and atomic data transmission service. It further supports a dynamic group management, i.e. the participants can join and leave the conference at any time. The joining of the conference is based on an explicit invitation of the participants. The invitation procedure is not part of GCP. This can be done, for instance, by using SIP [15]. The leaving of the conference can be either explicit by the participant, called *regular LEAVE* in the sequel, or forced by the system due to a participants' failure, called *forced LEAVE*. GCP runs in parallel to the video and audio transmission protocol.

GCP was originally designed for the application over ATM AAL5. ATM applies a sender related multicast paradigm that supports closed group conferences. The design of GCP_{ATM} is based on overlaid multicast connections between the conference participants at the ATM level. Overlaid multicast optimally supports the pursued distributed management concept in contrast to a centralized multicast server or to the mapping of the multicast connections on to single duplex connections which are often used for group communication over ATM [11], [12]. It directly uses the ATM multicast capability.

For the design of GCP, we studied existing multicast protocols and concepts. A detailed discussion of these approaches cannot be included here due to lack of space. A short summary was given in Section 2. In this section we focus on the basic principles of GCP needed for the discussion in the next section. We show how the features ordered delivery, reliability, and atomicity are achieved. More details are given in [4].

GCP is a token-based multicast protocol like [6] and [9]. The participants form a logical ring on which a token rotates. Only the token holder is allowed to send messages. The other hosts have to store the messages they intend to send until they get the token. All participants have to acknowledge the reception of the messages. Unacknowledged messages are repeated. After receiving all acknowledgments the token is forwarded to the next participant. The reception of the token must be also acknowledged. If a participant has no data to send the token is immediately forwarded. GCP assures the 3 features in the following way:

Ordered delivery is given by applying the rotating token mechanism that only allows the token holder to send messages. It provides a total ordering of

the message exchange. In the original design [4] the token holder could send only one message. This appeared to be inconvenient, especially for the exchange of messages about the conference state when new participants joined the group. Now several messages are allowed.

Reliability is achieved by the acknowledgment of all messages. Each message sent by the token holder must be confirmed by the participants. This concerns all protocol procedures: the join and leave of participants, the exchange of signalization messages, and the token forwarding. Figure 1 shows this for the exchange of signalization messages. Note that all

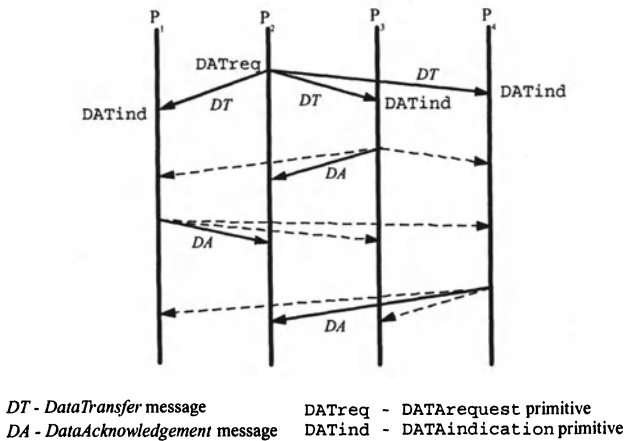
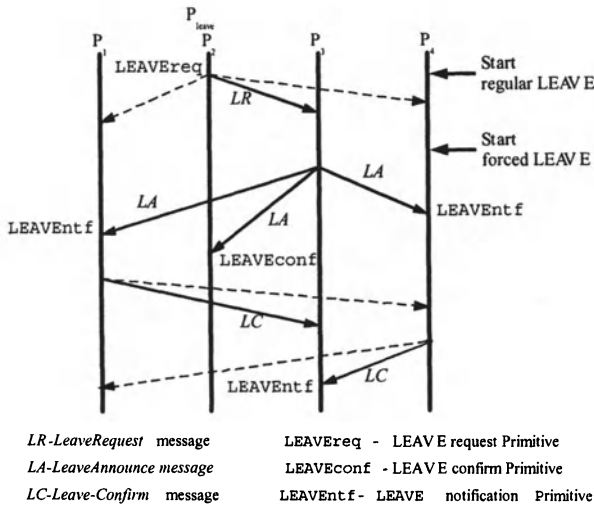


Figure 1. Data exchange in GCP_{ATM}

messages are multicast. The participants discard the received messages if they are not needed. If a participant does not acknowledge a message it is resent up to 3 times. After that the *forced LEAVE* procedure is started which excludes the participant from the conference. This procedure is discussed below. A checksum algorithm to detect distortions of messages is not contained. It is assumed that messages are not distorted in an ATM network, but they may be lost.

Atomicity is assured by the selective retransmission of messages in combination with the *forced LEAVE* procedure. The latter excludes participants from the group who do not confirm the reception of a message or of the token after 3 requests. Thus it is guaranteed that the remaining participants received the message. The *forced LEAVE* may look very severe and unusual, but in a video conference over ATM it is very probably that a missing acknowledgement of a signaling message sent several times indicates a grave error at the participant which may also affect the transmission of the audio and video streams. In such a case it seems more appropriate to exclude the participant so that he/she can explicitly rejoin the

Figure 2. GCP_{ATM} LEAVE procedure

conference instead of setting up a complicated error recovery procedure. Figure 2 depicts the time sequence diagram of the LEAVE procedure. The *forced LEAVE* begins with multicasting a *LeaveAnnounce* message (*LA*) to all participants which is indicated to their group management modules by a *LEAVEnotification* service primitive (*LEAVEentf*) and to the excluded participant by *LEAVEconfirm* primitive (*LEAVEconf*) (if this is still possible). All participants have to confirm the exclusion by sending a *LeaveConfirm* message (*LC*) to the token holder. The *forced LEAVE* procedure is part of the *regular LEAVE* procedure called when a participant intends to leave the conference. The latter differs from the *forced LEAVE* procedure by first multicasting a *LeaveRequest* message (*LR*). The rest is identical.

Similar procedures are applied for handling token loss and duplication. To avoid token losses and duplications a 3-way-handshake is used for the token forward to the next group member. If the *TokenForward* message (*TF*) is not confirmed the forward message is repeated up to 3 times and then the *forced LEAVE* procedure for the next token holder is triggered if there is no response of that participant. Thus the rotating token also helps to detect participants' failures even if no data are transmitted, because an outstanding acknowledgment for the token leads to the exclusion of the respective participant. Token duplication can only appear if the messages of the second and third handshake are lost. This is supervised by a timer at the new token holder. If a time-out occurs the old token holder is excluded by *forced LEAVE*.

A prototype of COVIS was implemented on an experimental ATM network consisting of a FORE ASX200WG switch and five SUN workstations of different performance¹, each with an ATM interface card SBA200. In order to evaluate the performance and the applicability of GCP_{ATM} the token round trip time was measured. The measurements were carried out for system configurations with two, three, four and five workstations. In the experiments no data were sent, only the token rotated. The results are contained in Table 1. The token round trip time was too short. This led to a strong overload for the network and the involved workstations because they had too often to handle the token. Therefore, an additional delay of 10 ms was introduced in the COVIS prototype implementation (see Table 1). We extrapolated the measurements without delay to estimate the token round trip time for larger groups. It is about 40 msec for a group of 10 participants. The measurement results show that the GCP approach is applicable to small group video conferences.

Table 1 Token round trip time in GCP_{ATM}

Number of participants	Token round trip time [msec]	Token round trip time with delay [msec]
2	6,749	36,060
3	11,193	54,383
4	14,946	75,038
5	25,974	95,581

4. GCP_{IP}

The original GCP cannot be applied directly for applications in the Internet, because it assumes a sender related overlaid multicast communication infrastructure in the underlying protocol and not a point-to-point communication. IP multicast, however, uses a receiver related multicast paradigm which is not suited for closed groups. Protocols which are based on this paradigm like RMTP [8] and RMP [9] cannot be used to replace GCP although they in part support the required properties. A mapping of GCP on point-to-point connections is not acceptable, because it increases the group management and the network load for larger groups. The Small Group Multicast (SGM) protocol [10] might be a solution. It supports small closed multicast groups with a sender related multicast. The reliability of the transmission has to be assured by higher layer protocols. A shortage is that in the last segment between router and end systems it uses unicast

¹ 3 UltraSPARC1 (170 MHz, twice 140 MHz), 1 HyperSPARC (125 MHz), and 1 MicroSPARC (70 MHz).

transmission. This might be disadvantageous if there are several group members in this subnetwork. In addition, SGM requires a specialized routing software which must be available everywhere.

A software solution which can be installed in the end systems is more appropriate for this purpose. Therefore, we redesigned GCP for IP. It is applied in the video conference system OCTOPUS and for the new COVIS version over IP.

4.1 The video conference system OCTOPUS

OCTOPUS is a scalable global multiparty video conference system which was developed in a joint project between the Hong Kong University of Science and Technology and the Brandenburg University of Technology at Cottbus [5]. The objective of OCTOPUS is to set up a global video conference by connecting a number of local video conferences at different places over long distances via the Internet or ATM networks (see Figure 3). The links between the local conferences are point-to-point connections. The

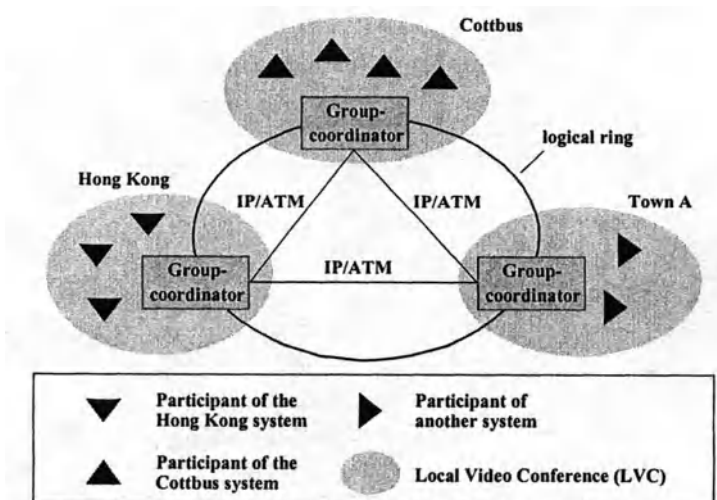


Figure 3. Architecture of the OCTOPUS videoconference system (global view)

global conference is again a closed group conference. It possesses a joint floor control and a global dynamic group management. OCTOPUS does not make any assumption on the architecture of the local videoconference systems. Similarly to the principles of open systems it aims at connecting video conference systems of different structure and technology.

A unified interface to the global conference is defined by the group coordinator. It covers the possibly various local system architectures and takes over the management of the conference group. The activities comprise, among others, the group composition, the floor control, the QoS tuning, and

the delivery of the video and audio streams between the local conference systems. For example, a join or leave of a (local) participant is indicated to all participants. The local floor control strategies are replaced by a global strategy.

For the communication between the group coordinators, the GCP_{IP} is applied. RTP [14] is used to transmit the audio and video streams within the closed group. Unlike COVIS the transmission of the audio and video streams between the group coordinators is point-to-point. Not all video streams are transmitted to the remote groups. The selection depends on the decision which participants are shown at each screen and on the available bandwidth. For the selection, a voting procedure was introduced [5], [13]. The conference scenarios and the operation modes of OCTOPUS are described in [5].

4.2 Protocol principles

GCP_{IP} is a signalization protocol for systems which communicate via UDP/IP. It provides an ordered, reliable, and atomic sender related multicast service. Furthermore, it supports a dynamic group structure. The participants form a logical ring (see Figure 5Error! Reference source not found.). In contrast to GCP_{ATM} each message in GCP_{IP} are simultaneously sent in both directions: clockwise and anticlockwise. When a participant receives both messages it can be assumed that the ring is not broken and that the other group members also received these messages. The reception of the message has to be confirmed the same way. In case of a broken ring the token holder repeats the message up to 3 times. To determine the gap in the ring the third message must be confirmed by all participants passed. Then the *forced LEAVE* procedure is started to exclude the nonactive participant. The ring is set up during the initialization phase. A new participant is included into the ring in such a manner that his/her neighbours are situated, if possible, in the same network segment. This avoids long token rotation times.

GCP_{IP} assures the three basic features of GCP as follows. **Ordered delivery** is forced by the token mechanism which only allows the token holder to send signalization messages. The other participants only confirm the reception of the messages. **Reliability** is achieved by the double logical ring which connects the group members. It guarantees that all messages are delivered to all active group members. **Atomicity** is ensured by the *forced LEAVE* mechanism which as in GCP_{ATM} excludes nonactive group members to reestablish the ring.

In the sequel we describe the protocol phases of GCP_{IP} in more detail.

SETTING UP A CONFERENCE GROUP

When a conference group is supposed to set up the initiator first establishes a connection with one conference partner using a 3-way-handshake. The initiator (in the OCTOPUS system this is the group coordinator) delivers a *JOINreq* primitive to its GCP_{IP} entity which sends a *JoinRequest* message (*JR*) to the selected partner. There the request is indicated by a *JOINind* primitive. The invited partner can accept or reject the invitation with a *JOINresp* or a *JOINrej* primitive, respectively. The answer is sent back to the initiator by a *JoinConfirm* (*JC*) or *JoinDisconnect* (*JD*) message. When the session is rejected the initiator is informed by a *JOINdis* primitive. The acceptance of the invitation is confirmed by a *ConnectAddition* message (*CA*) by the GCP_{IP} entity. Both participants are

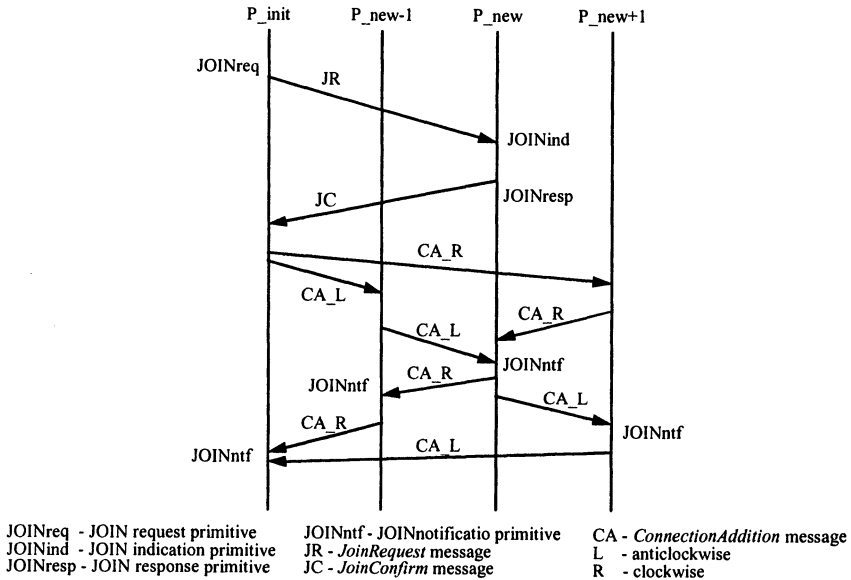


Figure 4. Invitation of a new member

informed about the successful connection by a *JOINntf* primitive. Then the initiator creates a token and forwards it to the partner. Further group members are invited as new participants as described next.

INVITATION OF A NEW PARTICIPANT

Each group member is allowed to invite a new participant upon an oral agreement with the other conference partners. The invitation is triggered by a *JOINreq* and follows the same procedure as the conference initialization (see Figure 4). After receiving the confirmation the inviting group member determines between which partners the newcomer is included into the ring to optimize the token round trip time. The group members are informed by a *ConnectionAddition* (*CA*) message about the position of the new participant.

Each group member proves whether the new partner is its neighbour. In this case they update their forward address and forward the *CA* message to the new partner. The *CA* also contains state information of the ring for the new partner. The new partner forwards the message to its new neighbour. The same procedure is executed for the other direction. In Figure 4 we distinguish the *CA* messages sent in opposite directions by adding *L* and *R* for left and right, respectively. The successful join of the new group member is indicated by a *JOINntf* to the application.

DATA TRANSMISSION PHASE

The data transmission in GCP_{IP} corresponds to that of GCP_{ATM} as described in Section 3.2. The difference is that the *DT* messages are also sent in both directions of the logical ring. Figure 5 represents the principle. The Figures 5a) and 5b) show the forwarding in both directions. In Figure 5c) the group members received the *DT* message and indicate this to their application by a *DATind* primitive. The successful transmission is implicitly

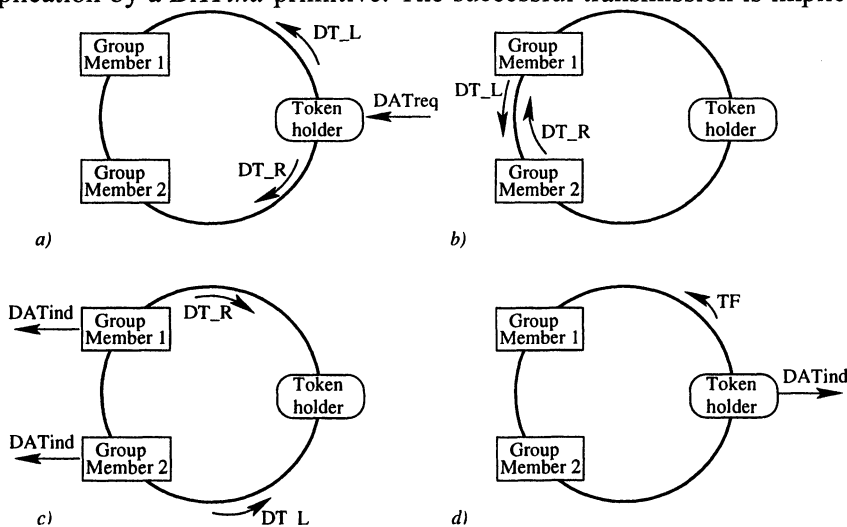


Figure 5. Data Transmission of GCP_{IP}

confirmed when the both messages pass every participant. An unsuccessful transmission is repeated up to 3 times. After that the *forced LEAVE* procedure is called. In Figure 5d) the token is forwarded to the next group member.

LEAVING THE CONFERENCE

GCP_{IP} applies the *regular* and the *forced LEAVE* procedures as GCP_{ATM} . Figure 6 depicts the time sequence diagram for this procedure. In case of a *regular LEAVE* the leaving group member informs their neighbours by an *LR* message. The neighbours confirm this by a *LC* message and change the

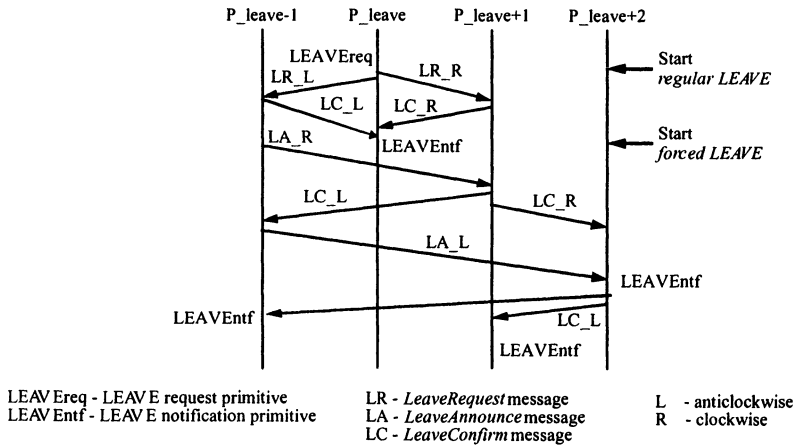


Figure 6. Leaving of a participant

addresses of their respective neighbours. The leaving member indicates the leave to the application by a *LEAVEntf*. The rest of the procedure concerns only the remaining group members. This is also the starting point for the *forced LEAVE* procedure. The both neighbours of the left or excluded member inform all group members by sending *LA* messages in both directions. This has to be confirmed by an *LC* message. After having received both *LC* messages each group member sends a *LEAVEntf* primitive to its application to indicate the leave. If there remains only one group member the conference is terminated.

4.3 Performance evaluation

To evaluate the performance of the protocol we measured the round trip time of the token and the average delay of data packets between sending and delivery. The round trip time indicates how fast requests of the application to the group management are handled. The delay is a measure for the distribution time of the signalization messages. The sum of both times is the maximum reaction time of the system to group events. The assumption of a simple addition can be made here, because the frequency of the data exchange is relatively low. The measurement results are depicted in Figure 7.

The measurements were carried out on a 100 Mbps Ethernet with an average daily load. The involved hosts were SUN UltraSparc 10, SUN SparcStation 20, and SUN SpacStation LX workstations. As expected the round trip time and the delay increased quasi linear. The mostly slightly higher delay of the data packets can be explained by the double handling of the packets for both directions. For 8 participants, the maximum reaction time is about 210 msec and approximately for the planned upper size of 15

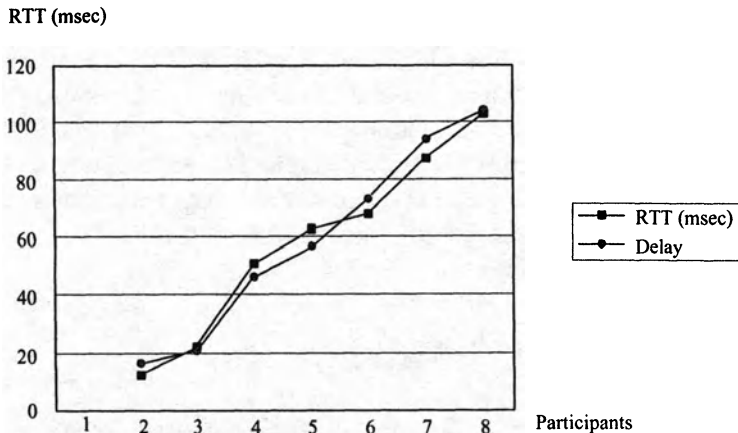


Figure 7. Token Round-Trip-Time and average packet delay

participants about 360 msec. The estimated lost of bandwidth due to the rotating token mechanism is approximately 3 per cent of the average bandwidth needed for a video conference with 15 participants. The measurements prove that GCP_{IP} can be deployed for the planned purpose, since the delay of events such as speaker change or participant's join and leave are very low. For small groups, GCP_{IP} represents an alternative to other protocols in the Internet.

