Arequipa: Using TCP/IP over ATM with quality of service is simple... if you have ATM end-to-end

Werner Almesberger,
Jean-Yves Le Boudec*,
Philippe Oechslin
ICA
EPFL,
Lausanne, Switzerland.

Abstract: We will present the design, implementation and our first experience with Arequipa. Arequipa is a method for providing the quality of service of ATM to TCP/IP applications without requiring any cooperation in the network between IP and ATM. It does not need any modifications in the ATM or IP networks; however, it requires end-to-end ATM connectivity. We will describe how we implemented our approach and made it publicly available in Linux, how we applied it to the Web, and finally, how we tested it on a European ATM wide area network.

Keywords: Arequipa, Resource reservation, ATM, TCP/IP, Quality of Service, QoS, Internet, Linux, Web.

1. Introduction

Transferring multi-media data and other time-critical data over networks may need a dependable Quality of Service (QoS). This is the case for all situations where a minimum guaranteed bandwidth is required. There are fundamentally two approaches to provide Quality of Service:

– The network can be dimensioned in such a way that traffic requiring quality of service is very unlikely to ever exceed the available resources. This normally requires that different priority levels be used, with higher priority given to flows that are expected to receive high quality of service. This is the approach taken in proprietary architectures such as [CIS 97]. The work currently underway at the Internet Engineering Task Force (IETF) on the topic of differentiated services [DIF] may also provide solutions in that direction.

– An alternative is to explicitly reserve resources for specific data flows [ZHA 95]. This line of thinking finds its origin in the past experience in telephony networks.

Our work is positioned within the second approach, namely the reservation approach.

One network technology for providing reservations is the Asynchronous Transfer Mode (ATM) [LEB 92, ATM 96], which promises to provide a scalable network architecture. The design of ATM considered reservation mechanisms from the very beginning. ATM networks therefore now offer reliable reservation mechanisms and well-understood traffic management concepts. Corporate and public ATM networks are already a reality in many places. Applications that are specifically written to use ATM, so-called *native* ATM applications, already guarantee end-to-end QoS today. However, one major problem is that the vast majority of networked applications is written to TCP/IP service interfaces, not to ATM. If you want to use TCP/IP applications in such environments, the standard solution is to run IP over ATM; however, with IP over ATM today, there is no simple way yet to benefit from the end-to-end QoS guarantees of ATM.

*Contact author ; Leboudec@epfl.ch

Another network technology for providing reservations could be based on enhancements to the Internet. The current Internet does not provide reservations, but the IETF identified the need for supporting integrated services years ago [CLA 92, BRA 94], and has been working on the design of reservation mechanisms for TCP/IP. One major result of this activity are the Resource reSerVation Protocol (RSVP [BRA 97]) and the corresponding mappings to specific link layers, which are currently still in draft status. This approach is based on the concept of *integration*: network nodes (here: routers) need to be upgraded in order to support an additional set of functions required by the reserved services. Once and if RSVP is deployed across the Internet, it is possible to use TCP/IP applications with some end-to-end quality of service.

In this article, we report on the feasibility of an alternative approach, called Application REQuested IP over ATM (Arequipa). Our purpose with Arequipa is to show that providing the quality of service of ATM to TCP/IP applications is straightforward with minimum changes to the TCP/IP implementation in hosts. For two end-systems to communicate using Arequipa, it is necessary that they are connected (1) to a common ATM network and (2) to the Internet or to the same Intranet. However, there is no cooperation required between the two types of networks. We say that our approach is based on a concept of *segregation*. Arequipa allows applications to establish direct point-to-point end-to-end ATM connections with a given QoS at the link level. These connections are used exclusively by the applications that requested them. After setup of the Arequipa connection (namely, the ATM connection that is used for Arequipa), the applications can use the standard TCP/IP service to exchange data.

We made the conscious choice to let the user, or the application, explicitly control the ATM connection. In our implementation, we support unspecified bit rate (UBR) and constant bit rate (CBR) connections. In the latter case, the user or application has to specify the requested peak cell rate. The additional traffic or QoS parameters required by the ATM signalling procedure are set transparently by our implementation. We believe that it is reasonable to limit the information requested from the user or application to just the choice mentioned above, namely: UBR with no rate information, or CBR with a specified peak cell rate. Our choice to make the traffic specification visible to the application is an essential part of Arequipa; we believe that it will become more common in the future for a large variety of applications.It is based on the concept that QoS comes with a price, and therefore we expect a dialogue between application and user to take place before a guaranteed QoS is requested. However, this does come with a drawback: existing application code has to be modified. We did the modification to a web client and a web browser, as reported in Section 5. Similar work is underway for video and audio conferencing applications. An alternative to our approach is to let the operating system choose the traffic parameters in lieu of the application. We do not follow this approach with Arequipa because we explicitly want to make quality of service visible to the end-user.

Considerable work has been devoted in the Broadband ISDN context on defining an application level signaling framework that would enable applications to negotiate services and determine service access points, depending on application profiles, terminal capabilities and service requirements by end-users [RAC 92]. We claim that such efforts are to a large extent redundant with the existing base of Internet applications (such as the Web). With Arequipa, it is possible for applications to use the Internet for exchanging short messages for purposes of service negotiation, address mapping, authentication, and then set up ATM connections as needed.

The article is structured as follows: section 2 describes mechanisms for running IP over ATM which are either currently in use or which are being defined by the IETF or the ATM Forum. In sections 3 and 4, we explain the concept of Arequipa and how we implemented it in a UNIX-like operating system. Section 5 introduces a way of using Arequipa with the World-Wide Web (WWW, [BER 91]). In section 6, we describe how we tested Arequipa to transfer video data across Europe.

Although of high importance, the related issue of pricing for ATM services is not considered in this paper.

2. Transmitting IP Packets over ATM

The IETF and the ATM Forum have defined various mechanisms that can be used to send IP traffic over ATM, and they continue developing new mechanisms and refining the existing ones. For what could be called the second generation of such mechanisms, both groups have joined forces and are closely synchronizing their efforts. This section briefly describes the current state of affairs.

