Guaranteeing Quality of Service for the Web using Application REQUested IP over ATM

Werner Almesberger, Eric Gauthier,
Jean-Yves Le Boudec, Philippe Oechslin *
Laboratoire de Reseaux de Communication,
Swiss Federal Institute of Technology,
Lausanne, Switzerland.

November 14, 1995

Abstract

When using the Web over the TCP/IP protocol suite of the Internet there is (yet) no possibility to guarantee bandwidth or delay for the transmission of a document. For the multimedia documents of the Web, transmission with guaranteed quality of service (QoS) is however very desirable. ATM networks which are being deployed now are one means of solving this problem. They offer high bandwidths and QoS negotiation at connection establishment. Current implementations of IP over ATM do not pass these advantages of the underlying network to the application level. Making the hypothesis that standard solutions involving RSVP and NHRP will not soon be available we present a service built on a dual stack (IP & ATM) approach which allows to transport IP streams over application requested ATM connections. We show how this service can be used to offer guaranteed QoS to Web users on ATM while preserving compatibility with users which can not benefit of ATM connections. Finally we show how this solution can even provide better QoS to Web users not directly connected to ATM.

1 Introduction

1.1 The problem

The problem we want to solve is to guarantee quality of network service to Web users. With the current implementation of the Internet the delay and throughput experienced when browsing the Web can vary greatly over time and location. These variations get more important as Web documents get richer in multimedia information. If an agent wants to display a multimedia stream while it is downloading it from a server, it will need the guarantee that the stream is delivered at least as fast as it is displayed. If the same agent wants to first

*contact author, *oechslin@di.epfl.ch*

download the multimedia document before visualizing it, it may still need the guarantee that the document will be received within a useful amount of time. If QoS sensitive documents are provided by commercial services, then the users will want the guarantee that they get a quality of service worth their money. This is also true for time critical documents like stock-exchange data.

Quality of network service is not actually a problem of the Web itself but much more of the network it is built upon. Therefore solutions are expected to be found in a better usage of available networks and services or in the development of new networks or services.

Solutions could be found using new IETF protocols like RSVP and NHRP (see Section 2.1 and 2.2). In this paper, however, we make the hypothesis that RSVP on ATM and NHRP will not soon be available and explore solutions using native ATM connections.

1.2 Background

The Internet Protocol (IP) is the network level protocol of the Internet. It follows the paradigms of *connectionless datagram forwarding* and *best effort* service. IP routers forward IP datagrams to intermediate routers until they eventually reach their destination. If the network is overloaded packets are dropped. These paradigms make it hard to control the QoS between two end users on the net. This is a well known problem and one solution for guaranteeing QoS is the recent RSVP protocol [4].

Asynchronous Transfer Mode (ATM) is the standard for Broadband Integrated Service Digital Networks (B-ISDN). It is a connection oriented network based on fast packet switching. Communications between ATM end-systems require that a connection be setup prior to communication. ATM connections have a high throughput and a low latency since ATM cells are switched in hardware. In addition, connection setup enables resource reservation. These features make ATM connections well suited for audio and video transmissions.

1.3 Current and Future Work

One example of real-time multimedia communication within the Web can be found in [12]. This paper describes an agent called Vosaic which allows to setup teleconferences through the Web. Interoperation between IP and ATM networks is being worked on at the IETF eg. in the ipatm work-group of the Internet area. An essay about how to transport HTTP directly over ATM can be found in [9]. At the Laboratoire de Réseaux de Communication we are currently working on the Web over ATM project [11] where we explore the interworking between IP and ATM, in regard to the Web as a main application. So far we have implemented IP over ATM on Linux [2]. We have chosen Linux because it gives us full access to the kernel and because the results can be distributed freely. The work presented here is a part of the Web over ATM project.

1.4 Structure of this document

In Section 2.1 we describe how IP currently works over ATM networks and how the standards are evolving. In Section 2.2 we explain how the RSVP protocol reserves resources in IP networks. Section 3 introduces our solution with a

dual stack approach while Section 4 discusses interoperability issues between our solution and non-ATM hosts. Section 5 is the conclusion of our paper.

2 The current situation

2.1 IP over ATM

The scalable bandwidth, the guaranteed QoS and its ability to integrated many different services make ATM a big competitor for future LANs, WANs and backbone networks. This has been recognized by the Internet Engineering Task Force (IETF) which has developed standards on how to overlay IP networks on ATM networks. For a good overview of the aspects of interworking with ATM see [1].