For applications in global networks like the OCTOPUS project, the transmission delay has to be taken into account and added to the above measured values. The round trip time for transmissions between Cottbus and Hong Kong fluctuate between 350 and 500 msec. This is still a remarkable overhead. Closed group video conference meetings in their majority, however, will take place in much smaller regions (e.g. university campuses, cities, states, countries). GCP_{IP} offers an applicable solution for this.

5. FINAL REMARKS

In the paper we have presented the group communication protocol GCP for supporting a distributed management of closed video conferences. GCP aims at small closed groups. It provides an ordered, reliable, and atomic data delivery services which ensures the consistency of the management data at the participants. GCP exists in two variants. GCP_{ATM} has been successfully applied for video conferences over ATM. In this paper we have shown with how the features of GCP can be also provided in IP networks. The measurements show that GCP_{IP} is applicable for supporting closed groups meetings in the Internet. GCP can also be applied to other areas like joint editing which require a floor control.

In our current research we deploy both GCP variants. We are reimplementing COVIS on the German scientific research network G-WiN that provides high bandwidth up to 2,5 Gbps without guaranteed QoS. This requires changes in the COVIS_{IP} architecture which affect beside the video stream scaling also the group communication. The architecture of COVIS_{IP} is similar to that of OCTOPUS. It replaces the group coordinators by video gateways. GCP_{IP} is used for the group management.

REFERENCES

- [1] Handley, M.; Crowcroft, J.; Bormann, C.; Ott, J.: Very large conferences on the Internet: the Internet multimedia conferencing architecture. *Computer Networks* 31 (1999)
- [2] Beier, I.; Koenig, H.: GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management. *Proc. of the 7th International IEEE Conference on Computer Communications and Networks (IC³N'98)*, Lafayette, USA, 1998
- [3] Beier, I.; Koenig, H.: Distributed QoS Management for a Multiparty Videoconferencing System. In: Yongchareon, T.; Agesen, F. A.; Wuwongse, V. (eds.): *Intelligence in Networks. IFIP TC6 WG6.7 5th International Conference, SMARTNET'99*, Pathumthani, Thailand, Kluwer Academic Publishers, 1999
- [4] Beier, I.; Koenig, H.: A Protocol Supporting Distributed Group and QoS Management. In: *Proc. IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking (PROMS-MmNet'97)*. Santiago, Chile, 1997
- [5] Chanson, S. T.; Hui, A.; Siu, E.; Beier, I.; Koenig, H.; Zuehlke, M.: OCTOPUS - A Scalable Global Multiparty Video Conferencing System. In: Dixit, S.; Somani, A.; Park, E. K. (eds.): *Proc. of the 8th International Conference on Computer Communications and Networks (IC³N'99)*, Boston, Massachusetts, USA, IEEE Operations Center, 1999
- [6] Agarwal, D.A.: Totem: A Reliable Ordered Delivery Protocol for Interconnected Local-Area Networks, Ph.D Thesis, University of Santa Barbara, Dec 1994
- [7] Floyd, S.; Jacobsen, V.; McCanne, S.; Liu, C.-G.; Zhang, L.: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, *Proceedings of the ACM SIGCOMM'95*, Cambridge, USA
- [8] Lin, J.C.; Paul, S.: RMTP: A Reliable Multicast Transport Protocol, *IEEE INFOCOM'96*, San Francisco, March 1996
- [9] Whetten, B.; Montgomery, T.; Kaplan, S.: A High Performance Totally Ordered Multicast Protocol, *IEEE INFOCOM'95*
- [10] Boivie, R.; Feldmann, N.: SGM - Small Group Multicast. Internet Draft Proposal, July 2000

- [11] Chan, D. L. S.; Chanson S. T.: Scalability Support for Multiparty Multimedia Communications. In: ACM/Springer Multimedia Systems Journal, 6 (1998) 2
- [12] Gauthier, E.; Le Boudec, J.-Y.; Oechslin, P.: SMART: A Many-to-Many Multicast Protocol for ATM, IEEE Journal on Selected Areas in Communications, 15 (1997) 3
- [13] Koch, H.: A solution of the visibility problem for the video conference system OCTOPUS. Master thesis, BTU Cottbus, Department of Computer Science, 2000.(in German)
- [14] Schulzrinne H.; Casner S.; Frederick R.; Jacobson V.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889, 1996
- [15] Handley, M.; Schulzrinne, H.; Schoolre, E.; Rosenberg, J.: SIP: Session Invitation Protocol. RFC 2543, March 1999

III

PROTOCOLS

Developing MLS Location-Based Service Pilot System

Jouni Markkula, Artem Katasonov, and Artem Garmash

Information Technology Research Institute, University of Jyväskylä

P.O. Box 35, FIN-40351 Jyväskylä, Finland

E-mail: markkula@jyu.fi, artem.katasonov@titu.jyu.fi, artyom.garmash@titu.jyu.fi

Abstract Development of mobile networks and terminals provides basis for advanced mobile services and applications. One of the new features is support for locating the mobile user by her/his terminal. The Location-Based Services (LBS), utilising this feature, are expected to form an important class of the new services. At the Information Technology Research Institute, University of Jyväskylä, LBSs have been studied in MultiMeetMobile research project. The project developed an LBS pilot system, called MLS. This paper presents the experiences collected during the development of the LBS pilot and describes the MLS system.

Keywords: Location-based services, mobile computing, XML, Java

1. INTRODUCTION

In the past few years, the mobile networks and terminals have been under intensive development. Voice capabilities have been augmented with data capabilities of increasing speed and other new features have been combined to the networks and terminals. One of these features is location support. The mobile user can be located by her/his mobile terminal, and the location can be transmitted to third parties offering mobile Location-Based Services (LBS). According the forecasts, the LBSs will have over 300 million users by the end of the year 2004 (Singhal et al. 2001).

At the Information Technology Research Institute, University of Jyväskylä, the research on LBS was started in summer 1999 in MultiMeetMobile project. The MultiMeetMobile project produced also a

pilot system called MultiMeetMobile Location-based service System (MLS). MLS is a research prototype for studying LBS in emerging location-aware mobile environment.

The development of MLS was started in autumn 1999. At that time, the mobile data services for general public were practically non-existent. The WAP environment development was in its first stage; no WAP terminals were publicly available. Location Services (LCS) and positioning methods were specified for public mobile networks (GSM), but they were not yet implemented by any operator. After a while, the first map services on WAP were introduced. However, they were not very useful, because of the limitations of WAP environment. It took about two years, until summer 2001, before the first LBSs using network based positioning (cell area) were publicly available and the first "navigation phone", Benefon Esc! (Benefon), using GPS positioning, was on the markets.

LBSs are still in the early stage. The services in Europe are rather simple and mostly relying on text based (SMS or WAP) output. The services and applications utilising maps in mobile environment are using server centric approach and bitmap graphics for transmitting the geographic information to the mobile terminals. For example Benefon Esc! requires the user to download the (expectedly) needed bitmap maps beforehand to the phone through PC computer. The purpose of MLS was to concentrate on problems and features of more advanced and usable mobile LBSs that will be expectedly available within a few years time.

This purpose of this paper is to present some key features of LBS system development on the basis of the collected experiences in MultiMeetMobile project's research and development work. It presents the framework of the development, design principles and features of MLS, and describes the current stage of the system.

2. FRAMEWORK OF MLS DEVELOPMENT

MLS was aimed to be a general LBS system for supporting location-based information access in wireless client-server environment. In the development, the properties and restrictions of distributed mobile computing environment were taken into account.

Mobile computing environment is formed of mobile network infrastructure and mobile terminals. The targeted network infrastructure is public land mobile network, supporting data transmission and locating of the mobile terminals. In Europe, it is current GSM data, HSCD, GPRS, and emerging EDGE and UMTS networks. The targeted terminals are current and evolving personal pocketsize mobile terminals, combined with mobile

network access. The expectation is that the terminals will have increasing computing power and reasonable input/output features. Advanced LBSs with map display require new generation of communicators and smartphones with high-resolution (120x120 or more) colour displays.

Mobile computing environment has certain features that impose restrictions (see for example Garmash, 2000). The properties of mobile networks are: (relatively) low bandwidth, high bandwidth variability, long latency, unpredictable disconnections and communication autonomy. The properties of mobile terminals are: small and low-resolution displays, limited input capabilities, limited computing power, limited power, small memory size, vulnerability, varying design forms.

On the other hand, the advantage of mobile computing environment is that it allows the user to access and use information in any place at any time, especially en route, on the move. The LBSs and other mobile services are particularly useful for conducting suddenly emerging “here and now” tasks which are urgent, well specified, restricted and tied to the present location and context.

The practical conditions, when and where the mobile devices are used, bring also additional restrictions. These restrictions have to be taken very carefully into account when designing usable mobile services. The using conditions cannot be expected to be constant, as usually is the case in “desktop” conditions. The mobile users are typically in very unstable environment in varying conditions, where their cognitive capacity is demanded for other tasks too. For that reason, the applications and service access should be very simple and user friendly and the amount of presented information content limited and well specified.

The content of LBSs can be divided into two components: digital map data and location-based information. The distinction is essential, because the nature and application of these data types is different. Typically they also come from different sources, possessed by different actors in the service architecture.

Digital map data is the geographic base data that form the geographic data infrastructure of LBSs. It is used by most of the services. Digital map data consists of general geographic features that are used for locational reference and spatial analysis; such as road networks and administrative areas. This base data is relatively stable in time and it is often maintained by public sector, e.g., municipalities or national mapping agencies. For example in Finland the National Land Survey is responsible of official mapping and Finnish Road Administration is preparing national road and street network database, DIGIROAD (Liikenne ja viestintäministeriö, 2001) for new location-based and mobile services. The digital map data used in LBSs are based mostly on the existing GIS databases.

In contrast to geographic base data, location-based information forms the value-added content of LBSs. Location-based information consists of any data that can be georeferenced, associated to location; for example location of the restaurants or weather forecast in a particular area. It is the content that various LBS providers are offering to mobile customers, often combined with the digital map data. Location-based information can be relatively static (e.g. the restaurants) or it can be rather dynamic (e.g. the weather). This data from different content providers can be either stored in GIS databases or dynamically created and geocoded by the service.

3. DESIGN PRINCIPLES AND FEATURES OF MLS

MLS was developed as a research prototype for general LBS system that is using maps for information presentation and access at the terminal. The approach was to take into account the properties and restrictions of mobile computing environment from the beginning of the development.

The starting point was a view of an advanced and usable LBS system supporting navigation and location-based information access operating on the target environment outlined in the previous section. The LBS system was considered to be utilising the two types of content, digital map data and location-based information. The information content can be combined from different heterogeneous data sources. In addition, the LBS system should support some analysis functions, such as spatial and non-spatial queries and route finding. In that sense a LBS can be considered as a specialised Geographic Information System, GIS for a mobile use (Burrough and McDonnell, 1998).

The MLS system was designed to be “GIS supported”, utilising existing geographic databases and GIS methods. The relationship between LBS and GIS and possibilities of GIS to support development of advanced LBS systems were studied in MultiMeetMobile project (Virrantaus et al. 2001).

The digital map data for MLS was provided by the National Land Survey of Finland. In MLS the data is stored in Oracle database in geographic vector data format. MLS utilises also methods of geographical data analysis and presentation developed in GIS. The special features of MLS are presented below.

Geographic vector data. The MLS system transmits geographical data from the server to the client in a geographic vector data format. In this the MLS deviates from most of the existing LBSs, which relay on bitmap graphics. The vector geographic data allows modification of the data presentation. But, most significantly, it makes possible to move some computation from the server to the client. It allows, for example, utilisation

of analysis functions on the client side, independently and without straining the server. Some of the advantages are the following:

- The vector model explicitly describes geographic features. This allows linking spatial data and attributes. Therefore, a map can be accompanied with various information associated to objects.
- In the vector model, the explicit description of the road network topology is possible. This allows implementation of route finding algorithms.
- Vector data enables the use of advanced, real-time generalisation routines.
- Vector data enables the user to customise a map by defining colours and symbols as well as excluding some data themes.
- Vector data, in many cases, requires less storage space than data in the raster form.

Delegation of computation to the client. MLS design makes possible to delegate some of the computation within the system to the client. For example, various queries, such as route finding or searching of the nearest points of interest, could be done in the client. Processing power of the new mobile terminals usually allow processing of limited volumes of data in a reasonable time. Frequent queries from the client also cause extensive use of mobile network, and possibly a greater latency because of the limited bandwidth of the network.

Extensible Markup Language (XML). The MLS system uses XML encoding for geographic data. The MLS uses two types of XML-based geographic data formats: the standard Geography Markup Language (GML), specified by OpenGIS Consortium (OpenGIS, 2001); and Geographical Mobile Markup Language (GMML), developed in the MultiMeetMobile project (Garmash, 2000). There are some reasons for using self-developed GMML. Initially, the standard GML specifications were not publicly available when GMML was developed. The MLS also prefers more lightweight data format for transmitting and using within the client. In addition, GML has certain limitations; it does not support explicitly network topology information. Network topology is necessary for many typical navigation applications, such as route finding. Currently the most recent GML version 2.0 enables implicit modelling of topology by the XLink mechanism. Better tools for topology are also promised in the future.

Moreover, MLS system utilises XML as the core technology for data representation. XML-based data formats can be used in interaction with mobile devices and content servers. XML-based approach offers the following advantages:

- Standard tools, such as XML parsers and XSLT processors, can be used for extracting and transforming data.
- XML allows data integration from heterogeneous sources. Data from different geographic data and information content providers can be integrated into one LBS system.
- XML facilitates interoperability between LBS components.
- Different data formats for different target platforms can be used.

Java. The client application of the MLS system is implemented in Java language. Support for Java technology is an important feature of modern mobile devices. Currently, all the Symbian EPOC and Microsoft Windows based pocketsize computers support Java. In Japan the new mobile phones are able to run Java programs. Java support is included also in a version of Symbian EPOC 6 operating system, Pearl, aimed for next generation smartphones. Currently, Java support is expected to be a standard feature of practically all of the future mobile terminals. Java language presents a powerful and flexible tool for mobile services development because of its platform-independence. Client-side software developed in Java can be run in nearly every device supporting this technology, thus eliminating the need for development of different versions for every target platform. The MLS client application is lightweight, and after rewriting the user interface, it could be run even under Java 2 Micro Edition (J2ME) used in smartphones.

Intelligent data selection. MLS system is capable of selecting geographic data that is relevant for a specific user at a specific time. It uses a user-centred approach. The location of the user and his preferences are taken into account. The MLS client allows the user to specify a set of values defining the preferred scale of map and the relative importance of data themes. The MLS server then uses this information together with data on the location (as well as the direction and the speed of movement, if available) of the user for calculating importance of every geographic feature. The importance of a feature then defines if it will be represented, and, if so, the level of detail LOD this feature will have (some attributes can be excluded and the geometric form simplified). The total data volume is kept constant. Progressive data transmission is also supported. This means that if for a geographic feature $LOD2 < LOD1$ then the data set representing this feature at LOD2 is a subset of the data set at LOD1. No new data is introduced during generalisation, what can occur if, for example, smoothing of the geometric form or generalisation of attributes values are used. Therefore, increasing the level of detail for a feature, we have only to transmit additional data, no retransmission of already sent data or deletion of any data is needed. The MLS server stores the current LOD value for every

geographic feature thus keeping track of data that user has currently available in the memory of his mobile terminal.

The objectives of such an approach are the following:

- Increase of the service usability, while only relevant data is transmitted to the user and unnecessary is filtered out.
- Reduction of the volume of transmitted data.

Details of the used computational algorithm can be found in Katasonov (2001).

Transaction management support. In order to protect itself from the unexpected disconnections and crashes, the MLS client application keeps track of the history of interaction with the server (what data was sent and received) and stores all the data needed for recovery. This saves the application from information losses and guarantees basic transactional properties.

4. RELATED WORK

Currently, the market provides a set of mobile mapping applications and services. Some of them are born in the GIS field and represent simplified GIS systems running on mobile devices, like ESRI's ArcPad. Some others represent rather navigational LBSs providing the user with maps and possibly other data, like Webraska system (Mitchell, 2001) accessible with a WWW or a WAP browser.

While existing systems rely primarily on bitmaps or some application-specific formats, XML is positioned currently to be a standard way for encoding geographical data in future, including data transmitted to LBS clients. OpenGIS Consortium developed GML language (OpenGIS, 2001) that is going to be probably a standard. LaserScan developed a GML-compatible Spatial Object Transfer Format (SOTF) (Mims et al., 2001) supporting incremental updates of data and topology information. XML-related work is going on in Finnish Geodetic Institute. In Lehto (2000), the applicability of XML and related technologies for geospatial applications is argued: using XSLT for transforming geographical data, XML Schema for spatial data modeling, XLink for description of spatial relationships. In Lehto and Kilpelainen (2001), an approach and a prototype system for real-time generalization of XML-encoded geographical data using XSLT are presented. Location Interoperability Forum (LIF) is developing currently a standard interface to location services (LCS) that is also XML-based.

Active research is going on concerning various aspects of mobile GIS. TeleCartography project of the Department of Cartography in Vienna University of Technology studies LBSs and impact of new technologies like

UMTS on them (Gartner et al., 2001). Research work in LaserScan is related to on-demand generalisation of geographical data before transmitting to a mobile device; approach is based on the active-objects technology (Hardy et al., 2001). Research in University of California in Santa Barbara considers questions related to human-computer interaction in mobile GIS, usability of their user interfaces and use of augment reality technology (Clarke, 2001). Reichenbacher (2001) presents a general view on mobile cartography, which emphasises a need for transmitting data in the vector form, adaptive map rendering depending on the context and preferences of the user.

5. DESCRIPTION OF MLS AT THE CURRENT STAGE

5.1 MLS architecture

MLS is a general LBS system for mobile users offering a map and navigation service accompanied with access to location-based information. Architecture of the MLS system (figure 1) follows the general LBS architecture and includes the following main components:

- ☐ Mobile terminal. Used for accessing the service and running the MLS client application.
- ☐ Mobile network. Used for communication between mobile devices and the MLS server.
- ☐ Location service (LCS). Provides information on the location of the mobile users. The dotted arrows between the mobile device and the LCS in figure 1 indicate that the mobile terminal could interact with an LCS directly and then send retrieved data to the MLS server in parameters of a query.
- ☐ MLS server. Runs the MLS server application.
- ☐ Content servers. Offer information or other content that is used by the service and possibly transmitted to mobile users.

In MLS, the two types of content, digital map data and location-based information, are differentiated, as seen in figure 1. In the MLS system, the two content data sets are used. The first data set consists of the street network and the second data set consists of points of interest information. As we stated in the previous section, XML-based interaction with content servers is to be used. However, it is not implemented on the current stage. The knowledge about the structure of the content databases, as well as

about the DTD and the meaning of the tags of the output GML format, is embedded into the server application code. The MLS server retrieves and handles data on the base of this knowledge. Oracle database (with Oracle Spatial) is used for storing the data.

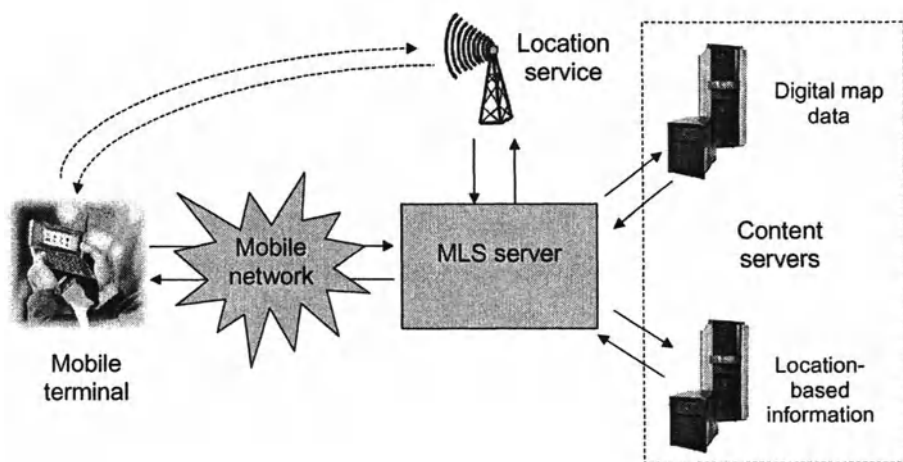


Figure 1. MLS Architecture

5.2 The MLS server

The server of the MLS system is an application accessible through a TCP/IP network. It is implemented in Java. Client-server interaction is implemented on the top of HTTP protocol. The MLS server supports authentication of users and three types of requests: request for a list of available data themes, request for data, and request for the location of the user (data is retransmitted from an LCS). Server response is encoded using GML format. When handling a request for data, the location of the user and the parameters specified by him are taken into account for selection of relevant data. GML data is transmitted in a compressed form using standard GZIP algorithm. This reduces the volume of the transmitted data about ten times. Use of algorithms developed specially for compression of XML data have also been studied.