2.1. *Classical IP over ATM*

The first standard developed by the IETF for running IP over ATM is the so-called *classical IP over ATM*, defined mainly in RFC1577 [LAU 94], but see also RFC1483 [HEI 93], RFC1755 [PER 95], and RFC1932 [COL 96]. In that scheme, IP hosts are grouped in Logical IP Subnets (LIS) which are typically interconnected with IP routers. The ATM network is treated much like a LAN and hosts within a LIS can obtain each other's ATM addresses through an address resolution protocol that maps IP addresses to ATM addresses. After obtaining the address of a destination (within the LIS), an ATM connection is established to it. If a packet has to go to another host outside the LIS, it is sent to a router which forwards it.

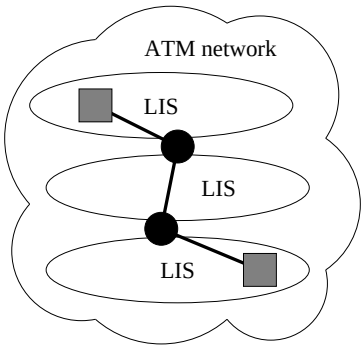

Figure 1. *Classical IP over ATM has to use routers even if a direct ATM connection could be established between communicating hosts*

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 between the communicating hosts would be possible, as illustrated in figure 1. Also, all the data flowing between two machines typically uses the same ATM connection, making it impossible to request a QoS for one specific data stream.

2.2. *LAN Emulation*

LAN Emulation (LANE, [ATM 95]) is ATM Forum's equivalent to classical IP over ATM. Like the latter, it limits direct ATM connections to a comparably small cloud of systems, the so-called Emulated LAN (ELAN). The main differences to classical IP over ATM are that LANE uses IEEE 802 [IEE] MAC addresses instead of IP addresses, and that it also includes support for multicast and broadcast mechanisms.

LANE version 1 has no concept of honoring QoS requirements of upper layers. Support for this is planned for LANE version 2.

2.3. *Next Hop Resolution Protocol*

An improvement for classical IP over ATM is the Next Hop Resolution Protocol (NHRP, [LUC 98]). This protocol tries to resolve ATM addresses for hosts which are not in the same LIS. With the ATM address of a remote host, a direct ATM connection can be established, bypassing intermediate routers (see figure 2). In cases were the ATM address of a remote host can be resolved, NHRP can thus provide end-to-end ATM connections.

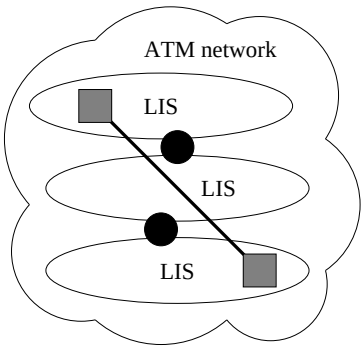

Figure 2. *NHRP can establish direct end-to-end ATM connections between hosts*

However, NHRP has no mechanism to manage the QoS of such connections. Data from different applications may transit through the same end-to-end ATM connection and the QoS an application experiences depends on the traffic load generated by the others.

2.4. *Multiprotocol over ATM*

Multiprotocol over ATM (MPOA, [ATM 97]) merges protocols developed by the IETF and the ATM Forum, and extends them for using end-to-end ATM connections also with non-IP protocols, such as IPX. This includes mainly: NHRP and the multicast mechanism described in RFC2022 [ARM 96]. In addition, MPOA has mechanisms for flow classification, in order to decide automatically at layer 3 when an ATM shortcut should be established. It also supports the decoupling from the data and control paths in intermediate systems.

Like LANE, phase 1 of MPOA does not consider QoS, but phase 2 will.

2.5. *RSVP*

Current IP networks are designed to provide a best effort service. This explains why the aforementioned solutions for running IP over ATM do not pass the notion of QoS guarantees that ATM provides to the IP layer.

The standard approach for providing QoS guarantees in IP networks is the use of the Resource Reservation Protocol (RSVP). RSVP is part of the Integrated Services framework for the Internet, currently being standardized. It typically hinges on mechanisms like packet schedulers, which make sure that data flows for which reservations have been made get their share of bandwidth on links. In this framework, RSVP is the signaling protocol, propagating information about available services and requests for reservation along the data path between sources and destinations.

2.6. *Guaranteed Internet Bandwidth*

A mechanism called "Guaranteed Internet Bandwidth" (GIB [ARA 96]) approaches the QoS issues by directing flows with QoS requirements over dedicated wide area network (WAN) connections (for example ISDN or ATM). End systems use a special signaling protocol to ask a GIB agent to change the routing tables of the gateway routers. Limiting flow selection to IP routes (namely, to the destination IP address) allows the use of standard routers, but makes the isolation of concurrent flows unreliable.

2.7. *Discussion*

The methods of running IP over ATM presented above (except for GIB) have one thing in common: They hide the fact that ATM is being used from the applications. Since ATM is used below the IP layer and IP has no notion of connections or QoS, the interoperation mechanisms hide those properties of ATM.

NHRP/MPOA and RSVP alleviates the problem by setting up end-to-end connections below the IP layer and by setting up reservations above it. This approach has the advantage of being very general but it adds some complexity and has the following restrictions.

– In order for NHRP to be effective, it must be deployed on all the nodes between communicating hosts. If this is not the case, NHRP is not able to create an end-to-end connection between the hosts and RSVP will not be able to guarantee reservations on the entire path. RSVP can operate even if some routers do not implement it, however, there is no reservation on such paths. On an ATM WAN, for example, end stations must rely on their service providers to deploy NHRP and RSVP over all parts of the WAN between them, in order to benefit directly from ATM guaranteed QoS.

It is true that QoS guaranteeing services need only be deployed on the congested parts of the network. However, the congested parts are usually the backbones and long-distance links, which is where it is most complicated to install new services. Arequipa avoids this problem since it only needs to be installed on end systems.

– With public ATM connections being offered by telecom companies, it is now possible to have long distance end-to-end ATM connections, even

across country borders. Thus two hosts in two distant ATM WANs may be able to open end-to-end ATM connections through their public network operators. However, the IP backbone on the long distance path between the WANs may well not be running on ATM. Thus NHRP may not be able to resolve the ATM addresses and to set up end-to-end connections.

In contrast to all methods presented above, Arequipa aims at providing the QoS of ATM directly to the application, and at letting the application control its use. It is not clear at this time which approach has superior benefits, but it should be emphasized that they pursue different objectives. Arequipa is based on network segregation, with integration in the hosts only; it is therefore less general but also considerably simpler.