A first standard is the so-called *classical IP over ATM* defined mainly in RFC1577 [8]. In that scheme IP hosts are grouped in Logical IP Subnets (LIS) which are typically connected to a default IP router. The ATM network is treated much like a LAN and hosts within a LIS can connect through an address resolution protocol that maps IP addresses to ATM addresses. If a packet has to go to another host outside the LIS, it is sent to a router which forwards it. The advantage of this solution is that it works in the same manner as existing IP networks, hence the name. The disadvantage is that packets may be sent through a set of routers and ATM connections even if a direct ATM connection would be possible. Also, all the data flowing between two machines typically use the same ATM connection, making it impossible to request a QoS for one specific data stream.

Some relief is to be expected from the Next Hop Resolution Protocol (NHRP) [7] currently under development. With this protocol, address resolution requests are propagated to different servers in the network. The reply to a request gives the information needed to establish a direct connection to the destination host or, if the host is not on the ATM network, to the closest router on the edge of the ATM network. NHRP currently has the status of an Internet draft.

The different schemes for IP over ATM will allow to run the Web transparently over ATM networks. However they don't provide any way to reserve resources or to guarantee QoS. The IETF has a solution for resource reservation in IP networks as we will see in the next section.

2.2 Resource reservation schemes for the Internet

Having identified resource reservation as a key aspect for transmission of real-time multimedia information the IETF has developed the Resource reSerVation Protocol [4]. In RSVP, a receiver can request a resource reservation along the path between the source and the receiver. This reservation is subject to call acceptance control by all intermediate IP routers which support RSVP. Once a connection is established, the routers schedule the transmission of IP datagrams in function of the priority of the stream they belong to. RSVP has been designed to work in a multicast environment. This means that resource reservations can not only be made between two endpoints but within a multicast tree. RSVP is about to become an IETF standard.

Using RSVP, a certain QoS can be guaranteed since resources are reserved at each intermediate hop. However, the datagrams still travel hop by hop. In an ATM environment, it would be much more efficient to establish direct connections.

3 AREQUIPA: a dual stack solution

Coming back to our original problem (Section 1.1) it seems that a solution is going to emerge all by itself. Once the routers of the Internet will implement RSVP, Web agents and servers will be able to use this protocol to guarantee the QoS for the delivery of Web documents. If by that time NHRP is implemented in ATM parts of the Internet (LANs, WANs, backbones) then it will even be possible to bypass intermediate routers and to have direct connections between servers and agents. There are two drawbacks of this solution. The first is its complexity, since two new protocols need to be implemented on all the routers of the network. The second drawback is the time to implementation. RSVP, which is about to become a standard, only provides for reservation of resources. It does not, however, give an exact definition of what resources are nor how they can be monitored and policed. This is what the Integrated Service IP work-group of the IETF (intserv) currently works on. NHRP on its side is still a draft and different issues need to be resolved before it can be proposed as a standard. Thus a complete solution using RSVP and NHRP can not be implemented in the nearest future.

We make the hypothesis that we are in a network were RSVP and NHRP are not yet available (which may be true for quite some time). Thus we explore a solution which does not make use of these protocols.

3.1 The dual stack solution

We propose a solution where the applications use both an IP and an ATM stack to obtain direct ATM connections with guaranteed QoS. Indeed, when an agent on an ATM network contacts a server on the same ATM network, it would clearly be inefficient to ask an intermediate layer of IP routers to route documents from the server to the agent if the server can connect directly to the agent. Our solution to the problem thus consists of opening a direct ATM connection between the server and the agent whenever a document has to be transmitted with guaranteed QoS. Two informations are needed for this: a) the server and agent must know each other's ATM address, and b) the server must know whether a document is QoS-sensitive and what QoS must be requested. The first information can be included in the header of HTTP requests or responses while the second information can be put in the header of HTML documents. To be able to open a connection with guaranteed QoS, the server software must also be able to interact directly with an ATM protocol stack. The server thus uses both a TCP/IP and an ATM protocol stack, hence the name of our solution.