The MLS server supports authentication of the users. For an authenticated user, a working session is opened. The following consecution of requests from the user is considered as belonging to this session. The session can be closed either by user's request or automatically, if there is no requests from the user for a certain time.

5.3 The MLS client

The MLS client application is responsible for data representation and for the execution of the queries. The queries can be, for example, search for the nearest point of interest of a specified type. The client is implemented in Java. It can be run in nearly every device supporting Java technology. However, modifications in the user interface would still be necessary, for example, for porting it to Java 2 Micro Edition (J2ME) used in smartphones. For this reason, the architecture of the client application was designed to support further porting to other platforms.

The architecture of the MLS client uses Model/View/Controller (MVC) user interface paradigm (Krasner and Pope, 1988), in order to separate the logic of the application from the code controlling presentation.

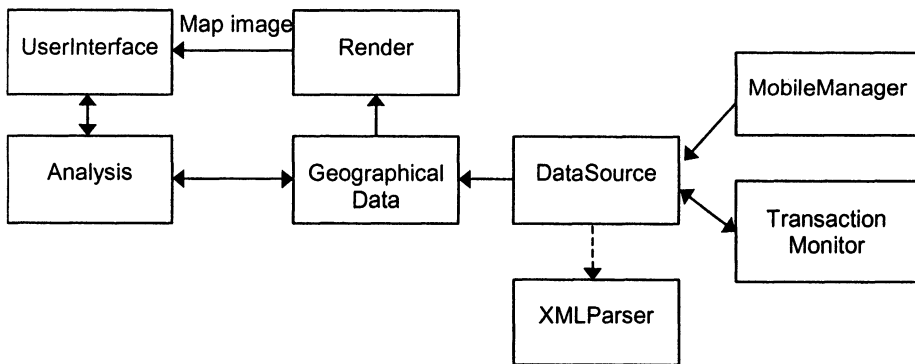


Figure 2. Architecture of the MLS client

Figure 2 depicts the main components of the MLS client application. The DataSource component interacts with the server in order to retrieve data and uses the XMLParser to process it. Based on obtained data, an internal model is constructed. This model is stored by the GeographicalData component. A Render component prepares a map image that is presented to the user. The map image is displayed by the UserInterface subsystem. The GeographicalData component represents the model. A Render and a UserInterface together form a representation. The UserInterface is responsible also for the user input and modification of the representation. Modification of the model is performed by the DataSource component. MobileManager component is responsible for updating the positions of mobile objects. However, the user is the only mobile object in MLS at the current stage. TransactionMonitor component is responsible for management of mobile transactions. Therefore, the model was maximally separated from

platform-dependent components. For a new platform, the UserInterface component has to be mostly rewritten. In addition, some modifications probably have to be made to the DataSource component related to interaction with the server through a network (at least it is needed for porting to J2ME).

The MLS client application uses third-party libraries for handling XML. Microstar XML parser implementing SAX interface and kXML API are currently used. kXML is a lightweight XML API optimised for the Java 2 Micro Edition. kXML was originally developed at the University of Dortmund, now evolving further as a part of Enhydra project (kXML).

The following series of figures presents the appearance of the MLS client application on the Symbian EPOC system and its functionality at the current stage. The user starts with connecting. First, the location of the mobile device is requested from the LCS. Then a transaction with the MLS server begins. User is requested to authenticate himself. If authentication is passed, the client application requests from the MLS server a list of available data layers and presents it to the user (figure 3).

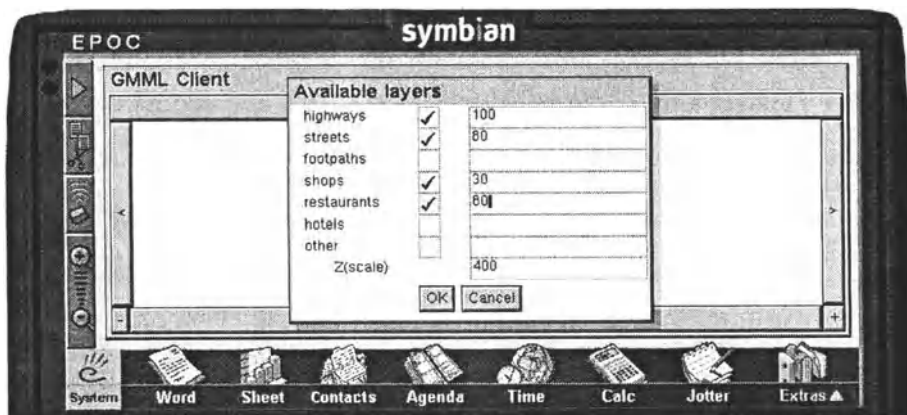


Figure 3. List of available data layers

After the user has selected the needed layers, defined the relative importance of them, and the preferred level of detail, the request for the data is made to the server. After retrieving data from the MLS server, it is presented to the user in the form of a map (figure 4). The current location of the user is also represented on the map and continuously updated. The interface supports basic operations like panning (long buttons on the borders of the screen) and zooming (small buttons in the down left and down right corners). The presentation of the map can be customised. The user can select colours and symbols to be used for representation. Some data layers can be excluded from the map.

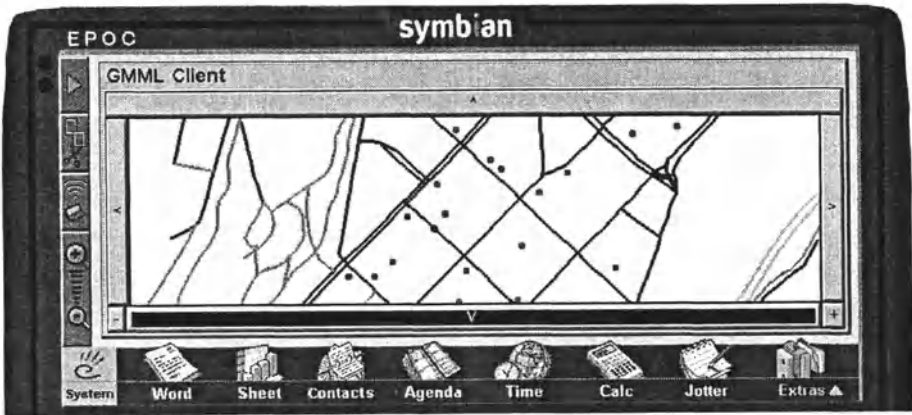


Figure 4. Map view

The user can retrieve information about a specific point of interest by pointing on the map (figure 5). As can be seen from the figure 5, the MLS client uses a separate view for point of interest presentation. Reason for this is, that the screen of a mobile terminal is usually too small to divide it into frames. User can also execute query for the nearest point of interest of a specified type (on the current stage, this query uses simple Euclid metric for selection of the nearest object).

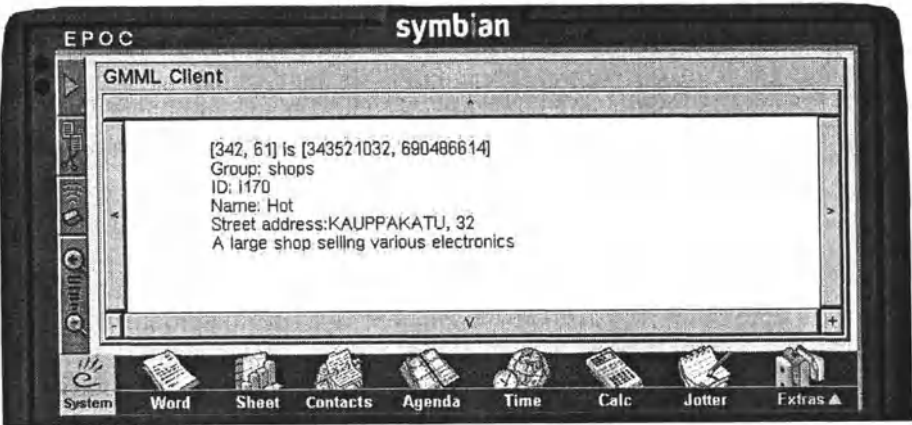


Figure 5. Point of interest description

Transaction Monitor component of the MLS client protects it from the unexpected disconnections and data losses. If a crash occurs, in most of cases the user is able to continue after restarting the application from the point where he was before the crash. When restoring a work session, the following routine is used. If the map data has already been received during

this session, then the map is presented to the user. If only information about data layers is already available, then the application requests the present location of the user from the LCS and reconnects to the MLS server. Then the list of the layers (figure 3) is displayed then to the user. If no substantial data has been received (layers data or map data), then the session is closed.

6. FUTURE WORK

The MLS pilot system is still under development. The implemented basic features are to be complemented with more advanced features and methods. Some new solutions will also be implemented and tested. This is related to both, the client application and the architecture of the system in general.

In relation to the MLS client application, the future work is directed to the following tasks:

- Improving the functionality and increasing the usability of the user interface.
- Implementing analysis functions. Currently, the only function working is search for the nearest point of interest of a specified type, based on simple Euclid distance. A more advanced search algorithm, taking into account the road network, could be implemented. The search could also be made based on other parameters specified by the user (search by name, by value of an attribute etc.). A route finding algorithm should also be added.
- Development of a version of the MLS client for J2ME. It could be interesting to test the application on a smartphone. When developing the client application, it was designed in a way that allows portability to other platforms, attempted to make it also as lightweight as possible. However, changes are needed at least to the user interface in order to port it to Java 2 Micro Edition.
- Further development of Transaction Monitor. On the current stage, TM is responsible primarily for data integrity on the client side. However, more research has to be done on questions related to the need to rollback an uncompleted transaction conducted via a mobile network and other related problems.
- Moving to the standard Geography Markup Language (GML). Development of GML is still going on. Therefore, this direction of future work strongly depends on the standardisation work held in the OpenGIS Consortium.

In relation to the server application of the MLS system, the main direction of the future work is further development of the data selection

algorithm and implementation of progressive data transmission. The selection algorithm supports progressive transmission. However, there are open questions about removing data from a mobile terminal memory when the level of detail of a geographic feature is reduced (e.g. user moves out of the feature). A discussion on this topic can be found in Katasonov (2001).

In relation to the system component responsible for content, we could define the following directions of future work:

- Integration of MLS with real location services.
- An effective exploitation of Oracle Spatial and probably other GIS software. Various spatial queries and analysis functions implemented in the GIS software could be utilised in the MLS system.
- Providing dynamic information. An example of such information is information on the current traffic situation and roads condition.
- Providing information on mobile objects. Currently, the user is the only mobile object in the MLS system. In the present area, there are typically also other dynamic objects of interest, e.g. taxis, busses and other mobile users.
- XML-based interaction with content servers. For flexibility and scalability reasons, the MLS server and content servers should be separated. Some XML-based data formats should be used in interaction between them. However, this direction of work depends on the standardisation process held in World Wide Web Consortium.

Another important direction of future work is testing the solutions of MLS in new environments. New mobile terminals with different operating systems are emerging and new mobile networks are appearing.

ACKNOWLEDGEMENTS

The present work was done in MultiMeetMobile research project [<http://www.cs.jyu.fi/~mmm>] at the Information Technology Research Institute, University of Jyväskylä, Finland. MultiMeetMobile project was financially supported by the National Technology Agency (TEKES) under contracts 40599/99, 40779/99, 40640/00, and industrial partners Hewlett-Packard, Nokia Networks and Yomi Vision.

REFERENCES

- [1] Benefon, web site, <http://www.benefon.com>.

- [2] Burrough, P.A. and McDonnell, R.A., *Principles of Geographical Information Systems*, Oxford University Press, 1998.
- [3] Clarke, K.C. "Cartography in a Mobile Internet Age", Proceedings of the 20th International Cartographic Conference, Beijing, China, August 2001, pp. 1481–1488.
- [4] Garmash, A., *Management of geographic information in mobile environment*, M.Sc. thesis, University of Jyväskylä, Department of Computer Science and Information Systems, Jyväskylä, Finland, 2000.
- [5] Gartner, G., Uhlirz, S. and Pammer, A. "Mobile Internet: Applying Maps to Mobile Clients", Proceedings of the Workshop on Maps and the Internet in the 20th International Cartographic Conference, Beijing, China, August 2001.
- [6] Hardy, P.G., Haire, K.R., Sheehan, R. and Woodsford, P.A. "Mobile Mapping On-Demand, Using Active Representation and Generalisation", Proceedings of the 20th International Cartographic Conference, Beijing, China, August 2001, pp. 3239–3247.
- [7] OpenGIS. *Geography Markup Language (GML) 2.0*, OpenGIS implementation specification, OGC document number 01-029, 20 February 2001. Available in <http://www.opengis.net/gml/01-029/GML2.html>.
- [8] Katasonov, A., *Intelligent user-centred transmission of geographical data in mobile location-based services*, M.Sc. thesis, University of Jyväskylä, Department of Mathematical Information Technology, Jyväskylä, Finland, 2001.
- [9] Krasner, G.E. and Pope, S.T., A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80, *Journal of Object-Oriented Programming*, 1(3): pp. 26–49, 1988.
- [10] kXML, web site <http://kxml.enhydra.org>
- [11] Lehto, L. "XML in Web-based Geospatial Applications", Proceedings of the 3rd AGILE Conference on Geographic Information Science, Helsinki, Finland, May 2000.
- [12] Lehto, L. and Kilpelainen, T. "Generalizing XML-Encoded Spatial Data on the Web", Proceedings of the 20th International Cartographic Conference, Beijing, China, August 2001.
- [13] LIF, Location Interoperability forum, web site <http://www.locationforum.org/>
- [14] Liikenne ja viestintäministeriö, *Tie- ja katuverkon tietojärjestelmä DIGIROAD. Toteutussuunnitelma*. 2001, available in <http://www.mintc.fi/www/sivut/dokumentit/julkaisu/mietinnot/2001/b16.html>.
- [15] Mims, P., Pilkington, P. and Blizzard, J. "An object oriented transfer format for the future", Proceedings of the ASPRS 2001 conference, St. Louis, USA, April 2001.
- [16] Mitchell, K., "Wireless Mapping and Guidance Services", Proceedings of the 20th International Cartographic Conference, Beijing, China, August 2001, pp. 1785–1789.

- [17] Reichenbacher, T. "The World in Your Pocket - Towards a Mobile Cartography", Proceedings of the 20th International Cartographic Conference, Beijing, China, August 2001, pp. 2514–2521.
- [18] Singhal, S., Bridgman, T. Suryanarayana, L, Mauney, D., Alvinen, J., Bevis, D., Chan, J., and Hild, S., *The Wireless Application Protocol*, Addison-Wesley, Boston, 2001.
- [19] Virrantaus, K., Veijalainen, J., Markkula, J., Katasonov, A., Garmash, A., Tirri, H., and Terziyan, V., Developing GIS-supported location-based services, Proceedings of the First International Workshop on Web Geographical Information Systems (WGIS), Kyoto, Japan, December 3–6, 2001, IEEE CS, 2001.

DESIGN OF A MINIMAL SIGNALING BASED BANDWIDTH BROKERING PROTOCOL

Mohit Agarwal and Abhik Majumdar

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley, CA 94720

{mohit, abhik}@eecs.berkeley.edu

Abstract The DiffServ architecture uses automated agents known as Bandwidth Brokers for provisioning, allocating and managing QoS resources within domains as well as communicating with agents of other domains so that QoS guarantees can be provided across domains. The Bandwidth Brokers attempt to establish an end-to-end Quality of Service through signaling between each other. This leads to an implicit trade-off between scalability and the end-to-end QoS guarantees that can be provided. In this work we propose a new Bandwidth Broker signaling mechanism and show, through extensive simulations, that it is possible to specify end-to-end QoS guarantees that are sufficiently high even when signaling is minimal and link utilizations are very high. We have focused solely on the inter-domain communication between the brokers, assuming that the intra-domain issues have been handled separately. A major aspect of our design is that negotiations between Bandwidth Brokers are modeled as a function of available bandwidth and cost to the respective ISPs of the domains (whom the Bandwidth Brokers represent in business dealings). This makes our approach close to the real world scenario in which brokers will always try to end up with a profit after each negotiation process.

Keywords: Bandwidth Broker, Quality of Service, DiffServ

1. INTRODUCTION

Internet applications designers are increasingly feeling the need for guarantees in network performance. Moreover, the nature of these service guarantees varies with the applications. For example, applications such as voice-over-IP and video-conferencing require low round trip delays as well as a small jitter while in audio streaming larger round-trip delays are acceptable if jitter is low. Generally, a best effort kind of service may not be good enough to support such applications.

IETF standards for Integrated Services (Intserv) in routers [1] and the end-to-end reservation set-up protocol (RSVP) [2] were proposed to provide such service guarantees in the Internet. These standards introduce a per-flow state in network nodes in the network nodes along the path of the flow. While fine-grain service agreements can be supported through these approaches, these models do not scale well since maintaining per flow state for thousands of flows incredibly increases the complexity of the routers. The resource requirements (processing and storage) for running RSVP on a router increase proportionally with the number of separate sessions (i.e., RSVP reservations). Hence, supporting numerous small reservations on a high-bandwidth link may easily over tax the routers [8].

A more scalable approach called Differentiated Services (DiffServ) was proposed in [3]. Scalability is achieved here, by introducing a notion of “flow aggregation” whereby all flows at a network node belonging to a particular “class” are treated as a combined flow. Classification, marking, shaping and policing operations are performed only at the network boundaries or hosts. Typically, this is thought of as a two-tier model where several autonomous systems exist and each one of them is responsible for internally managing traffic to provide class-based services while they communicate with peer autonomous systems to set up service agreements that ensure that such services can be offered on an inter-domain or inter-network basis.

The DiffServ architecture defines the Service Level Agreement or SLA as a service contract between a customer and a provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another Diffserv domain. Diffserv architecture uses automated agents known as **Bandwidth Brokers** for negotiating SLAs between different autonomous systems and managing resources within the domain. In this paper, we assume that mechanisms for resource management within the domain exist and propose a signaling mechanism for setting up inter domain service agreements.

There has been a substantial amount of prior work in the area of implementation of Bandwidth Brokers. The Internet2 Qbone BB Advisory Council’s paper on Bandwidth Broker requirements for Internet2 Qbone Deployment [4] discusses the requirements of a Bandwidth Broker. It provides certain guidelines for implementing Bandwidth Brokers within a DiffServ framework and proposes various deployment phases designed to provide a staged approach to providing inter-domain communication. Both inter - and intra - domain issues are discussed.

A ‘two-tier resource management model for the Internet’ is suggested in [5]. In the realization of this model a Bandwidth Broker acts as a

resource manager for each administrative domain and neighboring brokers communicate with each other to establish inter-domain resource agreements.

An agent based architecture for quantitative service provisioning in DiffServ capable networks is proposed in [6]. The Bandwidth Broker is the agent responsible for admission control. The architecture provides for resource reservations for aggregated virtual leased lines between network domains. This approach provides support for both immediate and advance reservations.

From these studies it becomes evident that a critical aspect of the design and deployment of Bandwidth Brokers is the inherent trade-off between the end-to-end Quality of Service (QoS) guarantees that can be provided and the amount of signaling needed. Before a request for differentiated service is granted, the Bandwidth Broker needs to check for the availability of the resources that were requested. This involves analyzing the state of QoS resources within the domain as well as signaling to brokers of neighboring domains. Different signaling protocols are evaluated in [7] and an alternative signaling protocol is proposed.

In this paper, our main contribution is the development of a novel signaling protocol for inter-domain resource reservation. We model the negotiation between two brokers for a SLA as a monetary transaction which is based on the estimate of expected cost for the new SLA. Each broker maintains this estimate for each for its adjacent domains and updates it based on the distribution of its outgoing traffic and a history of previous negotiations. This signaling is very scalable and experimental results on a simulated system show that very good QoS can be achieved even when the resource utilization is very high.

The rest of the paper is organized as follows. In section 2 we present various signaling methods that were previously studied in [7]. In section 3 the proposed signaling mechanism is presented. Experimental results are discussed in section 4. We conclude the paper with a discussion and suggestion for future work in section 5.

2. BANDWIDTH BROKERS AND SIGNALING

As mentioned above Bandwidth Brokers are responsible for the overall management of the QoS resources within their domains. This includes the configuring of the routers within the domain and the edge routers on links to adjacent domains. The Bandwidth Brokers also manage inter-domain communication by talking to peer bandwidth brokers in adjacent domains. They gather and monitor the state of QoS resources within

their domains and on the edges of the domains. The brokers make admission control decisions based on this information, certain intra-domain policy specifications and the information they gather by talking to peer Bandwidth Brokers in adjacent downstream domains. The Bandwidth Brokers are responsible for negotiating and renegotiating SLAs between adjacent domains. The SLAs will specify the aggregate amount of QoS traffic crossing domains. This aggregation is a key feature of DiffServ architecture, which ensures scalability in terms of network size and traffic. The state sharing between brokers is done through a signaling protocol.