3. Arequipa

Arequipa allows applications to establish end-to-end ATM connections under their own control, and to use these connections at the lower protocol layer to carry the IP traffic of specific sockets.

Unlike the connections set up by classical IP over ATM or by LANE, Arequipa connections are used exclusively by the applications that requested them. The applications can therefore exactly determine what QoS will be available to them.

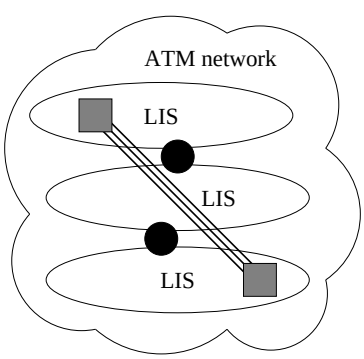

Figure 3. *End-to-end Arequipa connections for three applications with QoS requirements*

Figure 3 illustrates that Arequipa connections go end-to-end and that each flow has its own connection.

In its broadest sense, Arequipa offers a means to use properties of a network technology that is used to transport another network technology (e.g., IP on ATM) without requiring the explicit design and deployment of sophisticated interworking mechanisms and protocols.

Traditional protocol layering typically only allows access to functionality of lower layers if upper layers provide their own means to express that functionality. This approach can introduce significant complexity if the semantics of the respective mechanism are dissimilar. Also, if the upper layer fails to provide that interface, no direct access is possible and the lower layer functionality may be wasted or used in an inefficient way (e.g., if using heuristics to decide on the use of extra features). By allowing applications to control the lower layer, Arequipa enables them to exploit those properties.

Note that Arequipa coexists with "normal" use of the networking stacks, i.e., applications not requiring Arequipa do not need to be modified and they will continue to use whatever other mechanisms are provided.

3.1. *Example*

Figure 4 illustrates the case of TCP/IP over ATM: TCP connections between applications are built by multiplexing their traffic over an upper layer (IP), which is in turn carried by a lower layer (e.g., Ethernet or ATM). Routers terminate lower layer segments in order to overcome scalability limitations of either layer or of the interface between the layers.

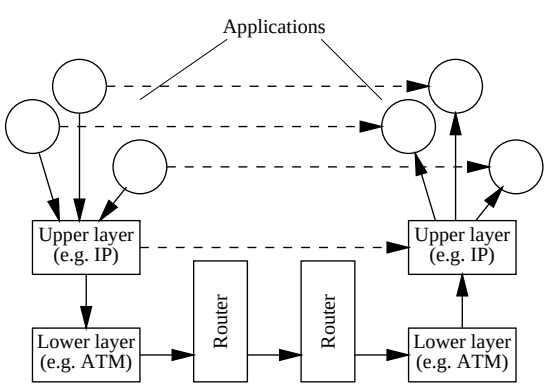

Figure 4. *Communication without Arequipa*

Figure 5 shows the same scenario, but this time using only Arequipa. The applications still have their TCP connections, but there is one dedicated end-to-end (Arequipa) connection at the lower layer for each of them.

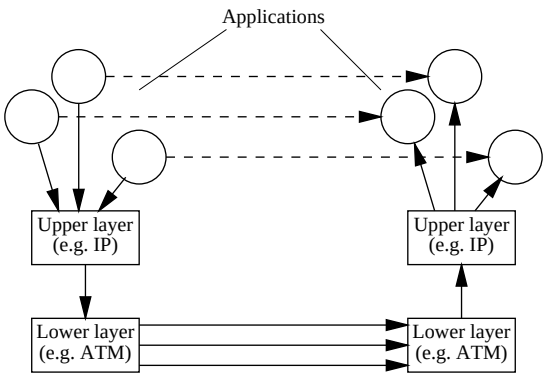

Figure 5. *Communication with Arequipa*

Note that, although traffic between applications using Arequipa does not pass the normal routed IP path anymore, general IP connectivity may still be necessary, e.g. for ICMP messages or for traffic of other applications.

3.2. *Applicability*

Arequipa is applicable if the following two conditions are met:

– applications can control "native" connections over the lower layer communication media

– the upper and the lower layer (e.g., IP and ATM) both allow communication between the same end-points (or they share at least a useful common subset of reachable end-points)

The next two conditions do not have to be met, but without them the use of Arequipa may be questionable:

– the upper layer is multiplexed over the lower layer (e.g., when using classical IP over ATM, all IP traffic between a pair of hosts typically shares the same ATM SVC)

– multiple lower layer connections are possible between a pair of end-points

In order to simplify interaction with the protocol stack, Arequipa assumes that data sent to destinations for which no Arequipa lower layer connection has been established will be delivered by some default mechanism.

Note that, despite its name (Application REQuested IP over ATM), Arequipa is not limited to IP and ATM only. The upper layer is typically IP or some similar protocol (e.g., IPX). The lower layer can be ATM, Frame Relay, N-ISDN, etc. Some of the advantages of using Arequipa in addition to the usual IP mechanisms are avoidance of routing overhead and the possibility of using dedicated connections with "hard" quality of service guarantees. This is of interest for flows with a lifetime which is long compared to the setup delay incurred by the lower layer.

3.3. *API*

The following primitives are available to applications using Arequipa:

```
int arequipa_preset(int sd,const struct
sockaddr_atmsvc *addr,const struct atm_qos
*qos);
```

Presets the specified INET domain socket to use a direct ATM connection to `addr` with the QOS parameters specified in `qos`. If the socket is already connected, the ATM connection is set up immediately and data is redirected to flow over that connection.

```
int arequipa_expect(int sd,int on);
```

Enables (if `on` is non-zero) or disables (if `on` is zero) the use of Arequipa for return traffic on the specified INET domain socket. When enabling the use of Arequipa for return traffic, the Arequipa connection on which the next data packet or incoming connection for the socket is received is attached to that socket.

```
int arequipa_close(int sd);
```

Dissociates an Arequipa VC from the specified socket. After that, traffic uses normal IP routing. Note that the Arequipa connection is automatically closed when the INET socket is closed.