3.1.1 Availability of a dual stack

As of today, Web servers or agents connected to ATM networks typically use classical IP over ATM. To support IP over ATM, a host must implement all the

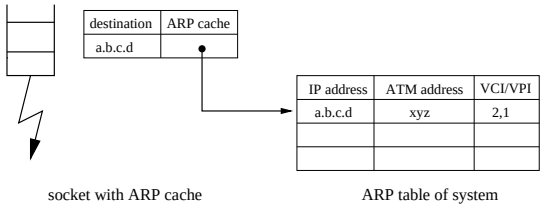

Figure 1: A socket with its ARP cache and the ARP table for IP over ATM

functionalities needed to establish ATM connections. Vendors of ATM Network Interface Cards (NICs) usually also provide a pure ATM API beside the standard IP API. This provides a way of establishing application requested ATM connections between servers and agents running on ATM networks. Furthermore, a standard ATM API is currently being defined by the ATM Forum [6]. Thus we conjecture that most machines running IP over ATM will also have some pure ATM API.

Our solution is made of two building blocks. First we propose a mechanism, called AREQUIPA, to open direct ATM connections between two hosts and to use it exclusively for one IP data flow identified by a pair of sockets (eg. a TCP connection or a UDP stream). Second, we propose a mechanism with which a web server and agent can negotiate and establish such connections.

3.2 Application REQUested IP over ATM (AREQUIPA)

The solution we propose makes use of the Address Resolution Protocol table (ARP table) used to map IP destinations to ATM connections. Each entry in this table typically contains a next hop IP address and a low level address. In the case of IP over ATM the low level address is the ATM address of the machine the data is being sent to (a router or the destination machine) together with the VCI and VPI used to send data to that machine . Each time an IP packet has to be sent, the IP address of the next hop is used to look up the corresponding low level address in the ARP table. To avoid making a lookup for every packet sent out, a pointer to the entry in the ARP table is usually cached in the descriptors of the sockets that send data to the corresponding destination (see Figure 1).

The table entries are created by the ATM ARP layer. Each time a packet has to be sent to a next hop which is not in the ARP table, an ATM connection is established to that next hop machine and the corresponding entry is added into the table. All traffic which has to go to that particular next hop then uses the same ATM connection.

We propose a solution which reuses most of the existing mechanisms. It requires an extra flag in the ARP table and in the socket descriptors, a field for ATM addresses in the socket descriptors as well as a slightly modified behavior of the ARP layer. The modified ARP table and socket descriptors are shown in Figure 2. The solution works as follows:

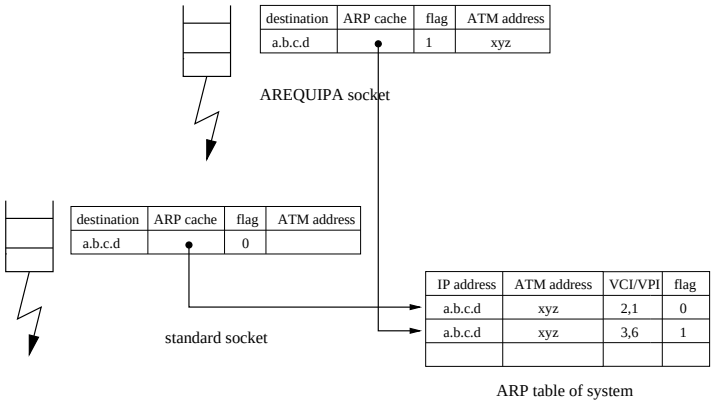

Figure 2: A standard socket and a socket with an application requested ATM connection in an AREQUIPA implementation.

The sender binds a socket to the IP destination as usual. Then it requests that the socket be bound to a dedicated ATM connection and indicates the ATM address of the destination and the required QoS. The system opens the requested ATM connection, creates the corresponding ARP table entry, marks it with the AREQUIPA flag and caches the ARP entry in the cache descriptor. After this operation, the socket can be used in the same way as any standard socket and the cached ARP entry makes sure that the data is sent over the requested ATM connection. The AREQUIPA flag in the ARP table indicates that an entry should not be used by sockets which haven't already cached it. This avoids that other data heading to the same IP destination uses the application requested connection. For reasons of symmetry with the receiver (see below), the ATM address of the destination is stored in the socket descriptor and the socket is marked with the AREQUIPA flag.

The receiver opens a socket for listening or reading, depending whether it expects UDP or TCP data. Then it requests that the socket be bound to an expected incoming ATM connection and indicates the ATM address of the source. The system marks the socket with the AREQUIPA flag, registers the ATM address in the socket descriptor and leaves the ARP cache empty. When new ATM connections are made to the system, they are entered in the ARP table as with classical IP over ATM. However, when data is delivered to a socket marked with the AREQUIPA flag and with an empty ARP cache, then the sender's ATM address (e.g. from the system ARP table) is matched against the one in the socket descriptor. If they are identical, then the entry in the system's ARP table is identified as the expected AREQUIPA connection. It is marked with the AREQUIPA flag and cached in the socket's ARP cache.