As was mentioned above, there is an inherent trade-off between the amount of signaling and the QoS guarantees that can be provided. This signaling may include notifying the adjacent brokers of how much Diff-serv traffic is being sent and/or renegotiating the SLA to reserve additional bandwidth. There are two possibilities:

- No notification: The simplest possible signaling protocol is when brokers make admission control decisions based on the state of QoS resources within the domain and the SLAs that exist with adjacent brokers. Naturally, it is almost impossible to provide QoS guarantees in such a scenario. However, the advantages of the protocol are that there are no notification costs and scalability is not a problem. It is possible, though, to provide some end-to-end statistical QoS guarantees by over-provisioning of the SLAs.
- End-to-End notification: Diametrically opposite of the no notification scenario is the end- to-end signaling protocol. Here each broker notifies its neighboring broker(s) of new Diffserv traffic before actually admitting the new request. If necessary, the broker also renegotiates the SLA. The neighboring brokers operate in a similar manner by notifying the downstream brokers. Hence, before a request is accepted all the brokers from the traffic source to the sink have been notified. While such an approach will enable the brokers to give end-to-end QoS guarantees, it is also obvious that such a method is not scalable because of the massive overheads in terms of bandwidth, delays and router complexity.

A “limited notification” signaling protocol was proposed in [7] as an alternative to the no-notification and end-to-end notification approaches. The proposed approach is to de-couple the notification process from the request reservation process. When a Bandwidth Broker receives a notification announcing DiffServ traffic on an incoming link it estimates the impact on the local network and the impact on the outgoing links. It then uses these estimates and a threshold called the reservation threshold to determine whether to renegotiate the SLA. Another threshold called

the notification threshold is used to determine whether to notify other Bandwidth Brokers. Typically this threshold is lower than the reservation threshold. SLAs are over-provisioned so that better (statistical) QoS guarantees may be provided. To limit the number of notifications, one notification is used for several aggregated flows. Hence, in this approach, the information regarding the destination of the flows cannot be specified in the SLA. To get around this problem, the use of exponential estimation is proposed, where each Bandwidth Broker maintains a matrix that specifies the probability that traffic coming in on link i will leave on link j . Periodically the broker updates the probability matrix based on the past history of how the traffic on an incoming link gets distributed into outgoing links.

Our proposed algorithm is, in part, inspired by the viability of the limited notification ideas as shown in [7]. We also use the idea of exponential estimation as suggested in [7]. However, [7] may face a scalability problem when the resource utilization is high and the amount of over provisioning is small. In such a scenario, most re-negotiation notifications would need to propagate from source of the traffic to its sink. In contrast, in our model, notifications/re-negotiations do not propagate beyond the adjacent Bandwidth Brokers and hence scalability is never a problem. Further, as explained below, we integrate a design for the monetary transactions involved in the negotiation process into our model. In fact, in our algorithm, the re-negotiations between brokers are based on comparisons between the expected revenue to a broker and the cost it incurs for renegotiating the SLA.

3. PROBLEM FORMULATION AND PROPOSED APPROACH

3.1. Problem formulation

The objective is to develop a scalable signaling mechanism that provides good statistical end-to-end QoS guarantees, using very limited notification. At the same time, the proposed solution should avoid a large amount of over-provisioning of the SLAs so that higher link utilization can be achieved.

3.2. Signaling Protocol

The Bandwidth Broker of each domain represents the Internet Service Provider (ISP) of that domain in its business dealings with other networks. The algorithm we design is presently designed for only a single class of DiffServ traffic, though the ideas presented may be readily ex-

tended to support multiple classes of DiffServ traffic. As in [7] we make the following assumptions:

- The ISP of a network demands money from other networks that want to reserve bandwidth for the injection of DiffServ traffic into its network.
- While the ISP for the source domain charges money from the client for sending DiffServ traffic, the sink domain does not ask money for sinking the DiffServ traffic (a kind of calling-party-pays regime).
- ISPs avoid breaking the SLAs

It is necessary for the Bandwidth Broker of the domain to dynamically renegotiate the bandwidth for DiffServ traffic with the Bandwidth Brokers of the downstream domains. This is because the amount of DiffServ traffic it needs to send fluctuates with time. For this purpose we propose a novel way to look at the dynamic re-negotiations of bandwidth for DiffServ traffic between two adjacent domains. Our algorithm is presently designed for only a single class of DiffServ traffic, though the ideas presented may be readily extended to support multiple classes of DiffServ traffic.

We model these re-negotiations as a function of available bandwidth, cost to the respective ISPs and a ‘shaping probability’ - the probability that the new DiffServ traffic will be shaped. The underlying assumption is that a Bandwidth Broker will update its SLA with an adjacent broker if the cost it faces for increasing the amount of reserved bandwidth is less than the revenue its ISP will get by accepting more DiffServ traffic from upstream domains and if the shaping probability is low enough. Thus we factor in the monetary transactions between the brokers into our model. The assumption that we make in the re-negotiations process also appeals intuitively since the broker will agree to upgrading the level of reserved bandwidth only if it estimates that the ISP which it represents stands to make a profit in the process.

For the purpose of renegotiating, each Bandwidth Broker (BB) maintains two metrics:

- An estimate of the **expected cost** it shall incur for increasing the DiffServ traffic (per unit) to each of its adjacent domains. This estimate actually evolves over time from an initial estimate as the brokers carry out dynamic re-negotiations for bandwidth. We anticipate that the expected cost shall vary over time due to the fluctuations in the way the traffic distributes. It may also vary if the brokers decide to vary their profit margins over time. An

exponential estimator is used for the dynamic updating of the expected cost. It is anticipated that the exponential estimation shall smoothen out the variations in the expected cost. We denote E_j^X as the expected cost that broker X maintains for its adjacent broker j . It represents a cost per unit of data.

- A **traffic distribution matrix** that specifies the probability that traffic (per unit) coming in on link i will leave on link j . These probabilities also evolve over time from an initial estimate (the initial estimate is that the traffic is likely to go to all outgoing links with equal probability). We denote T_{ij}^X as the (estimated) probability that traffic from broker i to X shall be forwarded to broker j . Here i and j are brokers adjacent to X. The idea of a traffic distribution matrix is drawn from [7].

We keep our admission control mechanism simple by admitting traffic at each domain only if the traffic being received from upstream brokers for forwarding is less than the negotiated bandwidth on the link in question and admitting the traffic shall not drive the total traffic on the link beyond negotiated levels. Of course there is a chance that DiffServ traffic even within negotiated amounts will be shaped (in some other downstream domain) since end-to-end signaling is not done. In practice more complicated admission control policies may be needed. However, for the purpose of demonstrating the efficacy of our minimal signaling inter-domain protocol, this simple protocol suffices.

3.3. Re-negotiations process

The notation used is:

T_{ij}^X	=	Traffic distribution probability at broker X
E_j^X	=	Expected cost at broker X (for j)
S_j^X	=	Negotiated bandwidth at X (for j)
D_j^X	=	Current Diffserv traffic from X to j
p_X^Y	=	Drop probability that broker Y returns to X
Δ_Y^X	=	Change in SLA that is requested by X to Y
Q_X^Y	=	Quote from Y to X
A_X	=	Set of brokers adjacent to X excluding Y
M_Y^X	=	Money that X gives to Y for the SLA
C_j^X	=	Data capacity of the link from X to j

The following series of steps occur during any re-negotiations process:

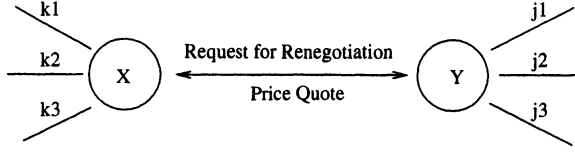


Figure 1. Re-negotiation process between brokers X and Y

- 1 The broker for domain X (BB_x) initiates re-negotiations with the broker for domain Y (BB_y) when the quantum of DiffServ traffic between the domains X and Y exceeds some predetermined fraction (f_h) of the currently reserved bandwidth for DiffServ traffic. By varying this fraction, the amount of over-provisioning built into the SLA can be varied.

$$\text{if } D_Y^X > f_h S_Y^X \quad \Delta_Y^X = \frac{2D_Y^X}{f_h + f_l} - S_Y^X \quad (1)$$

This increment Δ_Y^X is computed so that after the negotiations, the current traffic becomes less than $f_h S_Y^X(\text{new})$. One could use more complicated traffic estimators for computing this increment.

- 2 BB_y receives a re-negotiations request from BB_x which specifies the increase in the amount of reserved bandwidth X wants.
- 3 By using its traffic distribution matrix, BB_y computes the expected amount of extra traffic that it (Y) will have to pump in to each of its downstream links. Using this calculation and the expected cost of increasing its reserved bandwidth with each of its downstream domains, BB_y then calculates the expected cost it faces for carrying the extra traffic from X. This expected cost is sent as the price quote to BB_x . Specifically:

$$Q_X^Y = \Delta_Y^X \sum_{j \in A_Y} E_j^Y T_{Xj}^Y \quad (2)$$

- 4 BB_y also sends BB_x an estimate of the probability that the new increased traffic from X will be shaped. BB_y computes this probability based on the probable distribution of the traffic from X in to Y's downstream links and the maximum capacities of these links (that is the maximum up to which Y can increase the DiffServ traffic on these links). Specifically the drop probability is approximated by the expression:

$$p_X^Y = \frac{\sum_{j \in A_Y} \max\{(\Delta_Y^X + S_Y^X)T_{Xj}^Y - C_j^Y, 0\}}{\sum_{j \in A_Y} D_j^Y + \Delta_Y^X \sum_{j \in A_Y} T_{Xj}^Y} \quad (3)$$

This is the ratio of the total anticipated excess traffic on the outgoing links from Y to the total amount of outgoing traffic from Y.

- 5 BB_x receives the quote and drop probability from BB_y . BB_x updates its estimate of the cost it faces while increasing the DiffServ traffic between itself and Y according to the equation:

$$E_Y^X(\text{new}) = \alpha E_Y^X + (1 - \alpha)Q_X^Y \quad (4)$$

Here α is a number between 0 and 1 and is a simulation parameter.

- 6 The re-negotiations will be successful if BB_x sees that the drop probability is less than a predetermined (i.e. acceptable) level and the cost incurred due to the quote from Y is less than its estimated revenues. Let us denote estimated money that X receives from its upstream brokers for pushing traffic to Y by R_Y^X . This is the sum of the revenue that X is already receiving and the revenue that it expects to receive from its upstream brokers. This is calculated as

$$\begin{aligned} R_Y^X &= \sum_{k \in A_X} M_X^k T_{kY}^X \\ &+ \sum_{k \in A_X} \max\{D_X^k - S_X^k, 0\} E_X^k T_{kY}^X \end{aligned} \quad (5)$$

If X estimates its revenues to be greater than the cost incurred in pushing traffic to Y i.e.

$$R_Y^X - (Q_X^Y + M_Y^X) > 0 \quad (6)$$

it will accept the re-negotiation price sought by Y (provided the drop probabilities are below the predetermined level).

- 7 If the re-negotiations fail then X will again try to renegotiate with Y but this time with a smaller fraction of bandwidth than what was previously sought i.e $\Delta_Y^X(\text{new}) = \gamma \Delta_Y^X$ where γ is a simulation parameter between 0 and 1. This process continues until a negotiation succeeds or the additional bandwidth that is being sought becomes very small.

BB_x also initiates re-negotiations with BB_y when the quantum of DiffServ traffic between the domains X and Y falls below some predetermined fraction (f_l) of the currently allowed bandwidth for DiffServ traffic. At this point BB_x will inform BB_y that it wants to decrease the bandwidth for DiffServ traffic. In our model we assume that BB_y has to accept the request of BB_x for decreasing bandwidth. Here

$$\Delta_Y^X = \frac{2D_Y^X}{f_h + f_l} - S_Y^X \quad (7)$$

shall be negative and

$$M_Y^X(new) = M_Y^X + \Delta_Y^X E_Y^X \quad (8)$$

Periodically the traffic distribution matrix at each Bandwidth Broker is updated to reflect the changing distribution patterns within the network. Specifically BB_x will update its traffic distribution matrix using the equation:

$$T_{ij}^X = \beta T_{ij}^X + (1 - \beta) \frac{rate(i, j)}{D_X^i} \quad (9)$$

where β is a simulation parameter and lies between 0 and 1, $rate(i, j)$ is the total traffic currently going from i to j through X.

4. EXPERIMENTAL RESULTS

A simulator was built to study the performance of the proposed protocol. Experiments were carried out to study parameters of interest such as the expected cost and the kind of end-to-end QoS guarantees that can be provided at high network loads.

The simulations were carried out on a topology of 10 brokers. While the topology was that of a fully connected graph, different brokers were connected to different number of brokers and the capacities of the links between the brokers were chosen randomly. Further, the routing table for each broker was fixed at the start of simulations and was not varied with time. This was to prevent the routing algorithm from interfering with the performance of our signaling protocol. Each broker in our simulations is a source and sink of DiffServ traffic. We have currently implemented the simulator for only two classes of traffic namely DiffServ traffic and best effort traffic.

The objectives that we wished to achieve through the simulations were:

- 1 To study the total amount of DiffServ traffic (as a percentage of the total link capacities) flowing through the network over time.

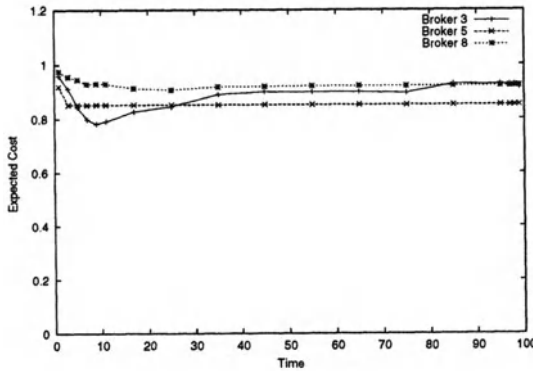


Figure 2. Evolution of Expected Cost with Time (at Broker 5).

Ideally we would want that the proposed algorithm would lead us to a situation where we will have a large amount of DiffServ traffic flowing through the network along with high QoS guarantees.

- 2 To study how the Quality of Service varies with the amount of over-provisioning of the SLA. Generally, we would expect that with increased over-provisioning the chances of shaping are lower and hence better QoS guarantees can be provided. Hence the yardstick on which we base our evaluation of the performance of our algorithm is whether a high QoS guarantee can be provided with sufficiently low over-provisioning.
- 3 To study the evolution of the metric of estimated costs that each Bandwidth Broker maintains. It is desirable that the estimated costs converge, which shall enable each Bandwidth Broker to accurately estimate the costs it faces while renegotiating for increasing the reserved bandwidth for DiffServ traffic with its adjacent domains.

4.1. Observations and Analysis

Figure 2 shows the evolution with time, of the expected cost at Broker 5 for its downstream brokers. This simulation was carried for 10% over provisioning (i.e. 10% of the negotiated bandwidth was kept as reserve) and it clearly shows that the expected costs settle down as time progresses. This means that as time progresses, the quote that the downstream brokers send back are very close to the expected value that the broker has for it, which is clearly very desirable. In this simulation the traffic generated at each Broker for different destinations had a fixed mean and small variance. In this case, the way traffic distributes was

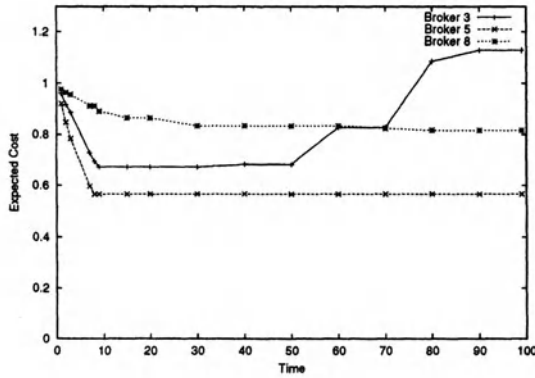


Figure 3. Evolution of Expected Cost with Time (at Broker 5). Mean traffic to various destinations is varied with time.

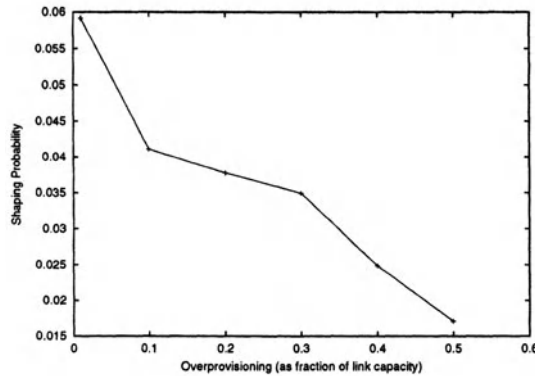


Figure 4. Shaping probability vs Over provisioning

seen to be almost constant since the routing is fixed and therefore the traffic follows the same paths. Figure 3 traces the same evolution when the traffic generated at each broker for different destinations had a time varying mean. For this case, the traffic distribution varies since the amount of traffic going to each destination changes as time progresses. This figure suggests that even though the quantum of traffic to different destinations varies with time, the expected cost is able to adapt to the changes quickly and stabilize. This can be observed from the fact that each time the traffic changes, the expected values change, but become constant after some time.

Figure 4 shows the variation of shaping probability with the amount of over provisioning. As expected, the shaping probability increases as the amount of over provisioning is reduced. However, what is encouraging is that even at very small over provisioning levels, the shaping prob-

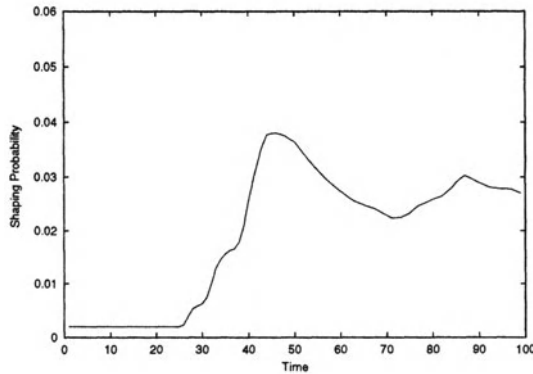


Figure 5. Evolution of Shaping probability with Time. Mean traffic to various destinations changes with time.

ability remains within acceptable limits. This implies that using our proposed protocol, the network is able to provide very high end-to-end QoS guarantees with very limited signaling. Further, it should be noted that these values for the shaping probabilities are obtained when the network is very close to saturation (as can be seen in Figure 6). This is significant since it is in the interest of the Bandwidth Brokers to be able to support high levels of traffic and still be able to provide its customers with very high QoS guarantees. Overall, therefore we can conclude that the Brokers are able to admit a large amount of traffic and still negotiate in time so that the traffic does not get dropped. Further, this also suggests that the estimates that a particular broker makes on behalf of its downstream brokers while sending back quotes to its upstream brokers are quite accurate. Figure 5 traces the shaping probability with time when the mean traffic to different destinations changes. These simulations were carried out with 10% over provisioning. It is clear that even with variation in the traffic the system is able to do quite well and the shaping probability remains quite low.

Figure 6 provides a clue to the level of network utilization. It depicts the evolution of actual DiffServ traffic that flows as a percentage of the negotiated bandwidth with time at Broker 5. Clearly, the utilization stabilizes to an amount close to the negotiated bandwidth with an over provisioning of 10%.

In all these simulations the cost to the customers is kept constant to simplify the understanding of the interaction of the quote sent by a downstream broker with the estimated inflows from upstream brokers during re-negotiations. However, it was observed that varying the costs does not affect the convergence of the algorithm and the performance of

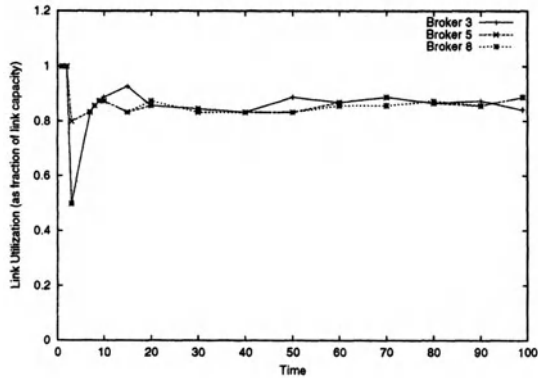


Figure 6. Evolution of Link Utilization with Time (at broker 5)

the protocol does not deteriorate in any way. Further, different topologies were used for carrying out the simulations and the performance for all the observed parameters were similar. These simulations were carried out for one hundred rounds. It was observed that carrying out more rounds of simulation did not result in any significant difference in the observations.

5. DISCUSSION AND FUTURE WORK

The control plane of the DiffServ architecture consists primarily of Bandwidth Brokers who establish an end-to-end quality of service through signaling between each other. The primary trade-off is between scalability and end-to-end Quality of Service. Through our simulations we were able to arrive at the following conclusions:

- Even though our signaling mechanism is minimal, in the sense that notifications and re-negotiations do not propagate beyond neighboring brokers, we are able to give sufficiently high end-to-end QoS guarantees to the clients. At the same time, the over-provisioning of the SLA is low enough to be attractive to the ISPs.
- The simulations show that the expected cost that each broker maintains converges if the variation in the traffic distribution matrix is not high. This indicates that our idea of modeling the re-negotiations not only as a function of available bandwidth but also of cost to the respective ISPs leads to a situation in which a broker shall be able to accurately estimate the costs it faces in any re-negotiations process. This is significant since the broker represents a business entity - the ISP of the domain - and will agree to any re-negotiations only when it sees a profit for itself.