3.4. *Use of ATM user-to-user signaling*

ATM connections for Arequipa use are used almost exactly like connections for IP over ATM. However, in order to avoid conflicts with the IP over ATM entity, Arequipa connections are signaled in a slightly different way, so the rule is as follows:

An Arequipa connection is signaled by using the procedures and codings described in RFC1755 [PER 95], with the addition that the Broadband High Layer Information (BHLI) information element be included in the SETUP message, with the following coding:

`bb_high_layer_information`		
`high_layer_information_type`		
	3	(vendor-specific application id.)
`high_layer_information`		
	00-60-D7	(EPFL OUI)
	01-00-00-01	(Arequipa)

4. Implementing Arequipa in a Unix environment

This section describes general aspects of implementing Arequipa for IP over ATM in a socket-based operating system kernel. The organization of kernel internal data structures is assumed to be similar to the one found in the networking part of the Linux kernel [COX 96].

4.1. *Kernel data structures without Arequipa*

Figure 6 shows some of the kernel data structures that are typically associated with a TCP socket when not using Arequipa. Incoming data is demultiplexed by the protocol stack (in figure 6, the circle with TCP/IP) and queued on the socket. Outgoing data is multiplexed by the protocol stack and sent to the corresponding network interface.

Each data packet consists of a packet descriptor and the actual data. The packet descriptor contains information like the socket the packet belongs to, the interface on which it was received, etc.

Most modern TCP/IP implementations also cache routing information (including the network interface) for each socket, so that route lookups only need to be done when a new connection is established or if the routing table is modified.

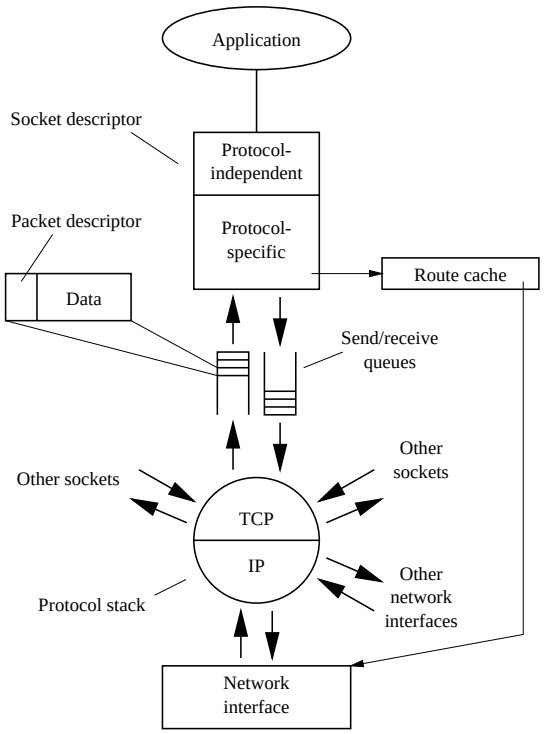

Figure 6. *Kernel data structures of a TCP socket (simplified)*

4.2. *Data structures for incoming Arequipa*

When using Arequipa, incoming packets are handled as if they were using Classical IP over ATM: after little or no ATM-specific processing, they are passed to the protocol stack, which then performs the usual demultiplexing, etc. The only significant difference is that they are marked in order to identify them as originating from Arequipa (and from which VC) when they arrive at the socket.

Figure 7 shows the data structures used when receiving from Arequipa. Note that all Arequipa VCs on a system can use the same Arequipa pseudo-interface.[1]

[1] The term "pseudo-interface" is used to make it clear that the Arequipa interface does not correspond to a physical network interface (namely, hardware) although the protocol

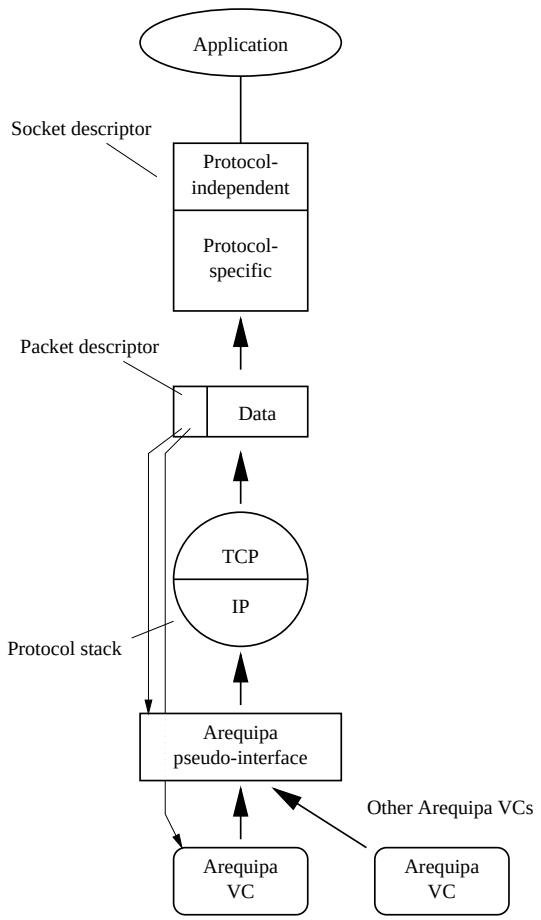

Figure 7. *Arequipa for incoming data*

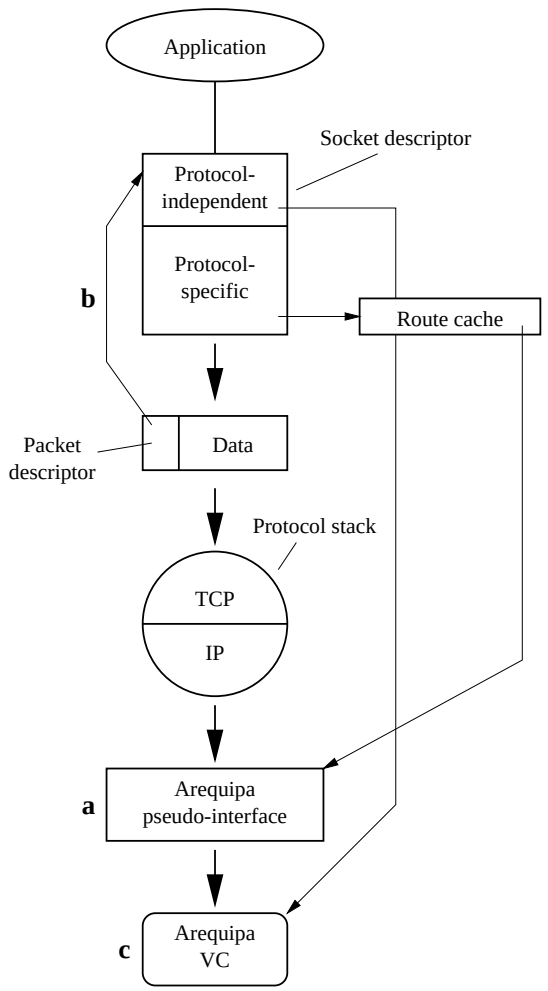

Figure 8. *Arequipa for outgoing data*

If the socket is not yet using Arequipa for sending (namely, if it has no associated Arequipa VC) and if it expects incoming Arequipa traffic (namely, if `arequipa_expect` has been invoked with the `on` argument set to a non-zero value), the Arequipa VC on which the packet has been received is attached to the socket, so that outbound traffic uses the VC.