Discussion: the mechanism we propose for application requested ATM connections between pairs of sockets can be implemented by modifying the ARP

layer of the machines and the socket descriptors. Special care has to be given to operations that could invalidate cached ARP table entries. The IP layer can virtually be left unmodified. Two new system calls are needed, one for the setup on the sender side and one for the setup of the receiver. Once a connection is set up, the sockets behave like standard TCP or UDP sockets and can be used transparently. The only condition for the mechanism to work is that both parties first exchange their ATM addresses and decide which will be the sender and what port and protocol will be used on the receiver. In the case of the Web over ATM the necessary information can be exchanged in an HTTP request and response prior to the transfer of the QoS sensitive data. This is what we will discuss in the next section.

3.3 Using AREQUIPA in the World Wide Web

We will now discuss how Web agents and servers can make use of the above mechanism to request ATM connections for the transmission of QoS sensitive documents. The first question we have to answer is which of both parties will open the connection. We have chosen to give the server the responsibility of requesting the dedicated ATM connection. This seems logical since the server will be the sender of QoS sensitive documents. Also, other functions like access control and billing are usually done by the server. However the agent as a receiver should be aware of the requested QoS for the connection to know whether it is capable of accepting it.

As we saw in Section 3.1, two kinds of information are needed for our solution, namely the ATM addresses of both the server and agent, and the QoS parameters of the connection. This information is given by the agent, the server and in the document.

Agent: The fact that an agent is able to use a dual stack is clearly related to the agent itself. Therefore, we propose to add this information in the `UserAgent` field of HTTP requests. This fields allows extra information elements (called product tokens [3]) besides the product name and version of the agent. An agent capable of receiving documents over a direct ATM connection can indicate it by putting the keyword ATM in this field. If it wants the server to open an ATM connection it adds the public and private ATM addresses of the agent to the ATM keyword. (Public and private ATM networks may not use the same addressing scheme and a connection may only possible if the public ATM address of a gateway to the private ATM network of the destination is given.) The public address is given in ASCII decimal digits and the private address in capital ASCII hex digits. The keyword ATM and the two addresses are separated by ASCII dots. If either address does not exist, it is replaced by an empty string.

example:

```
UserAgent:  MyAgent/1.0 ATM.PublicAddress.PrivateAddress
```

Also, the agent will need to tell the server on which socket it expects to receive the data. For this information it can use a `Pragma` field. Pragma fields are generic header fields used for implementation specific information. In our case the agent will use a pragma field to indicate the protocol and the port number of the socket.

example:

```
Pragma:  socket=TCP.8090
```

Server: HTTP responses from servers have a `Server` field which has the same usage as the `UserAgent` field of the agents. When the server responds to a dual-stack agent it adds its ATM address in the `Server` field of its response. The format of the address is the same as for the agent:

example:

```
Server:  MyServer/1.0 ATM.PublicAddress.PrivateAddress
```

Document: The server must recognize QoS sensitive documents and know the QoS requirements of these documents. We propose to add this information into the header of HTML documents. This can be done using `<meta>` elements which can be assigned a name and a content. In HTML 2.0 [3] meta elements are used to give additional information on how to retrieve a document. We propose to name the elements `ATM-Service` and `ATM-QoS-XXX` where `XXX` is the abbreviation for a traffic or QoS parameter according to the User Network Interface (UNI) specification of the ATM Forum [5]. The content of an `ATM-Service` element identifies a QoS service class corresponding to [5] and the content of the `ATM-QoS-XXX` contains the value of the parameter in digital ASCII representation.

example (constant bit rate service with peak cell rate = 2000 cells per second):

```
<meta NAME="ATM-Service" content="CBR">
<meta NAME="ATM-QoS-PCR" content="2000">
```

3.3.1 The HTTP request and response

Having defined the different information elements that will be used, we now define how the agent and server have to behave in presence of this information. The behavior we want to achieve is the following

IF both the agent and the server are on ATM, **THEN**

IF the requested document is QoS sensitive **THEN**

IF the agent chooses to get the document over ATM **THEN**

the server should use AREQUIPA to deliver the document.