The work carried out by us demonstrates that a limited notification approach to signaling between Bandwidth Brokers is feasible as well as attractive. In the re-negotiations protocol, we have made a number of simplifying decisions for some smaller problems. These could be replaced by more carefully tailored solutions in actual implementations. For example the additional bandwidth that a broker asks for reservation to its adjacent downstream broker could be based on a better estimator of the variations in traffic. During the re-negotiation process, the downstream Bandwidth Brokers return a quote as well as a shaping probability to the upstream Bandwidth Brokers. Better estimates for the shaping probability can be developed. Lastly, it might be possible to extend the above algorithm to handle advance reservations along with immediate reservations. This will enable ISPs to give their clients greater flexibility.

REFERENCES

- [1] R.Guerin, C.Partridge, and S.Shenker, "Specification of guaranteed quality of service," in *RFC (Standards Track) 2212, IETF*, Oct. 1997.
- [2] R.Braden, L.Zhang, S.Berson, S.Herzog, and S.Jamin, "Resource reservation protocol (rsvp) - version 1 functional specification," in *RFC (Standards Track) 2205, IETF*, Sept. 1997.
- [3] S.Blake, D.Black, M.Carlson, E.Davies, Z.Wang, and W.Weiss, "An architecture for differentiated services," in *RFC (Informational) 2475, IETF*, Dec. 1998.
- [4] "A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment: Version 0.7," Available at <http://www.merit.edu/working.groups/i2-qbone-bb/>
- [5] A.Terzis, L.Wang, J.Ogawa, and L.Zhang, "A two-tier resource management model for the internet.," in *Global Telecommunications Conference, GLOBECOM*, pp. 1779-1791 Vol.3, 1999.
- [6] O. Schelen, A. Nilsson, J. Norrgard, and S. Pink, "Performance of QoS agents for provisioning network resources," in *IWQOS*, pp. 17-26, 1999.
- [7] M. Gunter and T. Braun, "Evaluation of bandwidth broker signaling network protocols," in *ICNP*, pp. 145-152, 1999.
- [8] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment," in *RFC (Informational) 2208, IETF*, Sep. 1997.

A Client Puzzle Based Public-key Authentication Protocol

Chun-Kan Fung, M.C. Lee, and M.Y. Wong

Department of Computer Science and Engineering

The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

{ckfung, mclee, mywong}@cse.cuhk.edu.hk

Abstract Network denial-of-service attacks, which exhaust the server resources, have become a serious security threat to the Internet. Public Key Infrastructure (PKI) has long been introduced in various authentication protocols to verify the identities of the communicating parties. Although the use of PKI can present difficulty to the denial-of-service attackers, the underlying problem has not been resolved completely, because the use of public-key infrastructure involves computationally expensive operations such as modular exponentiation. An improper deployment of the public-key operations in a protocol allows the attacker to exhaust the server's resources. This paper presents a public-key based authentication protocol integrated with a sophisticated client puzzle, which together provides a good solution for network denial-of-service attacks, and various other common attacks. The basic strategy to protect against denial of service is to impose an adjustable cost on the attacker while it launches the attacks. The proposed client puzzle protocol can also be integrated with other network protocols to protect against denial-of-service attacks.

Keywords: Authentication Protocols, Client Puzzles, Denial of Service, Network Security, Attacks.

1. INTRODUCTION

Denial-of-Service (DoS) attack is characterized by an explicit attempt of the attackers to prevent legitimate users of a service from using that service [1]. There has been a growing concern about this attack in recent years,

especially after the series of denial-of-service attacks on the Internet on February 2000 [2].

Denial-of-Service attacks can be classified into three types. These include (i) consumption of scarce resources; (ii) destruction or alteration of configuration information; and (iii) physical destruction or alteration of network components [1]. Any protocol where the server commits extensive computations or memory allocation prior to or as a part of client authentication is vulnerable to network DoS attacks [3], which corresponds to the first type of denial-of-service attack above. The basic strategy in handling the denial-of-service attack is that the server should require the client to commit its resources before extensive resources could be committed to the client in an authentication protocol session.

TCP connection protocol was reported to be vulnerable to SYN flooding attack in 1996 [4]. This is one of the well-known DoS attacks against a network protocol. During the attack, the attacker sends a huge number of SYN messages to the server to initialize connections and then leaves the subsequently established connections unattended. Since the server allocates buffer space right after the reception of a SYN message, memory of the server can thus easily be exhausted by such connection requests. Juels and Brainard [5] pointed out that the SSL protocol [6] is vulnerable to a similar form of attack. However, it is the CPU resource rather than the memory space, which could be exhausted rapidly because expensive cryptographic operations are required on the server-side in the SSL protocol.

Entity authentication has long been used in communication protocols to tackle different attacks, e.g. the man-in-the-middle attack. Authentication based on the public-key infrastructure is computationally expensive, as the underlying cryptographic operations such as the modular exponentiation involve extensive computations. An attacker may be able to exhaust the computational or memory resources of its target if the underlying authentication or communication protocol is improperly designed. For instance, Ng and Tan's protocol [7] is one that is vulnerable to the resource consumption attack; the server is required to perform public-key decryption right after the reception of the client request message in this protocol. There are several other protocols identified as being vulnerable to this type of attack [8] [9] [10].

Dwork and Naor first introduced the concept of client puzzle to combat junk mail attack [11]. A client puzzle is a small cryptographic problem created by the server in response to a client request. The client should first commit its resources to solving the puzzle before completing the remaining part of the communication protocol. While the legitimate users could experience only a slight degradation of service under this scheme, a big cost will be imposed on a denial-of-service attacker who tends to create a huge

number of requests within a short duration, because the attacker is required to solve a unique puzzle for each of its service requests. With appropriate adjustment on the puzzle difficulty level, the client puzzle method can handle relatively fast attacks and allow for graceful service degradation. Aura, Nikander and Leiwo [12] proposed a client puzzle and applied it to an authentication protocol. The client puzzle used has an inherent problem that the solution to a puzzle may not exist since the protocol does not verify the existence of a solution during puzzle construction. A later authentication protocol designed also by Aura, Nikander and Leiwo [3] employs a puzzle without the above problem. However, similar to the approach proposed by Matsuura and Imai [10], the puzzle does not have a difficulty level; so graceful degradation of service would not be possible. The authentication protocol is also vulnerable to denial-of-service attack because the puzzle is not created in a stateless way. Juels and Brainard from the RSA Laboratories proposed a client puzzle protocol to address the TCP SYN attack [5]. This client puzzle protocol does not have the above-mentioned problems; however, there are still other identified defects.

In this paper, we propose an enhanced version of the client puzzle protocol of Juels and Brainard [5]. The defects in Juels and Brainard's protocol will be identified and removed from our client puzzle solution. In addition, a public-key based authentication protocol integrated with our client puzzle protocol would be proposed. The paper further demonstrates how the proposed integrated authentication protocol can resist effectively the network DoS attack and other types of common attacks.

2. AN IMPROVED CLIENT PUZZLE PROTOCOL

This section presents our client puzzle protocol for handling the denial-of-service attacks. This client puzzle can be added on top of an existing network protocol in order to protect the protocol against network DoS attacks.

2.1 Protocol Descriptions

Consider the use of our client puzzle to protect a protocol M against the network denial-of-service attacks. To initiate the execution of the protocol M , the client submits its unique identity C in the form of a request message to the server.

The server then checks the availability of system resources in order to determine the difficulty level k of the client puzzle to be sent out. The resource availability may refer to the availability of memory or CPU

resource or both. Normally, the difficulty level is inversely proportional to the availability of the system resources, allowing a graceful degradation of services during the denial-of-service attacks. In a normal situation, k is set to zero, and no puzzle needs to be solved by the clients.

After the determination of the difficulty level, the puzzle would be constructed using two one-way, collision-resistant hash operations. First, a bit-string $z1$ is generated by the hash operation $h(C, s, t)$, where the hash function h would operate on a single input bit-string formed by the concatenation of the parameters C , s and t . Then a second hash operation is performed on $z1$ to produce another bit-string $z2$. The client identity C is included in the hash input to make the puzzle client specific so that a solution, for a given puzzle, found by one client cannot be used by other clients. Parameter s is the server secret key, which should be long enough (e.g. 128 bits) to prevent brute force attacks. In order to impose a limit time for solving the puzzle by the client, the puzzle should be associated with a timestamp, t , which is set to the current date and time with a precision up to a second. The client needs to solve the puzzle before it expires; otherwise, the protocol execution will be terminated no matter whether the returned solution is correct or not.

Assume the hash output is of length L bits. The puzzle is made up of the hash image $z2$, the partial pre-image $z1<k+1, L>$, i.e. the $(k+1)^{th}$ to the L^{th} bits of $z1$, the timestamp t and the difficulty level k . In this way, the puzzle can be constructed in a totally stateless fashion. It is important to commit as little resource as possible in the construction of a puzzle because the DoS attackers intend to exhaust the server resources.

To solve the puzzle, the client finds by brute force a k -bit string $z1'<1, k>$, which, when concatenated with the given partial pre-image $z1<k+1, L>$, satisfies the following relation:

$$z2 = h(z1'<1, k>, z1<k+1, L>)$$

After solving the puzzle, the solution $z1'<1, k>$ together with the received $z1<k+1, L>$, t and k and the client identity C would be sent back to the server for verification.

Before verifying the puzzle solution, the server first checks the timestamp t to see if the puzzle has expired. If T is the maximum allowable time for the client to solve the puzzle and to return the solution, the server must ensure that $\tau - t \leq T$ where τ is the current date and time on server side. If the timestamp has not expired, the server performs the same hash operation during puzzle construction using the client-submitted C and t to get the bit-string $m = h(C, s, t)$. The last $L-k$ bits of this bit-string (i.e. $m<k+1, L>$) are compared with the received $z1<k+1, L>$ to make sure that the timestamp and

the client identity are the same as the original ones. After that, the client-purported solution is verified by checking the equality between $h(m)$ and the output from the hash operation on the bit-string formed by the concatenation of $z1' < 1, k >$ and $m < k+1, L >$. If the equality holds, the solution can be assured to be correct; the server can then commit its resources to the client, otherwise, the request is rejected.

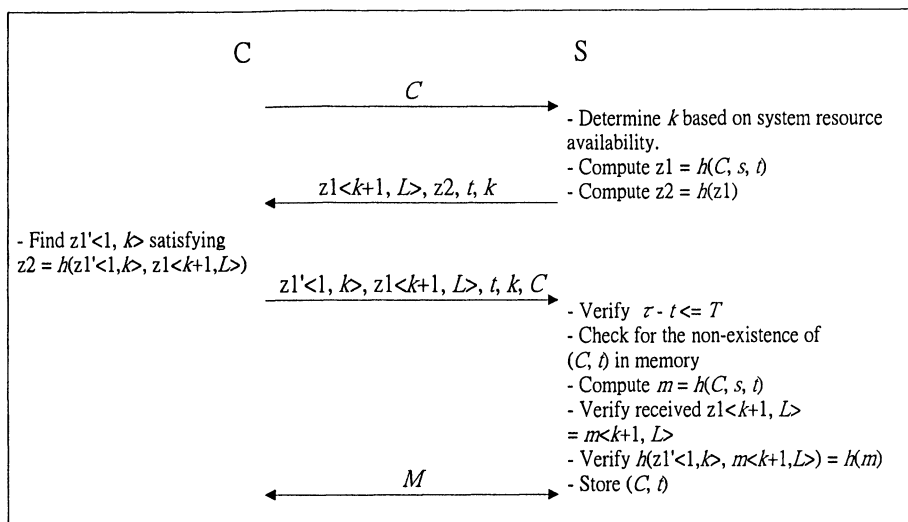


Figure 1. The client puzzle protocol

Prevention of the pre-computation of puzzle solution is a basic puzzle design issue. In our client puzzle protocol, a timestamp t and a long server secret key s are the parameters used in creating the client puzzle. This makes the puzzle component $z1 < k+1, L >$ unpredictable to the client; and so, the solution cannot be pre-computed. However, if the time parameter T is set to be too large, a client (or attacker) can submit the same solution a number of times before the puzzle expires. So the server should register the pair (C, t) in memory after the solution has been verified correct. This indicates that client C has submitted a correctly solved puzzle created at time t . Before verifying a given solution from any client C , the server should check to ensure that there is not any pair of the form (C, t) in memory for the client as depicted in Figure 1. This pair (C, t) does not need to be kept in memory for a long time; it will be removed once the puzzle has expired. In theory, upon confirming a given puzzle solution, if $\tau - t = T$, then no pair of the form (C, t) needs to be kept for client C . However, if $\tau - t < T$, the duration for which (C, t) needs to be stored is $(T - (\tau - t))$, which is usually a short period.

2.2 Comparisons and Improvements

In the client puzzle protocol proposed by Juels and Brainard [5] (referred to as the Juels-Brainard protocol hereafter), a puzzle can be made up of n (≥ 1) sub-puzzle(s) each having the same difficulty level k . The probability for an attacker to guessing the solutions correctly is $1/2^{kn}$. To reduce this probability to a certain level, the value of n can be increased. However, the same level can be achieved by increasing the value of k with a smaller n . The latter approach has the advantage of having a smaller overhead in puzzle transmission because the puzzle size is smaller with a smaller n . Another advantage is the lower cost in puzzle creation and verification because the number of sub-puzzle creation and verification is smaller with a smaller n . It is important to commit as little resource as possible in puzzle creation and verification by keeping n small in order to prevent DoS attack. The cost of this approach is that the average work factor $(1+2^k)n/2$ to a client in solving the puzzle is higher for the same $1/2^{kn}$. Our puzzle can be viewed as made up of one single sub-puzzle ($n = 1$). However, it can be easily extended to one consisting of more than one sub-puzzle by adding an additional sub-puzzle index to the input parameter list of the hash operation during the puzzle construction stage.

To allow for a fair comparison, a puzzle of the Juels-Brainard protocol is meant hereafter to be one consisting of one sub-puzzle only, i.e. $n = 1$.

In the Juels-Brainard protocol, the authors claim that a collision-resistant hash function is not a requirement. However, their puzzle solution verification process does not take this claim into account. In the puzzle creation stage of the Juels-Brainard protocol, a bit-string x is computed by the server as a hash of a set of service parameters and a server secret key. Then another bit-string y is computed as a hash of x , i.e. $y = h(x)$. The server then sends this y and part of x to the client who is required to find the remaining part of x by brute force using the equation $y = h(x')$. In the verification stage, the server generates x again and compares it against the client-purported solution x' to determine the correctness of this solution. However, a non-collision-resistant hash function means that apart from the server-generated x , there could be another x' satisfying the equation $y = h(x')$. So the Juels-Brainard protocol could treat a correct solution as wrong. In our client puzzle protocol, the client-purported solution will not be compared directly with the server-generated x . Instead, an additional hash operation would be performed on the client-purported solution to see whether the output is equal to y . With this modification, a correct solution will not be marked as incorrect even if a non-collision-resistant hash function is used. This modification would also be useful when using a collision-resistant hash function because collision-resistant is not a guaranteed property. Collision

could still occur in some claimed collision-resistant hash functions, e.g. MD4 [13].

In the Juels-Brainard protocol, the client needs to submit only the found leading k -bit sub-string of x as solution; the remaining $L-k$ bits of x are to be computed by the server during solution verification. When compared with our approach, their solution would be $L-k$ bits shorter. However, it is inappropriate to rely on the server to calculate the remaining $L-k$ bits. Consider the case with the puzzle difficulty level k equal to 1. After solving the puzzle, instead of submitting the solution immediately, the client stores the solution for later use. Suppose after the puzzle has expired, the client retrieves the solution from storage and submits it. The client knows that the server could possibly reject the solution as the puzzle has expired. So, before submission, it updates the timestamp t in order to cheat the server. On the server side, the 1-bit solution is computed using this fake timestamp, and is then compared with the client-purported solution. Although the timestamp has been modified, the probability of producing the same 1-bit solution as the client-purported one is still very high. The problem is that the solution is very short. In this example, this probability is $1/2$. For $k = 2$, this probability is $1/2^2$. In general, this probability is $1/2^k$. We can see that the lower the value of k , the more severe would be the problem. So one possible solution is to set a bigger lower bound for k (or n in the Juels-Brainard protocol). However, this makes graceful degradation of service difficult and the clients will experience a large degradation of service quality in a short time at the beginning of a potential DoS attack. Therefore, we require the client to submit both the solution and the received bit-string $z_{1<k+1, L>}$ instead of the solution $z_{1<1, k>}$ alone in our protocol. The server should check whether the received bit-string $z_{1<k+1, L>}$ matches the server-computed bit-string $m_{<k+1, L>}$ before verifying the solution. With such modifications included in our protocol, we consider again the probability of producing a matched $m_{<k+1, L>}$ given the submitted $z_{1<k+1, L>}$ when the timestamp has been modified. Among the 2^L possible values of m , 2^k of them have the sub-string $m_{<k+1, L>}$ matching the given $z_{1<k+1, L>}$. Assume the hash outputs are uniformly distributed. The probability of producing a bit-string m , using the fake timestamp, with the last $L-k$ bits (i.e. $m_{<k+1, L>}$) matching the client-submitted $z_{1<k+1, L>}$ is $1/2^{L-k}$. According to this probability $1/2^{L-k}$, the higher the value of k , the higher would be the chance of the matching when the modified timestamp is being used to generate the bit-string m . To make this probability close to zero, we suggest the value of k to be between 0 and $L/2$; and L should have a length of at least 128 bits. In fact, the current common hash functions such as MD5 and SHA have a hash output with a length not less than 128 bits.

Before a puzzle expires, a client could submit the same solution a number of times, leading to multiple allocations of server resources. In the Juels-Brainard protocol, it has a mechanism to prevent multiple allocations of buffer resources. This is done by associating a unique identifier derived from the first request message (M_i^1 in Juels-Brainard protocol notation) to the corresponding allocated buffer slot; and the server ensures that only one buffer slot would be allocated for each request. However, the authors assume the unavailability of PKI in the design of their protocol because they claimed that PKI could effectively prevent denial-of-service attacks. And no additional benefits could be achieved when using client puzzle in the presence of PKI. In fact, this is not true because the computationally expensive operations such as modular exponentiation in PKI could open up the CPU exhaustion attack. The client puzzle protocol could be used to protect a public-key based authentication protocol. In this case, after verifying the puzzle solution, the server may need to perform the expensive public-key cryptographic operations in executing the subsequent authentication protocol. In our protocol, instead of relying entirely on the client request message for the unique identifier, we make use of the high precision requirement (at least up to a second) of timestamp. The unique identifier is derived from the client unique identity C , which appears in the first request message, and the puzzle timestamp t , which is generated by the server, to form the unique pair (C, t) . This mechanism can reduce the number of parameters in the client request message to a minimal. Whenever a solution has been verified correct, the unique pair (C, t) is stored in memory until the corresponding puzzle expires. As depicted in Figure 1, the server needs to check for the non-existence of the unique identifier (C, t) for each service request before verifying the puzzle solution. To allow for a rapid search for the existence of an identifier, bucket hashing on the identifiers could be employed. If the existence of a submitted identifier (C, t) can be found, the protocol should be terminated immediately and both CPU and memory resources are not allocated. If the client submits a fake (C, t) , it cannot pass the subsequent solution verification. Therefore, if a client (or attacker) reuses the same solution before the puzzle expires, the solution will be rejected. Thus multiple allocations of both extensive CPU and memory resources would not occur.

3. THE PROPOSED AUTHENTICATION PROTOCOL

The proposed client puzzle protocol can be integrated with or added on top of a network protocol which it protects. In this section, we propose a

public-key authentication protocol, and demonstrate how our client puzzle protocol can be securely and properly integrated with the protocol which it protects. If the integration of the client puzzle and the network protocol is not done properly, not only the denial-of-service attacks cannot be prevented, and the underlying protocol may be liable to further security attacks.

3.1 Notations

Let k_{Σ} be the public key of the entity Σ , and k_{Σ}^{-1} be the corresponding private key. $E\{w\}_{k_{\Sigma}}$ denotes the encryption of message w using the key k_{Σ} . N_{Σ} is a random nonce generated by entity Σ . The other notations are the same as those in the client puzzle protocol.

3.2 Protocol Descriptions

The proposed mutual authentication protocol is a challenge-response and timestamp-based authentication protocol. Though a timestamp parameter is employed in the protocol, clock synchronization is not required. The proposed authentication protocol consists of 4 message exchanges between the client and the server. The first two are basically the same as the first two messages in the client puzzle protocol depicted in Figure 1.

It is assumed that the communicating entities have authentic copies of all the required public keys. The authentication protocol begins by client C sending an authentication request message to the server S .

Message 1: $C \rightarrow S$: C

Message 1 of the authentication protocol, the same as the request message in the proposed client puzzle protocol, is used as the authentication request message. In this message, client identity C is used as the only parameter to formulate the authentication request. It serves as a *hello* message only; and does not trigger the execution of expensive computations or the allocation of memory resources by the server. Otherwise, the integrated protocol could not protect against DoS attacks.

After the reception of the client request, the server performs the same tasks as outlined in the client puzzle section. The tasks include the determination of the puzzle difficulty level k based on the availability of system resources, and the construction of the client puzzle. If k is set 0, the computation of $z1$ would still be necessary. As mentioned in the client puzzle protocol, the string $z1 \langle k+1, L \rangle$ can be used to validate the timestamp in message 3 on the server side. So it is important to ensure message freshness for protection against replay attacks.