Note that if a packet is received over an Arequipa VC, then it is possible that the VC no longer exists at the time the data is delivered to the socket. It is therefore necessary to verify the validity of the incoming Arequipa VC before attaching it to the upper layer socket (the normal closing procedures only ensure that both layers are synchronized *after* establishing the association).

stack interacts with it as if it did.

This can be implemented as follows:

– all incoming Arequipa VCs are registered in a list (while they are dangling because they are not attached)

– there is a global generation number, which is incremented whenever a new Arequipa VC is created. The generation number at the time of VC creation is stored in the VC descriptor.

– the generation number of the Arequipa VC is recorded in the descriptor of each data packet arriving on that VC

A VC is still valid when the packet is delivered

to the socket only if a reference to that VC is on the list and if its generation number matches the one stored in the packet descriptor.

4.3. *Data structures for outgoing Arequipa*

When sending from an Arequipa socket, outbound packets must be associated with the corresponding Arequipa VC. As illustrated in figure 8, this is done by sending them all through an Arequipa pseudo-interface (a) which then looks up a back pointer (b) to the originating socket in the packet descriptor. The originating socket contains a pointer to the descriptor (c) of the VC over which the data has to be sent.

Note that an Arequipa connection may be removed (e.g. because the remote party has closed it, because of a network failure, etc.) without notification at the socket. In this case, the Arequipa route is removed and all outbound traffic is sent with the "normal" IP mechanisms again.

4.4. *Networking code changes*

If using the approach outlined in the previous sections, the networking code has to be modified at least at the following places:

– when creating a socket, the Arequipa information (namely, if Arequipa is in use on that socket, if the socket expects incoming Arequipa traffic, etc.) needs to be initialized

– when connecting a UDP or TCP socket, a cached Arequipa route exists if `arequipa_preset` was invoked before the `connect` system call. This cached route must be preserved.

– when delivering data from Arequipa to a socket, the Arequipa VC is attached to the socket if

– the socket expects incoming Arequipa traffic, and

– the socket does not currently use Arequipa, and

– the Arequipa VC is not already attached to a different socket

– if an incoming TCP connection is received on a listening socket which expects incoming Arequipa traffic, the new socket (the one returned by `accept`) is also set to expect incoming Arequipa traffic and, if the packet has arrived via Arequipa and if the constraints listed above are met, the Arequipa VC is attached to the new socket

– when an upper layer socket is closed, the underlying Arequipa connection has to be closed too

– when forwarding IP packets, packets received over an Arequipa connection must be discarded (see [ALM 97], section 6)

Additional modifications may be necessary depending on how per-socket route caches are invalidated. Also, socket destruction may be interrupt-driven and may therefore need special care.

4.5. *TCP issues*

The use of TCP over Arequipa raises two specific problems: (1) if the Arequipa connection is attached after establishing the TCP connection, the maximum segment size (MSS) of TCP may be very small, typically increasing processing overhead. (2) there are no generally useful semantics for listening on a socket for which an Arequipa connection has already been set up.

TCP implementations frequently limit the MSS to a value which is based on the MTU of the IP interface on which the connection is started. If connections are set up over a media with an MTU size that is small compared to the default IP over ATM MTU size [ATK 94], that MSS will have to be kept even if Arequipa is later used for that socket (see RFC1122 [BRA 89], section 4.2.2.6). It is therefore recommended to invoke `arequipa_preset` before `connect` and to invoke `arequipa_expect` before `listen`.

Note that this is the only way to ensure that the use of Arequipa is known at both ends when exchanging the initial SYN segments. Applications that require the TCP listener to set up the Arequipa connection are therefore not able to ensure the use of a larger MSS.

Although the API could allow associating an Arequipa connection with a socket that is used to

listen for incoming connections, the usefulness of such an operation is questionable.

Therefore, attempts to execute `arequipa_preset` on a listening socket or to `listen` on a socket for which an Arequipa connection already exists yield an error.

Further implementation details, including a step-by-step description of the changes that were necessary when adding Arequipa support to Linux, can be found in [ALM 96a].

5. Transmitting Web documents with guaranteed QoS

One of the most popular applications on IP networks is the World Wide Web (WWW, [BER 91]). Its popularity stems from the fact that it allows to access many different types of multimedia documents with a single intuitive user interface.

The Web is also a good example for an application that can benefit from dependable QoS: if the network can guarantee the required bandwidth, data with real-time constraints (e.g., video clips) can be displayed during reception and does not have to be downloaded and replayed from a file, as it is currently done. Also, users frequently have loose time constraints (e.g., the time to download stock exchange information). QoS guarantees ensure that users can obtain an adequate service and won't be subjected to the vagaries of best-effort.

5.1. *Arequipa and the Web*

In order to use Arequipa for Web applications, three types of information are needed:

– The side that establishes the Arequipa connection must know the ATM address of the opposite side.

– Likewise, the side that establishes the Arequipa connection must be able to specify the QoS information.

– Finally, the side that establishes the TCP/IP connection (normally using either TCP or UDP) also needs to know the destination port.

We have chosen to let the Web server open Arequipa connections to the client, thus the server needs to know the ATM address of the client. This information can be sent conveniently as a pragma in the client's request [BER 96]. This pragma can serve another purpose, namely to indicate to the server that the client is capable of using Arequipa.

The server also establishes the TCP/IP connection on which the document is transferred, so the client needs to indicate the port on which it expects the data. For convenience, it also includes the protocol type along with the port number.