To achieve this behavior, dual stack servers react in the following way to requests which contain the ATM keyword plus address in the `UserAgent` field: If the requested document has no `meta` field with QoS specifications then the server delivers it in the standard fashion. If the requested document has QoS specifications and the agent has also defined a socket in a pragma field, then the server requests an AREQUIPA connection and sends the document without any header over the new connection. If the requested document has QoS specifications but the agent has not given a socket, the server only sends the HTTP and HTML headers over the TCP/IP network. This is the same behavior as if the agent had only requested the headers using the `HEAD` method rather than the `GET` method. Having received the headers of the document, the agent can chose if it wants to get the document over AREQUIPA and make a second request with a pragma defining the reception socket.

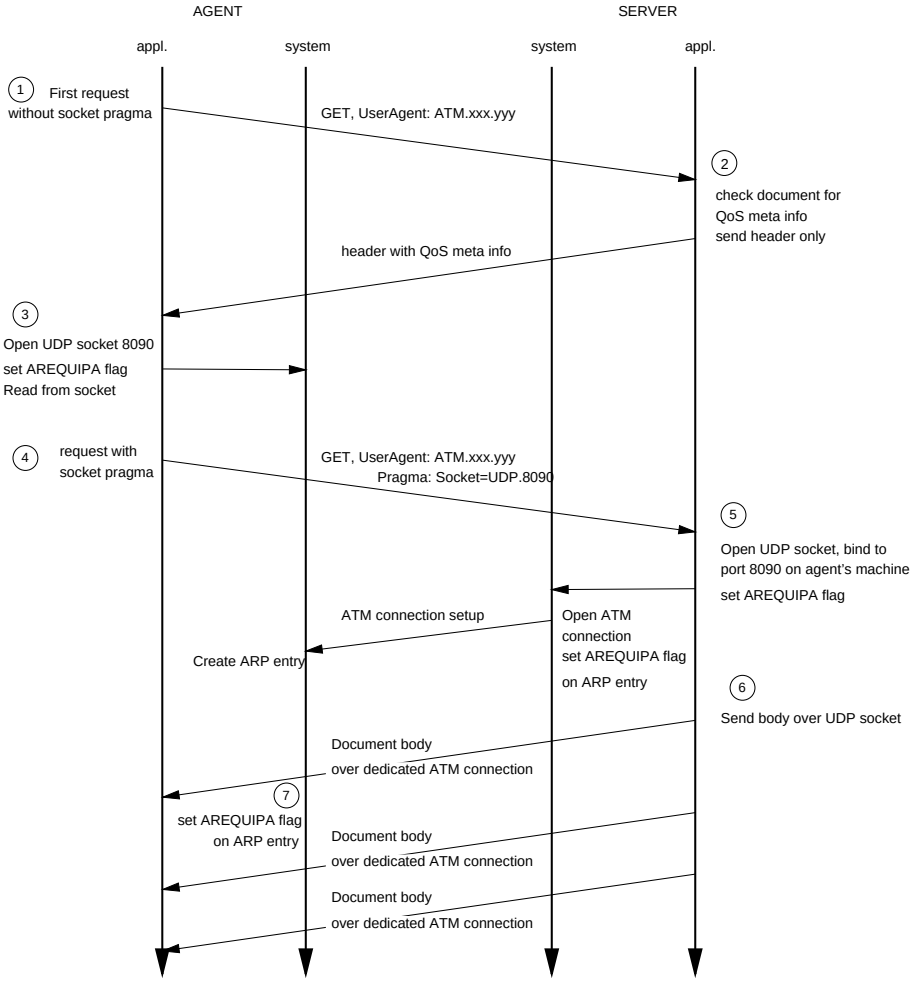

Figure 3: Request and transfer of a QoS sensitive document

Discussion: The only particularity of the HTTP requests of a dual stack agent are the address in the `UserAgent` field and the socket pragma. The server particularly to the ATM keyword by sending only the header over TCP/IP or the document over AREQUIPA. Since standard agents and servers are not supposed to use nor to react to the ATM keyword[1] and the socket pragma the modified agents and servers will interwork seamlessly with standard servers and agents.