Message 2: $S \rightarrow C: z1<k+1, L>, z2, t, k$

The second message is also the same as the second one in the client puzzle protocol. The server passes the formulated strings $z1<k+1, L>$ and $z2$ (for $k \neq 0$) together with the timestamp t and difficulty level k to the client.

On reception of message 2, if the difficulty level is greater than zero, the client finds a solution to the puzzle by brute force. Otherwise, no puzzle needs to be solved.

Message 3: $C \rightarrow S: z1'<1, k>, z1<k+1, L>, t, k, C, S, E\{N_C\}k_S, E\{h(z1'<1, k>, z1<k+1, L>, t, k, C, S, E\{N_C\}k_S)\}k_C^{-1}$

Message 3 from the client puzzle protocol should be enhanced before it can be used in our authentication protocol. In message 3 of the client puzzle protocol, $\{z1'<1, k>, z1<k+1, L>, t, k, C\}$ is the message that needs to be passed to the server S . To authenticate the client to the server, the client produces a signature of the message using its private key, i.e. $E\{h(z1'<1, k>, z1<k+1, L>, t, k, C, S)\}k_C^{-1}$. We emphasize here that the plaintext counterpart (i.e. $\{z1'<1, k>, z1<k+1, L>, t, k, C, S\}$) of this encryption must also be submitted to the server. Although the client can still authenticate to the server by sending only the encryption output $E\{z1'<1, k>, z1<k+1, L>, t, k, C, S\}k_C^{-1}$, the server will then need to perform a computationally expensive public-key decryption operation in order to retrieve the puzzle solution and the timestamp for verification. Denial-of-service attack can then be achieved by repetitively sending bogus encryption outputs to trigger the server executing the decryption operations repetitively. So the plaintext counterpart should also be submitted to allow the server to perform the necessary verifications, like the client puzzle protocol, before performing the expensive public-key decryption in order to authenticate the client.

To prevent replay attack, a nonce N_C is randomly generated and sent to the server. This nonce serves as a challenge to the server, which is to be returned in the next message. It is signed using the client's private key to form $E\{h(E\{N_C\}k_S)\}k_C^{-1}$. This signature can be combined with the previous signature to form $E\{h(z1'<1, k>, z1<k+1, L>, t, k, C, S, E\{N_C\}k_S)\}k_C^{-1}$. This enhanced message 3 of the client puzzle protocol forms the message that should be sent in the third flow of the authentication protocol.

On the server side, the server performs as similar set of verifications like those in the client puzzle protocol as shown in Figure 1. If $k = 0$, no solution needs to be verified and the verification $h(z1'<1, k>, m<k+1, L>) = h(m)$ can be omitted. After that, the signature of the client is verified.

Message 4: $S \rightarrow C: S, C, E\{N_C\}k_S, E\{h(S, C, E\{N_C\}k_S)\}k_S^{-1}$

The remaining task of the protocol is to authenticate the server to the client. The client challenge $E\{N_C\}k_S$ is signed by server's private key in order to authenticate the server to the client.

On the client side, if the signature on message 4 is verified to be correct, the server is said to be authentic to the client. Figure 2 shows the overall picture of the proposed authentication protocol. If the submitted solution is incorrect, the execution of the protocol will be terminated.

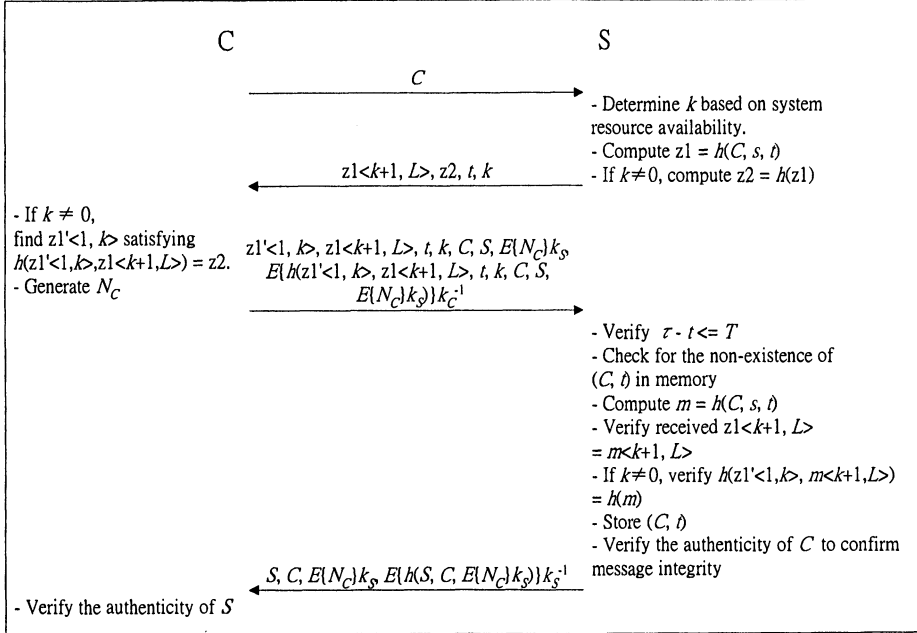


Figure 2. The DoS-resistant public-key authentication protocol

3.3 Security Analysis

The logic of the proposed authentication protocol has been analyzed and verified formally using the Coffey-Saidha technique [14]. In this section, the protocol is further analyzed against the common set of attacks on an authentication protocol [15].

3.3.1 Denial-of-Service Attack

This section analyzes the proposed authentication protocol to check if an unauthenticated client (or attacker) could exhaust server resources.

In our authentication protocol, the server receives two messages from the client. After the reception of message 1, no expensive computations are needed. The construction of the puzzle requires just two relatively fast hash

operations. Also, the puzzle is constructed in a stateless fashion; so no memory resource is consumed. After the reception of message 3, computationally expensive public-key operations are needed to verify the client signature, and memory allocation is required to store the (C, t) pair. However, these resources will only be consumed after the submitted solution has been verified correct, i.e. the client commitment of its resources has been assured. The non-existence checking of (C, t) and the timestamp checking prevent an attacker from using a previously submitted solution. Therefore, the proposed authentication protocol ensures that the server will not be required to commit its extensive resources before confirming that the client has consumed some of its resources during the authentication protocol session. Further, an attacker could experience the risk of exhausting its own resources if it keeps on issuing authentication requests to the server. However, the server will assign only one copy of its memory resource to an authenticated client.

3.3.2 Replay Attack

Replay attack refers to those attacks involving the use of information from a single previous protocol execution, on the same or a different verifier [15].

The basic techniques to avoid replay attack include the use of the challenge-response technique; the use of nonces; and the embedding of target identity in a response message [15].

In a public-key cryptographic protocol, the potential messages that an attacker may replay are those that the attacker is unable to produce. In our authentication protocol, messages 3 and 4 are two such messages as they are encrypted using private keys unknown to the attacker.

In message 3 of the proposed authentication protocol, there is a bit-string $z1<k+1, L>$ embedded inside the encryption using private-key k_C^{-1} . This bit-string is generated and sent by the server to the client in message 2 of the protocol. It is unpredictable to the client as it is generated using one-way hash function with a long server secret key s and current timestamp t as inputs. Also, as shown in Section 2.2, the length of $z1<k+1, L>$ must be sufficiently long (at least 64 bits) for the server to detect a fake timestamp. The bit-string $z1<k+1, L>$ thus serves the purpose of a nonce for the server to ensure the freshness of message 3 and so the replay attack is prevented.

The replay of message 4 is prevented by challenge-response technique. A long random nonce N_C is generated and sent by the client to the server in message 3. It is then embedded inside the signature of message 4 in order to assure to the client the freshness of this message.

3.3.3 Interleaving Attack

Interleaving attack refers to those attacks involving selective combination of information from one or more previous or simultaneously ongoing protocol executions (parallel sessions), including possible origination of one or more protocol executions by an adversary itself [15].

To compromise the proposed protocol using interleaving attack, the attacker must be able to derive those ciphertexts produced using the unknown private keys. In the proposed protocol, none of these ciphertexts in any of the protocol messages has the same format as the other three protocol messages or their components. Also, none of them has a format, which is a combination of the formats of the other three messages and their components. This makes the attacker unable to derive the desirable ciphertexts in one message from the other 3 messages of our protocol for an interleaving attack. If the attacker uses the i^{th} protocol message of a previous protocol instance as the i^{th} protocol message in the current instance, this corresponds to the replay attack, which has been addressed in Section 3.3.2 above. Another possible mode of interleaving attack is by forced delay [15], in which an attacker intercepted the i^{th} protocol message of one protocol instance, relayed it at some later point in time and then uses it as the i^{th} protocol message of another protocol instance in order to achieve impersonation or other deceptions. However this is not considered as a true attack because the attacker acts only as part of the communication link between the communicating parties without altering the content and aliveness of the protocol messages [15].

3.3.4 Others

As all the four protocol messages are asymmetric to each other, with identities of the sender and receiver embedded inside the cryptographic messages, together with the use of asymmetric keys, reflection attack [15] is thus prevented in the proposed protocol. Besides, the man-in-the-middle attack can also be easily prevented in the protocol as this attack can be foiled by the adoption of certificate-based public-key signature [16].

4. CONCLUSIONS

This paper proposes a client puzzle based authentication protocol which can effectively resist the denial-of-service attacks, and other common contemporary attacks. The integrated authentication protocol relies on our carefully designed client puzzle protocol, which can be seen as an enhanced

version of the client puzzle proposed by Juels and Brainard [5], to tackle the network denial-of-service attack. Our client puzzle protocol has overcome problems inherent in the proposal of Juels and Brainard.

Concerning our proposed integrated authentication protocol, for each protocol execution request from a client, a unique puzzle would be generated and sent by the server to the client. The client must commit its resources to solve the puzzle before expensive computations and memory allocations would be performed by the server. This cost of attack serves as a deterrent to the attacker in carrying out server resource exhaustion attacks. The proposed client puzzle protocol is quite flexible as the difficulty level of the puzzle can be adjusted according to the degree of availability of the server resources. This means that if the attackers manage to consume some of the server resources, the server would increase accordingly the difficulty level of the client puzzles which would subsequently consume more resources of the attackers. Therefore, the protocol could intelligently monitor the attackers and launch counter through adjusting adaptively the puzzle difficulty level.

The proposed authentication protocol remains stateless and refuses to perform expensive public-key cryptographic computations until the solution of the client puzzle has been verified. Thus it can resist server resource exhaustion attacks. Besides, the proposed authentication protocol has been able to resist various common attacks. As the DoS attacks have become widely known and can easily be exploited to attack websites and various network protocols, they have created a real problem to protocol design. Protocols vulnerable to DoS attacks could be strengthened with the integration of our proposed client puzzle protocol. In addition, network protocol designers should consider integrating the proposed client puzzle protocol into their protocols to avoid being harassed by the DoS attackers.

REFERENCES

- [1] CERT Coordination Center, "Denial of Service Attacks," Tech Tips, June 4, 2001. Available: http://www.cert.org/tech_tips/denial_of_service.html
- [2] Garber, L., "Denial-of-service attacks rip the internet," *IEEE Computer*, vol. 33, Issue 4, pp. 12–17, April 2000.
- [3] Jussipekka Leiwo, Pekka Nikander, Tuomas Aura, "Towards network denial of service resistant protocols," in *Proceedings of the 15th International Information Security Conference (IFIP/SEC 2000)*, Beijing, China, August 2000. Kluwer.
- [4] CERT Coordination Center, "TCP SYN Flooding and IP Spoofing Attacks," CERT Advisory CA-1996-21, September 19, 1996. Available: <http://www.cert.org/advisories/CA-1996-21.html>

- [5] Ari Juels and John Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proceedings of the 1999 Network and Distributed System Security Symposium (NDSS '99)*, pp. 151-165, San Diego, California, February 1999.
- [6] Alan O. Freier, Philip Karlton and Paul C. Kocher, "The SSL Protocol Version 3.0," Internet Draft, March 1996
- [7] Li Huang Ng and Tan. D.T., "A novel JavaCard-based authentication system for secured transactions on the Internet," in *Proceedings of 2000 IEEE International Conference on Networks (ICON 2000)*, pp. 262-266, 2000.
- [8] S. Hirose and K. Matsuura, "Enhancing the Resistance of a Provably Secure Key Agreement Protocol to a Denial-of-Service Attack," in *Proceedings of the 2nd International Conference on Information and Communication Security (ICICS'99)*, pp.169-182, Sydney, Australia, November 1999. Springer.
- [9] S. Hirose and K. Matsuura, "Enhancing the resistance of a secure key agreement protocol to a denial-of-service attack," in *Proceedings of the 1999 Symposium on Cryptography and Information Security (SCIS'99)*, Vol. II, pp. 899-904, Jan. 1999.
- [10] K. Matsuura and H. Imai, "Protection of Authenticated Key-Agreement Protocol against a Denial-of-Service Attack," in *Proceedings of 1998 International Symposium on Information Theory and Its Applications (ISITA'98)*, pp. 466-470, Oct. 1998.
- [11] Cynthia Dwork and Moni Naor, "Pricing via processing or combatting junk mail," in Ernest F. Brickell, editor, *Advances in Cryptology -- CRYPTO '92*, Volume 740 of Lecture Notes in Computer Science, pp. 139-147, 16-20 August 1992. Springer-Verlag, 1993.
- [12] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo, "DOS-resistant authentication with client puzzles," in Bruce Christianson, Bruno Crispo, and Mike Roe, editors, *Proceedings of the 8th International Workshop on Security Protocols*, Lecture Notes in Computer Science series, Cambridge, UK, April 2000. Springer-Verlag.
- [13] Hans Dobbertin, "Cryptanalysis of MD4," *Fast Software Encryption, Third International Workshop*, Volume 1039 of Lecture Notes in Computer Science (D. Gollmann, ed.), pp. 53-69. Springer-Verlag, 1996.
- [14] T.Coffey, and P.Saidha, "Logic for verifying public-key cryptographic protocols," *IEE Proceedings in Computers and Digital Techniques*, vol. 144, Issue 1, pp. 28-32, January 1997.
- [15] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
- [16] Dong Hwi Seo and P. Sweeney, "Simple authenticated key agreement algorithm," *IEE Electronics Letters*, vol. 35 No. 13, June 1999.

NAAP: A User-To-Network Authentication Protocol

Henry Haverinen
Nokia Mobile Phones

Abstract This paper presents the design of an IP-level user-to-network authentication protocol called the Network Access Authentication Protocol (NAAP). The protocol is an extensible UDP-based protocol that is used by a mobile terminal and an edge router for IP network access authorization for example on public Wireless Local Area Networks (WLANs). NAAP can use any Extensible Authentication Protocol (EAP) method that is capable of distributing a session key. It includes authentication service discovery, message integrity protection, protection against message replays, deregistration and a mechanism to detect if a terminal has silently left the access network. The protocol has been implemented and it is used in the Nokia Operator Wireless LAN solution [1] with GSM SIM authentication.

Keywords: Network access authentication, user-to-network authentication, extensible authentication protocol

1. INTRODUCTION

Network access authentication is often performed at layer 2 between a terminal and the network. For example the Point-to-Point Protocol (PPP) [2] performs network access authentication before the Internet Protocol (IP) configuration phase. Data services in current cellular networks also implement network access authentication below the network layer. IEEE is specifying new authentication procedures for IEEE link layers such as IEEE 802.11 WLANs. These protocols are sometimes called user-to-network protocols, because a user device and a network element performing attendant functionality use the protocol.

Usually, the attendant uses another protocol to communicate with backend servers in order to authenticate the user, authorize the user for network access and account for the usage of resources. Examples of such

network-to-network Authentication Authorization and Accounting (AAA) protocols are Radius [3 - 5] and its successor Diameter [6], used in IP access networks with PPP.

Link-layer user-to-network protocols are currently not available for all access technologies such as the current Ethernet and Wireless LAN implementations. Implementing such link-layer protocols in an existing system is hard because it implies changes to low level software or even hardware. Because link-layer protocols are specific to a certain link layer, the functionality needs to be implemented separately for each link layer.

Mobile IPv4, specified in [7 - 12], includes access network authentication features implemented at the network layer. The visited network may deploy foreign agents, with which mobile nodes must register in order to get network access. The foreign agent may make use of an AAA protocol to authenticate and authorize the mobile node. However, it does not make sense to deploy Mobile IP just for network access authentication on hosts that do not need IP level mobility. Mobile IPv4 implementations are non-trivial to deploy on current operating systems. Mobile IP also requires the network to have home agents, and to use IP tunneling, which are unnecessary if the goal is to provide for network access authentication only.

The Network Access Authentication Protocol (NAAP) is a user-to-network client/server protocol, which performs authentication and access authorization over IP and hence decouples authentication from the access technology and the link layer. NAAP can be used over any link layer, including WLANs, Ethernet, ADSL etc. *Figure 18* shows the architecture of the authentication network. NAAP protocol entities, the NAAP Client and the NAAP Agent, reside in the user terminal and the edge router, respectively. The edge router communicates with the user's AAA server using a network-to-network AAA protocol. Although the NAAP Agent usually resides in the edge router, it could be placed deeper on the network as well.

The NAAP Client software can easily be implemented as add-on software on current operating systems without changing network cards or drivers. It is considerably easier to implement and deploy than Mobile IP client software, which typically requires system level programming. A NAAP Agent is implemented on a router, where NAAP can be used in conjunction with a network-to-network AAA protocol. Because access enforcement (packet filtering) is implemented at the network layer with access control lists, NAAP allows the access network to provide free local services that are available without any authentication. For example a hotel network may allow free access to the hotel Intranet but require authentication for global Internet access.

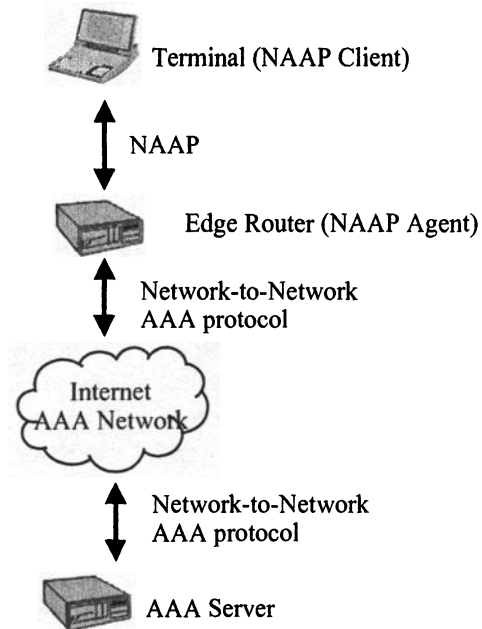


Figure 18. Authentication network

When a terminal enters an access network, it obtains an IP address for example with the Dynamic Host Configuration Protocol (DHCP). NAAP is independent from the method of obtaining the IP address. A terminal visiting a NAAP-enabled access network must register with the local NAAP Agent in order to get network access. After obtaining the address, the terminal uses the NAAP Agent discovery functionality to detect if the local network requires NAAP registration and to learn the IP address of the local NAAP Agent. The terminal then registers with the NAAP Agent and is finally granted network access.

Many NAAP features are modeled after Mobile IPv4. Agent discovery, protocol message formats, message integrity protection and replay protection resemble the equivalents in Mobile IP.

NAAP re-uses the PPP Extensible Authentication Protocol (EAP), discussed in Section 2 of this paper, in a novel way. Section 3 shows an overview on the NAAP authentication procedure. Section 4 discusses local NAAP messaging between the NAAP Client and the NAAP Agent. NAAP security considerations are covered in Section 5, and the use of NAAP with Virtual Private Network software in Section 6. Finally, Section 7 contains conclusions.

2. EXTENSIBLE AUTHENTICATION PROTOCOL

In first versions of the Point-to-Point Protocol an implementation that wanted its peer to authenticate with some specific authentication protocol requested the use of that authentication protocol during link establishment phase. The PPP implementations in both ends of the connection were required to know the authentication protocol. If a new authentication protocol was designed, existing Network Access Servers (NAS's) had to be updated.

To overcome this problem, the Extensible Authentication Protocol [13] was designed as a new PPP authentication framework to enable a PPP client to authenticate with a back-end EAP server without the NAS needing to know the details of the authentication method. The NAS simply passes through EAP packets between the PPP client and the EAP server, until it gets a success or a failure indication from the EAP server. In principle the EAP server functionality can be co-located in the NAS but typically the EAP server is a separate entity with which the NAS exchanges EAP packets over Radius or Diameter. In brief, EAP specifies a framing for multi-roundtrip authentication protocols.

During the EAP authentication, the EAP server sends EAP Requests to the client. The EAP Type field in the EAP packet indicates what it being requested. The client sends an EAP Response of the requested type in reply to each EAP Request. The first EAP Request is usually of the type Identity. It can be issued by the NAS even if a separate EAP server is used. The client's response to this request contains the client's identity. User identities of the Network Access Identifier (NAI) format [14], which contains information to find the correct AAA server, are used in roaming environment.

The Identity Request is followed by one or more EAP Requests for authentication information. Eventually, the EAP server ends the authentication with an EAP Success or an EAP Failure packet.

The EAP Type field in the EAP packet distinguishes different authentication protocols. EAP encapsulations have been specified or are being specified for a number of protocols, including MD5 Challenge (CHAP), one-time passwords, Transport Layer Security (TLS), Secure Remote Password (SPR), Generic Security Services (GSS) API, GSM authentication and UMTS authentication.