For each document, we want to be able to specify whether a connection with guaranteed QoS should be used. If yes, we also want to specify what kind of service and what QoS should be requested. To specify this information, we use the notion of meta-information for Web documents. Web servers are able to store meta-information for each document, either in the header of the document or in a separate file. We use this feature to store the ATM service and the QoS parameters to be requested.

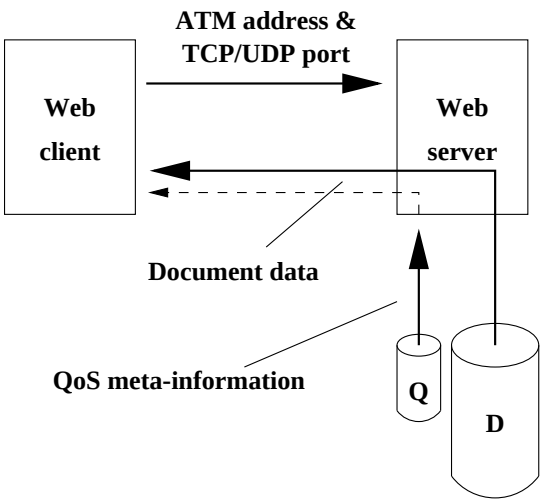

Figure 9. *General information flow when using Arequipa with the Web*

Although not strictly required, the QoS information is also useful to the client, so it is included in the meta-information the server sends in response to a request. Figure 9 illustrates the general information flow.

Because a client may want to know the QoS attributes of a document before downloading it with Arequipa (e.g., because the client wants to ensure that sufficient local resources are available to handle the document, or because the user is charged for ATM connections and therefore only wants to use Arequipa for selected documents), we also need a mechanism to obtain only the headers, which include the QoS meta-information.

While a client could always send a `HEAD` request before issuing a `GET`, this would add one extra round-trip time for every request, whether or not the document in question is eligible for being transferred with Arequipa. This is clearly undesirable. We therefore extend the semantics of `GET` to only return the header of the document under the following conditions:

– the document has associated QoS information, and

– the client indicated that it supports Arequipa (by sending its ATM address), and

– the client did **not** include the destination port number.

If the client decides to retrieve the document using Arequipa, it issues a second request, this time with the destination port number. Note that a client can avoid the extra round-trip if it has a priori knowledge about the document (e.g., if the headers are cached) or if it wants to use Arequipa anyway, whatever the requested QoS is.

The extended behavior of the Web server is shown in the pseudo-code below:

```
if (request_has_ATM_address &&
  document_has_QoS_metainfo)
    if (request_has_port_number)
        send_document_using_arequipa();
    else send_headers_only();
else /* non-QoS document or non-Arequipa
        client */
    send_document_standard_way();
```

Note that this extension is compatible to the standard HTTP protocol and that Arequipa capable servers and client will interact seamlessly with their standard counterparts.

5.2. *HTTP extensions*

When the client sends additional information required for Arequipa, it uses the following extra header fields:

`Pragma:` `ATM-address=`*pub_address.prv_address*

pub_address is the public E.164 address [ITU 91] of the client. If the client has no such address, that part of the field is empty. *prv_address* is the private ATM NSAP address of the client. If the client has no such address, that part of the field is left empty. Presence of the `ATM-address` pragma indicates that the client supports Arequipa and that it wishes to make this fact known to the server.

`Pragma:` `socket=`*protocol.port_number*

protocol is typically `TCP` or `UDP`. *port_number* is the corresponding port number or whatever information the protocol may use to identify end-points. Presence of the `socket` pragma indicates that the client wishes to retrieve the requested document over Arequipa, if the document is suitable for this, and if the server supports Arequipa.

QoS meta-information is sent by the server by adding the following new fields to the document header:

`ATM-Service:` *service*

service is either `UBR` or `CBR`.

`ATM-QoS-PCR:` *peak_cell_rate*

`peak_cell_rate` is the required peak cell rate in cells per second. This field can be omitted when using UBR.

5.3. *Example*

Figure 10 shows a sample HTTP dialog when using Arequipa.

The client first sends its request without the port information, so that the server only returns the header. After asking the user for permission to request retrieval with Arequipa, the client sets up its socket and repeats the request, this time with the port information. The server can now establish the Arequipa connection and sends the document with the specified quality of service.

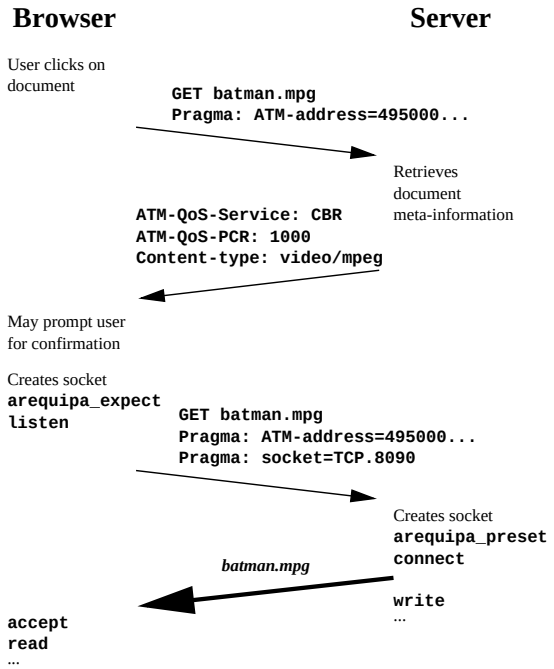

Figure 10. *Example request/response flow when using Arequipa on the Web*

5.4. *Arequipa with proxies*

A proxy Web server (short "a proxy") is a Web server that requests documents from other Web server on behalf of clients. Typical uses for proxies include Web caches and application-level gateways through firewalls.

When used in conjunction with a proxy, Arequipa can even be useful to client that are not directly connected to ATM: If the network between the client and the proxy is dimensioned to offer enough bandwidth so that congestion is very unlikely (the typical situation in a LAN), it is sufficient if Arequipa is used only over the – possibly congested – WAN.

Figure 11 illustrates the use of Arequipa with a proxy. The proxy uses Arequipa when transferring documents over the WAN from remote servers. The client uses the best-effort service of its LAN and doesn't even have to know about Arequipa.

The pricing question for cached documents is interesting, but, as mentioned earlier, is outside the scope of this paper. Note that in our example, the cache is located on the firewall or on the LAN of a company and all costs occurred by the users of the cache will be billed to the company.