3.4 Example

Figure 3 shows an example of the utilization of AREQUIPA between a Web agent and a Web server. **1.** The agent requests a document and gives its ATM address in its `UserAgent` field. **2.** The server sees the ATM address and checks

[1]Of course one has to make sure that the same keyword will not be used to identify other types of user agents.

the document for QoS information in meta elements. It finds QoS information and only send the header of the document to the agent. **3.** The agent decides to request the document and chooses a protocol (UDP or TCP) and a port number on which it wants to receive the document. It opens the corresponding socket and marks it for AREQUIPA with an AREQUIPA system call. **4.** It sends the request again but this time he adds a socket pragma. **5.** The server receives the request and gets the socket pragma. It creates a socket and binds it to the receiver's socket. Then, using the agent's address from the request and the QoS parameters from the document, it asks the system to open a direct ATM connection to the agent. **6.** The server sends the document without the header over the new socket and the dedicated connection to the agent. **7.** When the first datagram is delivered to the AREQUIPA socket of the agent the corresponding ARP entry is identified and marked for dedicated use.

4 Interoperability with non-ATM systems

There are no fundamental interworking problems of our dual stack approach with non-ATM hosts. Dual stack agents or servers can use their IP stack to communicate with non-ATM users. However, if most of the connection between two hosts is based on an ATM network and the IP based portion is small and controllable (eg. a Customer Premises Network, CPN), then the usage of an HTTP proxy [10] can give the advantages of the partial ATM connection to the IP based host.

A proxy HTTP server is an intermediate host which forwards HTTP requests from agents to servers. One use of HTTP proxies is to have a centralized point for caching of Web documents. Another use are firewalls where direct connections between a CPN and the Internet are avoided. In our case an IP based agent would connect to a dual stack HTTP proxy over a fast IP network. The proxy would then receive QoS sensitive documents over ATM and forward them to the agent over the TCP/IP network.

The fact that proxy HTTP servers are at the application level may induce a limitation in performance. More efficient solutions could be found by developing ATM-IP gateways at the router level. This would require more general solutions than just Web oriented ones and is not in the scope of this paper.

5 Conclusions

We have given a simple solution to add QoS guarantees to the delivery of Web documents between hosts connected by ATM. The solution is based on a new service called Application REQUested IP over ATM (AREQUIPA). The solution requires little modification of HTTP servers and clients, they basically need to exchange some extra data and to be able to use the AREQUIPA service. The implementation of AREQUIPA on its side requires a small modification of the networking software, mainly in the ARP layer.

The little modification in the behavior allows a tremendous gain in guaranteed quality delivery without compromising the interoperability with non-ATM servers or agents. The QoS guarantee can be used by information providers to deliver high-quality documents, like video on demand, real-time transmissions

of live events or other multimedia documents. These documents can then be charged on a quality basis. A 5Mbit data stream can be charged higher than a 1Mbit stream, with the guarantee that the user really gets what he pays for (WYPIWYG).

We have also shown how agents not directly connected to ATM will also be able to benefit from partial ATM connectivity by using an HTTP proxy. Finally, we want to point out that the use of AREQUIPA is not limited to Web applications. Using AREQUIPA, other QoS sensitive applications could profit from guaranteed quality service for socket to socket connections.

References

[1] Anthony Alles. ATM internetworking. Internal Paper, Cisco Systems, Inc, May 1995.

[2] Werner Almesberger. High-speed ATM networking on low-end computer systems. Technical Report DI 95/147, Laboratoire de Réseaux de Communication, Swiss Federal Institute of Technology Lausanne, 1995.

[3] T. Berners-Lee and D. Connolly. Hypertext markup language - 2.0. Internet Draft, September 1995.

[4] R. Braden, L. Zhang, D. Estring, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) - version 1 functional specification. Internet Draft, March 1995.

[5] ATM Forum. ATM user-network interface specification version 3.1. ATM Forum Specification, September 1994. Prentice Hall.

[6] The ATM Forum SAA API Ad hoc Work Group. Native ATM services: Semantic description. ATM Forum contribution 95-0008R4, 1995.

[7] D. Katz and D. Piscitello. NBMA next hop resolution protocol (NHRP). Internet Draft, May 1995.

[8] M. Laubach. RFC 1577: Classical IP and ARP over ATM. Internet RFC, January 1995.

[9] R. Leitman. Integrating HTTP with ATM. Master thesis, University of Waterloo, 1995.

[10] Ari Luotonen. Wolrd-wide web proxies. In *Proceedings of the First International Conference on the World-Wide Web*, May 1994.

[11] Philippe Oechslin. The web over atm project. WWW document "http://lrcwww.epfl.ch/WebOverAtm/".

[12] S.-M. Tan, Z Chen, R. H. Campbell, and Y. Li. Real tmie video and audio in the world wide web. In *Proceedings of the Fourth International World Wide Web Conference*, Boston, 1995.