In PPP, the client never re-transmits an EAP packet on a timer. It is the responsibility of the Network Access Server to retransmit the EAP Request if it receives no response from the client.

NAAP re-uses the PPP EAP framework and the EAP AAA message formats. This makes it possible to use existing AAA servers, AAA protocols and EAP types with minimal modifications.

Some EAP types are capable of distributing a session key as part of the authentication procedure. TLS, SRP, GSM and UMTS authentication protocols are examples of such EAP types. Because EAP was originally designed for physically secure channels, NAAP is not just a simple EAP over UDP protocol. NAAP includes additional functionality to protect the NAAP messages against various attacks that may take place on a wireless shared-medium network. For this functionality, NAAP makes use of session keys distributed by the EAP type. Therefore, NAAP requires that the EAP type used must distribute a session key.

3. OVERVIEW ON NAAP AUTHENTICATION

A NAAP Client uses NAAP agent discovery to learn of local NAAP Agents. Agent discovery resembles Mobile IP agent discovery. It consists of two NAAP messages, the NAAP Solicitation, which a NAAP Client may multicast or broadcast, and the NAAP Advertisement, sent by a NAAP Agent to a NAAP Client in response to client's NAAP solicitation. Analogously to Mobile IP, the NAAP Advertisement includes the IP address of the NAAP Agent and the minimum and maximum registration lifetimes supported.

If the client discovers that the current network employs a NAAP Agent, the client needs to register with the agent in order to get network access. A NAAP registration is the exchange of NAAP Request and NAAP Reply messages. The NAAP Reply message contains a Code field, which tells the outcome of the registration. For example, the Code value SUCCESSFUL says that the terminal was registered successfully. During the EAP authentication, the NAAP Client and the NAAP Agent need to exchange several NAAP Requests and NAAP Replies before the client is successfully authenticated. The NAAP Replies that require the NAAP Client to send more NAAP Requests use the Code value AUTHENTICATION_INCOMPLETE.

Figure 19 illustrates the NAAP authentication when the terminal enters an access network. First the terminal discovers the NAAP Agent by sending a NAAP Solicitation message and receiving a NAAP Advertisement message from the agent.

The NAAP Client starts the EAP authentication after receiving the NAAP Advertisement. The first NAAP Request the terminal sends includes an EAP extension, which contains the EAP Response packet of the type

Identity. As usual in EAP, the EAP-Response/Identity packet contains the NAAP Client's AAA identity, the Network Access Identifier (NAI). The EAP-Request/Identity packet is not used in NAAP, but the EAP authentication always begins with the client's NAAP Request that includes EAP-Response/Identity.

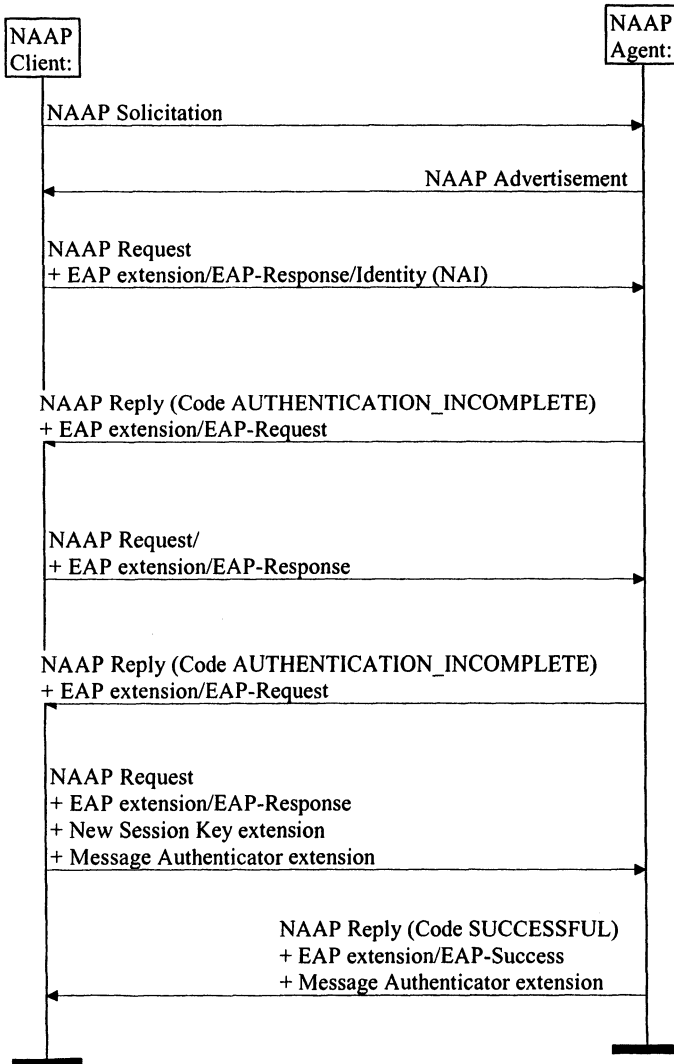


Figure 19. Successful NAAP authentication

Following the NAAP Request that contains EAP Response/Identity, the NAAP Agent sends the terminal several EAP Requests (in NAAP Reply

packets) and the terminal replies with NAAP Requests that contain the corresponding EAP Responses. These EAP packets contain the actual authentication information.

The EAP authentication results in that a new session key is generated and becomes shared by the NAAP Client and the NAAP Agent. The NAAP Client derives the key using the information in the EAP Request packets, and the NAAP Agent receives the key from the AAA server.

The key is associated with a Security Parameter Index (SPI) number that identifies the key. It is useful in distinguishing an old key from a new key that has been derived during a re-authentication. The NAAP Client chooses the SPI value and transmits it to the NAAP Agent in the last NAAP Request shown in *Figure 19*. The last NAAP Request contains the last EAP Response. This NAAP Request message must include the Authenticator extension (Section 5) calculated with the new session key. The NAAP Request also includes the New Session Key extension, which specifies how the new session key will be used. For example it contains the SPI value picked up by the client. In a successful case, the AAA response the NAAP Agent gets from the Authentication Server concerning this message includes the EAP Success packet and the new session key. The NAAP Agent verifies the message authentication code (MAC) in the Authenticator extension and replies with the NAAP Reply containing the EAP Success packet from the AAA response.

If the authentication fails or if an error occurs on the EAP server, the NAAP Agent sends the NAAP Client an EAP Failure packet in a NAAP Reply with the Code FAILURE. The AAA message from the AAA server may contain an attribute that contains an error message to the user. The NAAP Agent may include the error message as a Reply-Message extension in the NAAP Reply that it sends to the NAAP Client. *Figure 20* shows an example of an unsuccessful authentication.

A design principle in NAAP is "smart terminal, dumb network". By keeping the NAAP Agent workload per each client as small as possible the agent is expected to be able to serve more clients. Hence, the NAAP Client is responsible for retransmissions and soft state refreshing. The retransmissions in NAAP are different from EAP over PPP, where the Network Access Server is responsible for retransmissions.

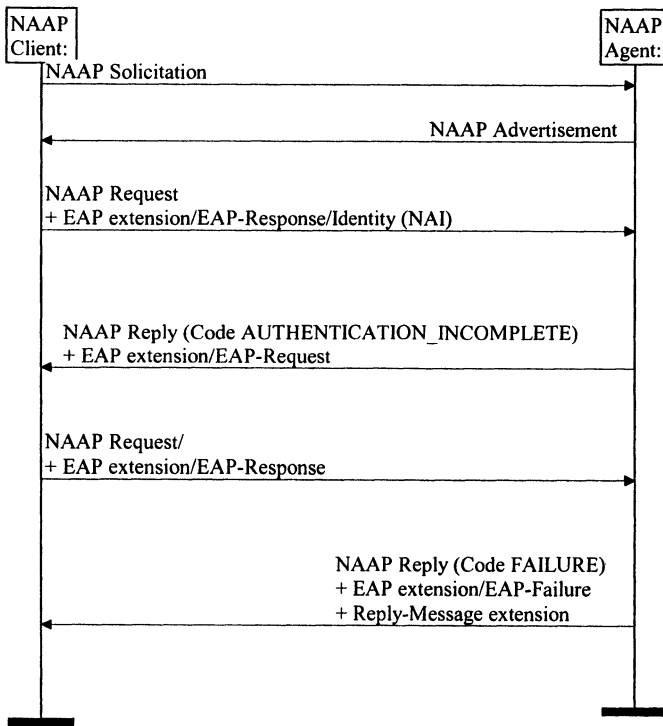


Figure 20. Unsuccessful NAAP authentication

4. LOCAL REGISTRATIONS AND DEREGISTRATION

After the full AAA registration, the NAAP Client and the NAAP Agent may use local NAAP messaging to extend the registration lifetime, to disconnect and to enable the agent to detect silently disconnected clients.

The NAAP Client maintains two timers: a preferably short-term registration lifetime timer, which makes sure that the terminal periodically refreshes its NAAP registration, and a typically long-term EAP authentication timer, which makes sure that the full EAP authentication and key exchange is executed periodically. As explained in Section 6, local NAAP messages cannot always be used. In this case, the registration lifetime timer is also a long-term timer and the client does not need to frequently refresh its NAAP registration.

The local NAAP registration, illustrated in *Figure 21*, occurs locally between the terminal and the agent to refresh the NAAP registration. When local registrations are used, a short registration lifetime, for example 60

seconds, is chosen. If the terminal does not re-register during the registration lifetime, the agent can deduce that the terminal has left the access network. Knowing when the terminal leaves is important if the network accounts for the connection time.

The NAAP Client may explicitly close the NAAP session by sending a local NAAP Request with the requested lifetime field set to zero. The agent responds to these deregistration requests with a NAAP Reply. The formats of the de-registration NAAP Request and the corresponding NAAP Reply are similar to the local re-registration messages.

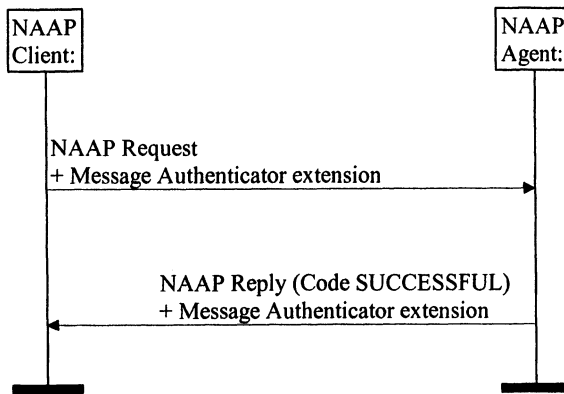


Figure 21. A local NAAP registration

5. PROTECTION AGAINST MESSAGE TAMPERING AND REPLAY

The EAP type used in NAAP must be strong enough so that the EAP packets can be transmitted in the clear during EAP authentication. The first NAAP messages do not contain any cryptographic protection. Once a session key has been distributed, the Authenticator extension can be included in a NAAP Request or a NAAP Reply message to protect the message against tampering. The Authenticator extension contains a Message Authentication Code (MAC) that is calculated over the NAAP message using the key that the NAAP Client and the NAAP Agent share as a result of EAP authentication. The MAC is calculated as a HMAC keyed hash [15] using the MD5 hash function [16]. Besides the MAC, the Authenticator extension contains the Security Parameter Index that identifies the session key.

The Authenticator extension must be included in two cases:

1) In an EAP authentication and re-authentication, the NAAP Request that contains the last EAP Response must include the New Session Key extension and the Authenticator extension calculated using the new key. The corresponding NAAP Reply must also be protected with an Authenticator extension calculated using the new key. The purpose of the message authentication in this case is to protect against a man-in-the-middle attacker from changing the NAAP fields outside the EAP packet. For example, the NAAP Agent ascertains that the IP address of the NAAP Client is correct.

2) Local NAAP Requests and Replies that do not contain an EAP extension must include the Authenticator extension. This includes the local registrations that are performed to extend the lifetime of the NAAP registration, and deregistrations that are performed to disconnect. This prevents a malicious node from hijacking the client's session after the client has left the network. It is also impossible to mount a denial of service attack by simply sending a deregistration request on behalf of another terminal.

Some error messages in NAAP do not include the Authenticator extension. If the authentication fails, no key is distributed and hence the NAAP packet that contains EAP Failure cannot be protected. Therefore it is easy to mount a denial of service attack by injecting a false EAP Failure packet to the client. The NAAP Client protects against these attacks by accepting unauthenticated error messages only after the retransmission timer has been expired. If a correct message is received before the timer expires, the false error message is ignored.

Another way to handle these cases would be to only send error messages if they can be authenticated and let timer expiration take care of other error cases. In NAAP, unauthenticated error messages are used, however, because they can contain a potentially useful error message. For example, the user may be told that the visited network does not have a roaming agreement with the user's home service provider, or that his connections are barred because of unpaid bills.

In order to prevent an attacker from replaying old NAAP registration messages, NAAP protects against replay attacks with a timestamp in the NAAP Request and NAAP Reply. The replay protection is required for the local NAAP registrations that do not involve the AAA network, because the authentication of these messages is only based on the message authentication code in the Authenticator extension.

Usually in protocols, protection against replay attacks is based on either timestamps or the exchange of random values. In the latter alternative, one of the parties sends a fresh random number to the other party, which replies with a message that include a value calculated using the random number and a shared key. In NAAP, replay protection is based on timestamps and it is similar to Mobile IP timestamp replay protection. The basic principle is that

the NAAP Client inserts the current time of day in the NAAP Request message, and the NAAP Agent receiving the message checks that this timestamp is sufficiently close to its own time of day. Because the timestamp is included in the calculation of the Authenticator extension, an attacker cannot make an old message current just by modifying the timestamp. Obviously the two nodes must have adequately synchronized time-of-day clocks. The timestamp difference tolerance is a configurable parameter on the agent. The NAAP Request and NAAP Reply messages can be used to synchronize the clock on the mobile terminal. Time synchronization NAAP messages are always protected against tampering with the Authenticator extension, so that an attacker cannot fool the client to adjust its clock to wrong time. This could enable the attacker to obtain NAAP Requests with timestamp values that are in the future, and then use these requests to hijack the client's session when the timestamps become current.

Upon receipt of a NAAP Request with the Authenticator extension, the NAAP Agent checks the timestamp for validity. In order to be valid, the timestamp must be close enough to the NAAP Agent's time of day clock and the timestamp must be greater than all previously accepted timestamps for the requesting NAAP Client. If the timestamp is not valid, the NAAP Agent includes a timestamp from its own time of day. The NAAP Agent rejects the registration by returning a Code that indicates incorrect timestamp in the NAAP Reply.

The NAAP Agent does not check the timestamp for validity if the NAAP Request does not include the Authenticator extension. In these cases, the field is only used for matching NAAP Replies to Requests.

If a NAAP Client retransmits a NAAP Request, it uses a new timestamp value containing the current time of day.

6. NAAP AND VPN SOFTWARE

Users of wireless LANs often use a Virtual Private Network solution to provide for user data confidentiality. When a VPN connection is active, all IP traffic is typically tunneled to a VPN gateway. The VPN client software typically includes a configurable security policy with which the user can specify the processing required for different types of traffic. Ideally, the policy file should allow local exchange of DHCP and NAAP packets so that these protocols are not tunneled to the VPN gateway.

Sometimes the user is not allowed or willing to modify the VPN policy. We discovered during the system testing that if the VPN software tunnels all traffic to a remote VPN gateway, NAAP packets will not reach the NAAP Agent after a VPN connection has been activated. In these cases, DHCP

leases or NAAP registrations cannot be renewed, which means that communications will fail when the NAAP registration or the DHCP lease expires.

If local NAAP messages cannot be used, NAAP terminal presence detection based on NAAP local registrations is not available. To support this scenario, NAAP protocol was enhanced so that the agent may advertise a long maximum registration lifetime in the NAAP Advertisements. The NAAP Client chooses a registration lifetime between the advertised minimum and maximum, and supplies it in the lifetime field of the NAAP Request packet. If the client chooses to use a short registration lifetime, the agent can rely on the NAAP registrations for terminal presence detection. However, if the terminal chooses to use a long registration lifetime, the agent may try to detect inactive terminals for example with an ARP request or an ICMP echo request (ping) after the terminal has been idle for some time. If the terminal does not reply to a couple of successive ARP request or ICMP echo request, the agent may deduce that the terminal has left the access network. Relying on ARP or echo replies makes it possible to hijack the session of a terminal that has left the network, if the hijacker assumes the terminal's IP and MAC addresses and continues using the network on the original user's account. Therefore NAAP Client should use local registration whenever possible.

7. CONCLUSIONS

The NAAP protocol described in this paper has been successfully implemented in the Nokia Operator Wireless LAN product. The author is participating in the standardization of an IP layer user-to-network authentication protocol (User Registration Protocol, URP), which is currently beginning in the Internet Engineering Task Force (IETF).

NAAP is a rather straightforward registration protocol and therefore easy to implement even on current operating systems. However, during the system testing, some problems in the co-existence of NAAP and Virtual Private Networking software were discovered, and the NAAP protocol was enhanced to work around these problems.

Currently, the NAAP Agent implementation enforces the access control by filtering out the packets of unauthenticated clients based on IP and link-layer addresses. The keying material distributed in NAAP could easily be used for packet security between the client machine and the router that contains the NAAP Agent. For example, there could be an IP security (IPsec) tunnel between the client machine and the router. Packet

authentication and encryption would provide for better access control and user data confidentiality.

ACKNOWLEDGEMENTS

Thanks to Antti Kuikka, Jukka Latva, Lassi Lehtinen, Jyri Rinnemaa, N. Asokan and all others who have contributed in the development of the NAAP protocol.

REFERENCES

- [1] Nokia Operator wireless LAN WWW page, http://www.nokia.com/networks/wireless_lan/index.html, August 2001
- [2] W. Simpson, Editor, "The Point-to-Point Protocol (PPP)", IETF Request For Comments 1661, July 1994
- [3] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", IETF Request For Comments 2865, June 2000.
- [4] C. Rigney, "RADIUS Accounting", IETF Request For Comments 2866, June 2000.
- [5] C. Rigney, W. Willats, P. Calhoun, "RADIUS Extensions", IETF Request For Comments 2869, June 2000.
- [6] The Diameter protocol is being specified in the IETF and it is currently work in progress. Current versions of the Diameter documents are available at <http://www.ietf.org/html.charters/aaa-charter.html> (October 2001)
- [7] C. Perkins, "IP Mobility Support", Internet Engineering Task Force, Request For Comments 2002, 1996.
- [8] C. Perkins, "IP Encapsulation within IP", Internet Engineering Task Force, Request For Comments 2003, 1996
- [9] C. Perkins, "Minimal Encapsulation within IP", Internet Engineering Task Force, Request For Comments 2004, 1996
- [10] J. Solomon, "Applicability Statement for IP Mobility Support", Internet Engineering Task Force, Request For Comments 2005, 1996
- [11] D. Cong, M. Hamlen, C. Perkins, "The Definitions of Managed Objects for IP Mobility Support using SMIPv2", Internet Engineering Task Force, Request For Comments 2006, 1996
- [12] G. Montenegro, "Reverse Tunneling for Mobile IP", Internet Engineering Task Force, Request For Comments 2344, 1998

- [13] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", IETF Request for Comments: 2284, March 1998
- [14] B. Aboba and M. Beadles, "The Network Access Identifier", Internet Engineering Task Force, Request For Comments 2486, January 1999.
- [15] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Internet Engineering Task Force, Request For Comments 2104, February 1997
- [16] R. Rivest, "The MD5 Message-Digest Algorithm", Internet Engineering Task Force, Request For Comments 1321, April

Test Bed for Plain C/C++ Protocol Implementations

Henry Haverinen and Lassi Lehtinen

Nokia Mobile Phones

Abstract This paper presents a protocol software test bed, which was developed for the testing of a Network Access Authentication Protocol (NAAP) implementation. We demonstrate how modular software design can help in protocol testing and we present a simple software interface that enables flexible protocol testing of protocol implementations. Besides normal use scenarios, the test bed allows automatic testing of various error cases, such as dropped, delayed, duplicated and modified packets.

The presented protocol software test bed was successfully used in the testing of a NAAP implementation, and several normally difficult-to-find errors were fixed during the testing. The same test bed design was also used to test a Mobile IP implementation, and it can be applied to other implementations as well.

Keywords: Protocol testing, modular software design

1. INTRODUCTION

Formal Description Techniques (FDTs) and high-level protocol implementation tools that are based on FDTs often include simulation and testing features. SDL [1] is an example of such a language. There also are C and C++ protocol implementation libraries that can assist in the testing or simulation of implementations, such as CVOPS [2] and x-kernel [3]. However, many communication protocols are still implemented without implementation frameworks or libraries, in plain C and C++. This paper describes a protocol software design, which can be used in the testing of such plain C/C++ protocol implementations. The goals of the software design are to make the protocol implementation portable to various platforms and to allow the testing of the implementation in different error

scenarios. Thanks to its simplicity, the underlying idea of the test bed implementation could be used to test almost any C/C++ protocol implementation, in many cases even after the implementation has been completed. This is contrary to the testing support of high-level protocol implementation frameworks, which only work for protocol software that has been implemented using the particular frameworks.

The protocol test bed was developed for the testing of a Network Access Authentication Protocol (NAAP) [4] implementation. NAAP is a client/server protocol that runs over the User Datagram Protocol (UDP).