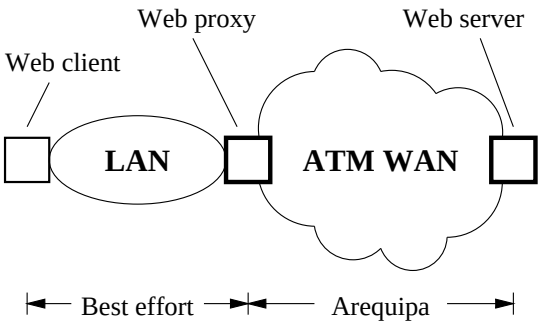

Figure 11. *Arequipa with a proxy Web server*

6. An Arequipa test in the WAN

In October 1996, a demonstration of Arequipa was performed in an ATM WAN environment. This was done as part of an interim presentation of the "Web over ATM" project [OEC 96], which also comprises the work on Arequipa. A number of other tests and demonstrations of Arequipa were performed in 1997 as part of the European ACTS project.

The demonstration consisted of the transmission of raw uncompressed live video over TCP with Arequipa across Europe (see figure 12). The purpose of this demonstration was to show how bandwidth-intensive applications can benefit from Arequipa.

Arequipa is part of the ATM-Linux distribution. The test reported here used computers with an Intel processor, a PCI bus and ATM adapters from two different vendors. The network used a number of different ATM switches from various vendors.

6.1. *The Network*

The transmission was done from sites in Helsinki (Finland) to EPFL in Lausanne (Switzerland), using the JAMES (Joint ATM Experiment on European Services)[2] network. The partner sites in Finland were Nokia and Telecom Finland.

[2]See `http://btlabs1.labs.bt.com/profsoc/james/`

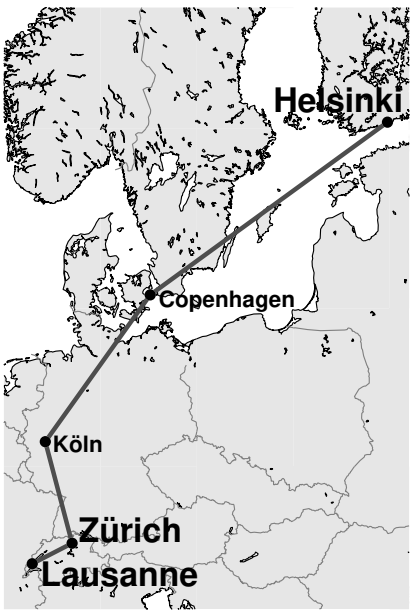

Figure 12. *Arequipa test in European WAN*

In order to experiment with the setup without wasting bandwidth in the international network, preliminary testing was done with ETH in Zürich (Switzerland). An overview of the sites involved is shown in figure 13.

The WAN connections with Finland were virtual paths with a constant bit rate of 77'200 cells per second (corresponding to a user data rate of almost 30 Mb/s). The connection with ETH was a virtual path with a bandwidth of approximately 34 Mb/s.

As described in section 5, the Web was used to start and to control the video transmissions.

6.2. *Results*

The demonstration setup worked as expected and, using the video application with traffic shaping set to allow a user data rate of 27.3 Mb/s, a throughput of approximately 25 Mb/s was obtained for video traffic from Finland.

Also, the throughput for TCP over Arequipa without application overhead was tested on the 34 Mb/s virtual path with ETH. This benchmark was done with `ttcp`, a program that sends/receives

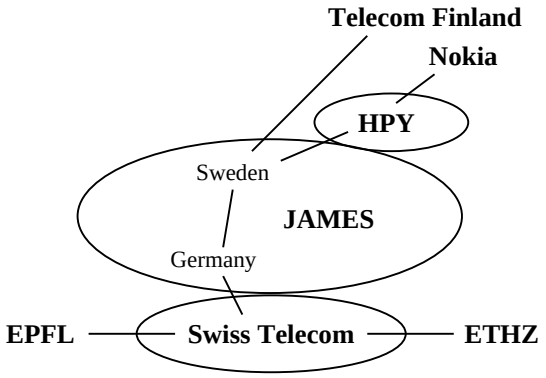

Figure 13. *Schematic overview of the network structure*

to/from memory without doing any further data processing. With traffic shaping set to 33.3 Mb/s, we obtained a throughput of up to 33.0 Mb/s.

While those results tend to indicate that our implementation of Arequipa is able to sustain high rates, it should be noted however that the goal of Arequipa is *not* to obtain a high throughput. In contrast, Arequipa allows an application to receive a specified throughput, with hard guarantees, and in most cases for a price. This was illustrated in the demonstration using video images of a remote clock, with the display sent over a TCP connection using Arequipa. Depending on the selected peak cell rate, the watch would either run too fast, too slow, or just at about the right speed, obviously something, you couldn't obtain with the normal Internet.

7. Conclusions and lessons learned

We have presented Arequipa, a method for providing the quality of service of end-to-end ATM connections to TCP/IP applications. It makes it possible to use ATM natively, in those cases where end-to-end ATM connectivity exists, while preserving the TCP/IP environment, and with only minimal changes to the application code. We have implemented Arequipa in Linux, tested it extensively, made it a part of the ATM-Linux distribution, and published an Internet RFC documenting it. We have described a way of using Arequipa with the Web without sacrificing compatibility with stan-

dard Web browsers and servers. We have indicated how Arequipa can be of use even for hosts not directly connected to ATM, using application level proxies. Finally, we have presented the results of a test of Arequipa in a European WAN setup.

The approach taken by Arequipa is that of service integration in hosts only. It relies on the fact that TCP/IP implementations do not follow a strict layer separation: the IP destination tables in hosts are typically set per socket pair, rather than per IP destination address. This makes it possible to select a given ATM connection for one specific application flow, instead of for one IP destination address. Service integration in hosts rather than in the network makes it possible to use QoS immediately, since ATM commercial networks are already in operation.