An overview of the relevant parts of the NAAP software architecture is given in Section 2. The architecture and the operation of the test bed is described in Section 3. Section 4 presents the results of NAAP testing. The generality of the test bed is discussed in Section 5. Section 6 contains conclusions.

2. NAAP IMPLEMENTATION

The software architecture of the NAAP implementation, shown in Figure 1, uses a well-known principle in modular software design: the platform-independent parts have been separated from the platform-specific parts with a clearly defined interface. This principle is currently considered one of the advantages of object-oriented design, but software has been structured for portability already in the early 1970's [5].

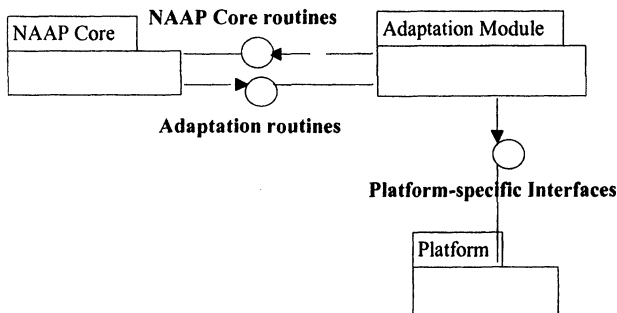


Figure 1. Software architecture of the NAAP implementation

The NAAP Core module contains the platform-independent parts of the protocol implementation, while all the system dependent code is in a platform-specific adaptation module. This division makes it easy to port the

implementation to new platforms. One only needs to write an adaptation module for the new platform.

The NAAP Core is a collection of functions and data. It does not own any threads of execution. The adaptation module controls the operation of the NAAP Core by calling the NAAP Core routines in a suitable order. The NAAP Core routines are non-blocking; they complete without waiting for any external events to occur. All waiting needs to be implemented in the adaptation module.

Sending and receiving data are examples of system-specific tasks that are needed in all protocol implementations. For instance, when the protocol needs to send a packet to the network, the NAAP Core calls the `am_sendto()` routine in the adaptation module, which then invokes a platform-specific routine to send the packet, as illustrated in *Figure* When the conventional socket interface is used, the socket operation to send a packet is `sendto()`.

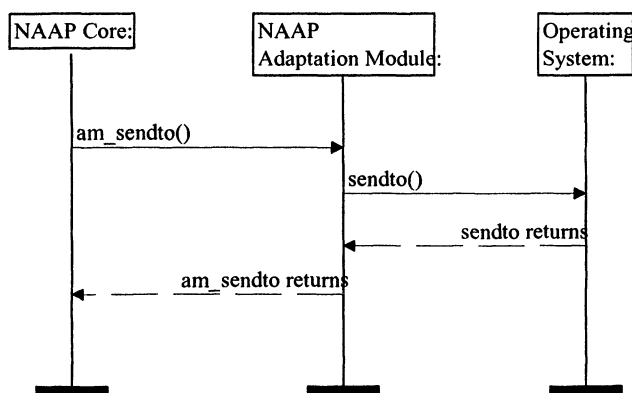


Figure 2. Packet sends in the NAAP implementation

Figure shows how packet receives are implemented. Because the implementation is run by a single thread of execution, non-blocking receives are used. First, the operating system signals the NAAP process that data is available in a socket. On the Unix adaptation, the NAAP adaptation module uses the `select()` system call to wait for multiple waitable objects. When the `select` call returns and indicates that data has been received to one of the sockets, the NAAP adaptation module passes the indication on to the NAAP Core by calling the NAAP Core `nc_handle_socket()` operation. The NAAP Core has allocated a data buffer, which it passes to the `am_recvfrom()` adaptation routine. This adaptation routine calls the

system call to receive the data to NAAP Core's buffer. The adaptation module does not need to manage buffers for NAAP packets.

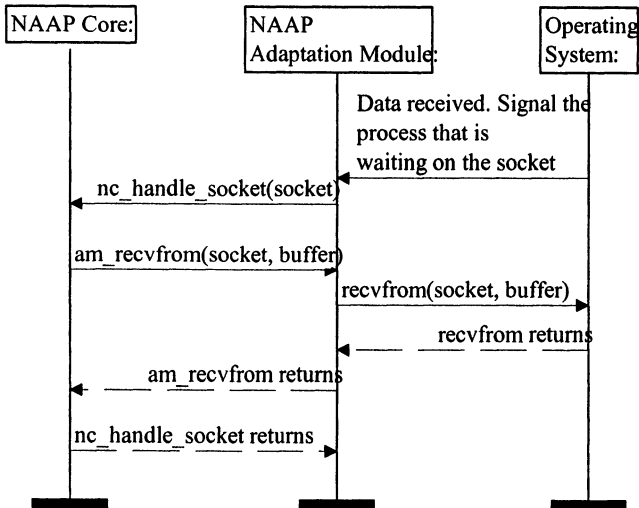


Figure 3. Packet receives in the NAAP implementation

Other examples of system-dependent functionality that is often needed in protocol implementations are memory operations, getting the current time of day for timestamps, and the scheduling of timeouts. In the NAAP implementation, the timeouts are implemented as an event queue in the NAAP Core. The event queue is a list of event descriptors, sorted in increasing order of when the event is to occur. There is a NAAP Core routine that the adaptation module can call to learn the timeout to the first event in the queue. It is the responsibility of the adaptation module to call the NAAP Core routine which processes the expired events after this timeout.

The NAAP Core adaptation interface is also to control the protocol. For example, all the commands the user can give, such as "connect", "cancel connect", "disconnect", are invoked by calling a NAAP Core routine. The NAAP Core gives indications of various events by calling an adaptation routine. There are indications for successful connection, unsuccessful connection, expired connection and a notification for all state transitions in the protocol so that progress bars can be updated and so on.

3. TEST BED IMPLEMENTATION

The implemented NAAP test bed is a special adaptation module that contains additional testing functionality. In other words, the test bed hooks up between the NAAP Core module and the actual adaptation module. The adaptation routine calls are routed through the test bed module, which can then generate various error conditions.

For example, to test the protocol implementation when a packet is dropped, the test bed implementation of `am_sendto()` returns a successful value without actually sending the packet to the network, as illustrated in *Figure*. To duplicate packets, the test bed implementation of `am_sendto()` calls the underlying actual adaptation module routine twice.

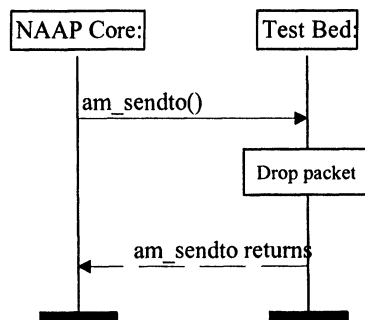


Figure 4. Testing of dropped sends

The testing of delayed sends is shown in Figure 5. For simplicity, the test bed and the actual adaptation module are shown as a single entity in the figures of this section. To delay a packet, the test bed `am_sendto()` routine schedules a timer to send the packet later. The test bed can only refer to NAAP Core's data buffer and other parameters given to the `am_sendto()` routine, such as the destination IP address and port, while in the context of `am_sendto()`. Therefore, the test bed needs to copy the data buffer and other parameters to a send descriptor it has allocated itself. While the timeout is pending, the adaptation module processes all events as usual. When the timeout elapses, it calls the underlying operating system routine to send the packet using the information in the send descriptor. After sending the packet, the test bed frees the send descriptor. Our implementation of the test bed delays packets using the same event queue that is used for NAAP timeouts.

The test bed `am_sendto()` routine can also truncate or otherwise modify the packet. Introducing random errors in the packet is trivial, but the

NAAP test bed is also able to parse NAAP packets and take actions that depend on the contents of the NAAP message. For example, the test bed is able to modify a given field, or insert a given extension to the packet.

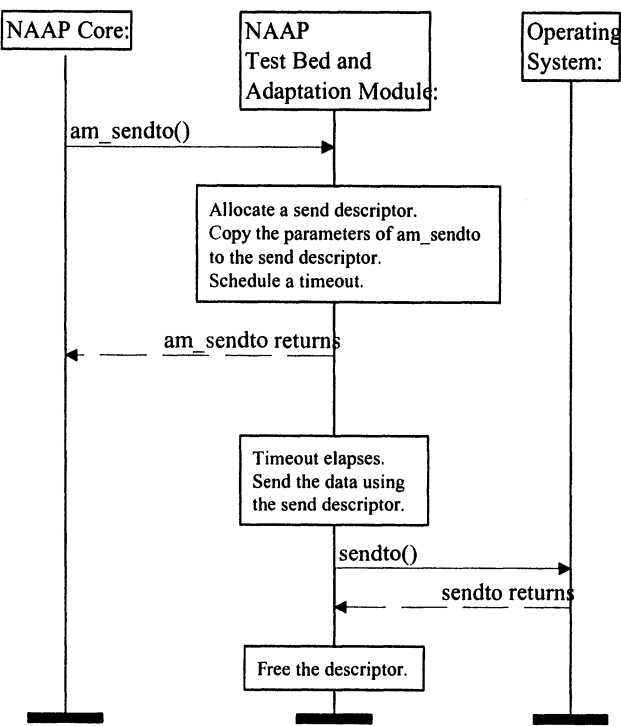


Figure 5. Testing delayed sends

Because we were testing both the NAAP client and the NAAP server entities, it was sufficient for us to implement testing functionality to sends only and not hook up to receives. Testing the sends in the client and in the server covers the NAAP packets both ways. However, if it was necessary, similar operations could be implemented when receiving packets too. In the NAAP implementation, the adaptation module is responsible for detecting when data is available to be read from sockets. Hence it is easy to layer test functionality for received data too. For a simple example, to drop a received packet, the adaptation module can simply fail to call the `nc_handle_socket()` routine, which is used to indicate the NAAP Core of received data, and ignore the received data.

Modifying received packets can be implemented by reading the received data to a buffer allocated by the test bed, as shown in Figure 6. When the

operating system indicates that data has been received, the test bed allocates a receive descriptor. The test bed then reads the received data to a buffer that is included in the receive descriptor and saves all the related information of the received packet, such as the source IP address, to the receive descriptor. Before indicating the packet to NAAP Core with `nc_handle_socket()`, the test bed modifies the received data. When the NAAP Core calls `am_recvfrom()` to read the received data, the test bed does not pass the buffer to the underlying operating system, as the actual adaptation module does, but the test bed copies the contents of the modified buffer from the receive descriptor to NAAP Core's buffer, and also supplies the related information it has stored in the descriptor.

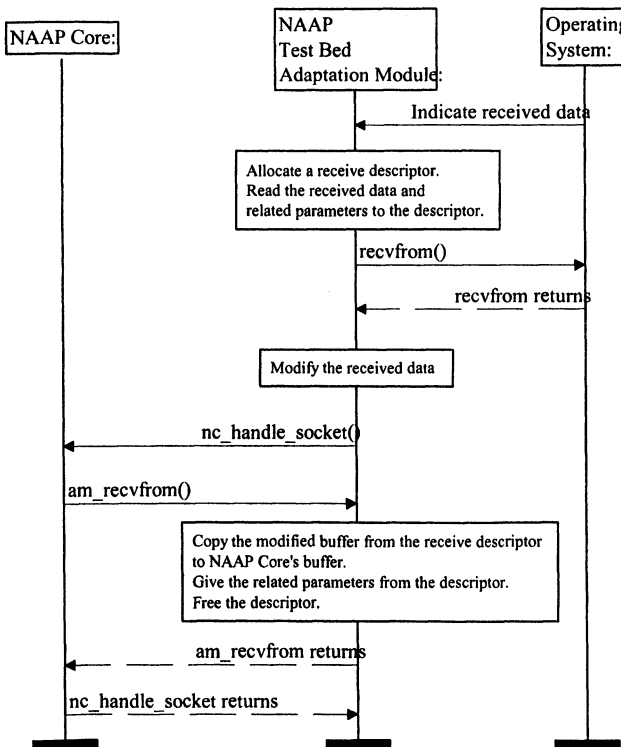


Figure 6. Modifying received packets in the test bed

Modifying data could also be implemented in the `am_recvfrom()` function of the test bed by reading the data to NAAP Core's buffer and then modifying it. However, using a receive descriptor is a more general way of implementing test functionality for receives as it also enables the testing of delayed and duplicated receives.

Duplicated receives can be implemented with a similar mechanism. The test bed needs to copy the received data in a buffer it has allocated, because the underlying `recvfrom()` routine would only give the data once. After reading the data to a buffer, the test bed can indicate the data to the NAAP Core twice, and copy it to the NAAP Core's buffer from its own buffer.

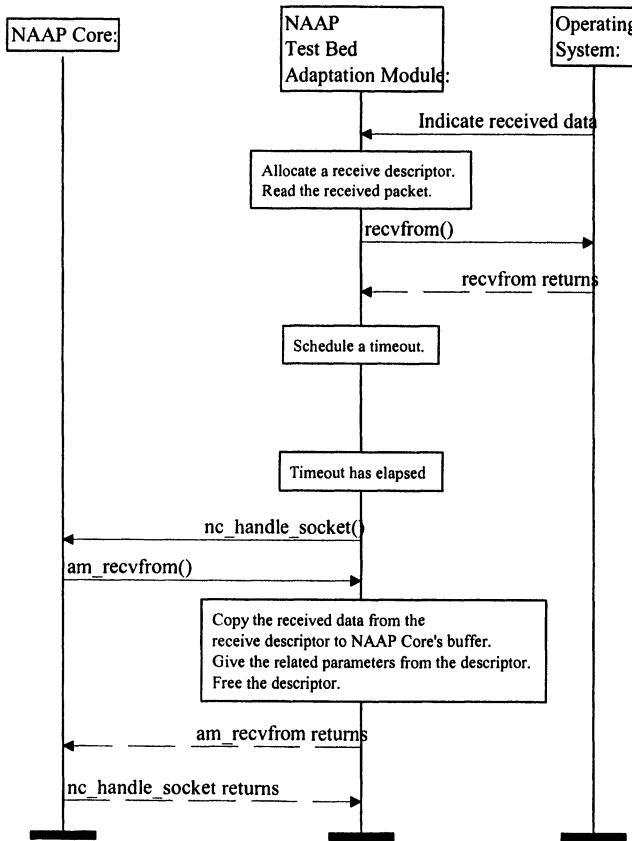


Figure 7. Testing delayed receives

The testing of delayed receives is illustrated in *Figure 7*. To delay a received packet, the test bed schedules a timeout and indicates the packet to the NAAP Core only after the timeout has elapsed. The test bed implementation for delayed packets is also implemented by copying the received data and the related parameters to a receive descriptor when the operating system indicates that data has been received. If the test bed didn't copy the data but left the data in the operating system buffers, the packets would still be received in the correct order, which is not the desired

operation when testing delayed packets. After copying the received packet to a receive descriptor, the test bed then schedules a timeout after which NAAP Core is indicated. While the timeout is pending, the adaptation module processes all events as usual. Other packets can be received without delaying them. When the timeout eventually elapses, the test bed indicates the NAAP Core, and the received data from the receive descriptor is copied to NAAP Core's buffer.

Our test bed implementation includes a configuration file with which different test cases can be specified. There are different conditions when to employ any of the above mentioned error cases. For example, the configuration file can include an entry that specifies the probability of dropping a certain NAAP packet.

The NAAP test bed makes use of the notifications given by the NAAP Core. By hooking up to these notifications, the test bed is able to monitor the state of the protocol. This is exploited for example by employing test case after a certain state transition has occurred. The testing of the "cancel connect" feature is a good example of this. Because the user can click the Cancel button in any phase of the connection establishment, it is hard to test this feature manually. We had a test case for cancelling the connection in each intermediate state of the connection establishment, based on notifications from the NAAP Core. The notifications also enable the test bed to automatically tell if a test case has passed or failed. This makes it possible to prepare a set of test cases, which can be automatically performed when the protocol implementation has been changed.

4. NAAP TESTING IN PRACTISE

In total, the NAAP testing plan defines 88 test cases that make use of the test bed. Six cases were related to the basic use scenarios of the protocol. In 15 cases, random errors were introduced in different types of packets. There were 15 cases of truncating packets, 11 cases for dropping packets and 15 cases for duplicated packets. Certain fields of the packets were modified in 19 cases. In addition, there were 7 cases to test the cancel connection feature, already discussed in the previous section.

As usual in software development, many bugs and anomalies in the NAAP implementation were found and fixed during the implementation in informal "tryouts" performed by the programmer. The actual testing begun only after the programmer had tried out the new build by running the most typical use cases and fixed any problems encountered. These bugs were not counted, but they numbered in hundreds. In the actual testing that was performed using the test bed, circa 50 bugs were found and fixed.

In six bugs, the protocol implementation did not check for the validity of a field in a protocol message as required in the protocol specification. The implementation accepted a message, which had been modified by the test bed to be invalid and thereby should have been silently ignored. There were two bugs, which caused the implementation to crash when the test bed set a protocol field to an invalid value.

In NAAP, some of the messages are required to include the Authenticator extension, which provides for message integrity protection against tampering. The specification lists two cases when exactly one Authenticator extension must be present. Three different bugs were found by removing the mandatory Authenticator extension or inserting an extra Authenticator extension in the test bed. The protocol implementation accepted the messages with invalid or missing Authenticator extensions.

One category of the bugs was related to message retransmissions. When a packet was lost, there were cases when the sender did retransmit the packet as specified, but the recipient did not accept the retransmissions. In some of the cases, a lost packet caused the protocol to enter a wrong state or to mischedule retransmission timers.

About half of the bugs were individual cases, which cannot be categorized with other bugs. For example, there were only individual errors caused by delayed or duplicated packets.

5. GENERALITY OF THE TEST BED

To prove that the concept of the test bed implementation can be generalized to other protocol implementations, we tested an earlier Mobile IP implementation, described in [6] with a version of the test bed. Thanks to the modularity of the Mobile IP implementation, it was easy to hook up the test bed to the software. Because we had used the same software interface for packet operations as in the NAAP implementation, we were able to plug in the test bed without changing the platform-independent part of the Mobile IP implementation.

We could not re-use the portions of the test bed that tested NAAP-specific functionality or hooked up to NAAP specific notifications. Obviously, writing tests that are aware of Mobile IP features requires Mobile IP specific work.

The test bed could be used to test other modular protocol implementations with small modifications. In most operating systems, the socket interfaces are descendants of the Berkeley Software Distribution (BSD) socket interface and hence the routines for sending and receiving data are very similar. If the original designer of the software has used wrapper

functions around the socket operations, it should be easy to hook up the protocol test bed.

Some protocol implementations do not separate the platform-specific parts in a different place but call socket operations directly. If the socket calls are all over the code, then preprocessor directives can be used to hook up a test bed. For example, there can be a preprocessor definition that replaces calls to the socket function `sendto()` with calls to `testbed_sendto()`. Some protocol implementations that have multiple threads or processes use blocking receives and call the `recv()` or `recvfrom()` routine without waiting for any separate indications first. Simple preprocessor definitions are sufficient to implement test functionality for receives in these cases.

Testing non-blocking receives requires more work. It is hard to come up with a general purpose preprocessor directive to take care of the `select()` system routine that is used for waiting for events to occur in sockets or file descriptors. Fortunately, there usually is only one or two places in the source code where `select()` is used, so it should be quite easy to hook up a test bed by modifying these parts. If we want to test delayed, duplicated or dropped receives, then the test bed may need to use a separate thread which waits in `select()`, and wakes up the protocol implementation thread when it wishes to indicate an event to the protocol.

Alternatively to protocol test beds that are linked with the protocol implementation, test functionality can also be implemented lower in the protocol software stack. For example on Windows, we could implement test functionality in an intermediate network driver, which resides between the TCP/IP stack and network interface card drivers. This test bed would be able to test any protocol implementation, even a binary executable that cannot be modified. However, when the test bed is linked with the protocol implementation, it is more convenient to debug the software when an error is found. It would also be harder to automatically detect passed and failed test cases in a driver-level protocol test bed. In order to test high-level issues, it is better to hook to higher level in the protocol stack.

6. CONCLUSIONS

The protocol test bed design described in this paper has been successfully used in the testing of a NAAP implementation and a Mobile IP implementation. Several errors that would have been hard to find in black box testing were fixed and the test bed proved to be easy to use and flexible.

Our experience is that modular software design facilitates not only implementation and porting but also testing. It is advantageous to implement

packet sends and receives, as well as protocol control and notifications so that test functionality can easily be hooked up.

REFERENCES

- [1] "Specification and Description Language (SDL)", ITU-T Recommendation Z.100, November 1999
- [2] J. Harju, A. Karila, J. Kuittinen, J. Malka: "CVOPS, a tool for the implementation and testing of computer communications software", Technical Research Centre of Finland, Telecommunications Laboratory, 1986
- [3] N. Hutchinson, L. Peterson, "The x-Kernel: An Architecture for Implementing Network Protocols", IEEE Transactions on Software Engineering, Vol. 17, No.1, January 1991
- [4] H. Haverinen, "NAAP: A User-to-Network authentication Protocol", Proceedings of Smartnet 2002 Conference, April 2002
- [5] P.C. Poole, W.M. Waite, "Portability and Adaptability", Advanced Course on Software Engineering, Lecture notes in Economics and Mathematical Systems 81, Springer 1973
- [6] H. Haverinen, A. Kuikka, T. Määttänen, "A Portable Mobile IP Implementation", Proceedings of the IEEE Local Computer Networks 2000 Conference, November 2000