There is a number of lessons we learned from the implementation and deployment of Arequipa, but in this conclusion we would like to focus on one major lesson. It has to do with the observation that, contrary to our expectation when we started the project in 1994, the penetration of end-to-end ATM remains minuscule. One obvious reason is the fact that ATM requires specific communication adapters, and cannot run today on the existing hosts, which, for the vast majority of them, use Ethernet. However, our work on Arequipa may give us some additional clues about the reasons for this state of affairs. Certainly, the lack of ATM penetration is *not* due to the difficulty of making QoS available and visible to the user. Indeed, with Arequipa, we have a solution readily available for the Unix environment, and we conjecture that porting that solution to the market dominating operating system would not be a major effort. We also do not believe that the minor changes required to web clients or servers are a major drawback, since the lifetime of this type of software is usually shorter than a year. If there would be a massive push to obtain QoS in hosts, then the ATM penetration would be higher. Therefore, we are lead to think that making QoS visible to the user is an idea that simply did not meet its market. Many users would like to have it, but hardly any organization is willing to invest in the network technology required to support it. This also leads us to conjecture that approaches based on RSVP that would attempt at making QoS visible to the end user will equally suffer from the same lack of penetration, because introducing RSVP into the Internet is also a major investment.

If we follow this line of thought, we conclude that it may not be a good idea to let the QoS be visible to the application, except maybe for niche application settings where the investment is justified. Such settings are, for example, remote lecture rooms used in medical teaching applications, or video on demand over ATM. In contrast, in a TCP/IP setting, it may be that what users need is "a better service", instead of, for example, "a guaranteed 250 kb/s flow". This could be supported by priority mechanisms, which are simpler to implement than reservations.

8. Available software

An implementation of the Arequipa mechanisms is part of the ATM on Linux [ALM 96b] distribution. This distribution contains full source code for kernel changes, system programs, and test applications. It can be found at `http://lrcwww.epfl.ch/linux-atm/`

An application package for Arequipa is available on `http://lrcwww.epfl.ch/arequipa/`. It includes the following components (with complete source code):

– Web server and proxy server: CERN `httpd` with Arequipa extensions

– Web browser: Arena with Arequipa extensions

– Video application: a modular video capture and playback package

References

[ALM 96a] Almesberger W, Arequipa: Design and Implementation, `ftp://lrcwww.epfl.ch/pub/arequipa/aq_di-1.tar.gz`, Technical Report 96/213, DI-EPFL, November 1996.

[ALM 96b] Almesberger W, ATM on Linux, `ftp://lrcftp.epfl.ch/pub/linux/atm/papers/atm_on_linux.ps.gz`, EPFL, March 1996.

[ALM 97] RFC2170: Almesberger W, Le Boudec J-Y, Oechslin P, Application REQuested IP over ATM (AREQUIPA), IETF, July 1997.

[ARA 96] Arango M, Cortés M, Guaranteed Internet Bandwidth, *Proceedings of Globecom '96*, vol. 2, pp 862–866, November 1996.

[ARM 96] RFC2022: Armitage G, Support for Multicast over UNI 3.0/3.1 based ATM Networks, IETF, November 1996.

[ATK 94] RFC1626: Atkinson R J, Default IP MTU for use over ATM AAL5, IETF, 1994.

[ATM 95] The ATM Forum, Technical Committee, LAN Emulation Over ATM, Version 1.0, `ftp://ftp.atmforum.com/pub/specs/af-lane-0021.000.ps.Z`, The ATM Forum, January 1995.

[ATM 96] The ATM Forum, Technical Committee, ATM User-Network Interface (UNI) Signalling Specification, Version 4.0, `ftp://ftp.atmforum.com/pub/specs/af-sig-0061.000.ps`, The ATM Forum, July 1996.

[ATM 97] The ATM Forum, Technical Committee, Multi-Protocol Over ATM, Version 1.0, `ftp://ftp.atmforum.com/pub/approved-specs/af-mpoa-0087.000.ps`, July 1997.

[BER 91] Berners-Lee T, World-Wide Web - Summary, `http://www.w3.org/pub/WWW/Summary.html`, 1991.

[BER 96] RFC1945: Berners-Lee T, Fielding R T, Frystyk Nielsen H, Hypertext Transfer Protocol – HTTP/1.0, IETF, May 1996.

[BRA 89] RFC1122: Braden R, Requirements for Internet Hosts – Communication Layers, IETF, October 1989.

[BRA 94] RFC1633: Braden B, Clark D, Shenker S, Integrated Services in the Internet Architecture: an Overview., IETF, June 1994.

[BRA 97] RFC2205: Braden Bob (Ed), Zhang L, Berson S, Herzog S, Jamin S, Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, IETF, September 1997.

[CIS 97] Cisco, Advanced QoS Services for the Intelligent Internet, `http://www.cisco.com/warp/public/732/net_enabled/qos_wp.htm`, May 1997.

[CLA 92] Clark D, Shenker S, Zhang L, Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms, *Proceedings of ACM SIGCOMM '92*, pp 14-26, August 1992.

[COL 96] RFC1932: Cole R G, Shur D H, Villamizar C, IP over ATM: A Framework Document, IETF, April 1996.

[COX 96] Cox A, Network Buffers and Memory Management, *Linux Journal*, issue 30, October 1996.

[DIF] IETF Differentiated Services (diffserv) working group, `http://www.ietf.org/html.charters/diffserv-charter.html`

[HEI 93] RFC1483: Heinanen J, Multiprotocol Encapsulation over ATM Adaptation Layer 5, IETF, 1993.

[IEE] IEEE Std 802, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture.

[ITU 91] ITU-T Recommendation E.164/I.331, Numbering plan for the ISDN era, ITU, August 1991.

[LAU 94] RFC1577: Laubach M, Classical IP and ARP over ATM, IETF, 1994.

[LEB 92] Le Boudec J-Y, The Asynchronous Transfer Mode: a tutorial, *Computer Networks and ISDN Systems*, Volume 24, Number 4, 1992.

[LUC 98] Luciani J V, Katz D, Piscitello D, Cole B, Doraswamy N, NBMA Next Hop Resolution Protocol (NHRP), IETF, April 1998.

[OEC 96] Oechslin P, Web over ATM – Intermediate Report, `http://lrcwww/WebOverATM/rapport/rapport.html`, Technical Report 96/209, EPFL, October 1996.

[PER 95] RFC1755: Perez M, Liaw F-C, Mankin A, Hoffman E, Grossman D, Malis A, ATM Signaling Support for IP over ATM, IETF, 1995.

[RAC 92] RACE Project MAGIC, Commission of the European Communities, Final report, 1992.

[ZHA 95] Zhang L, Shenker S, Clark D, Huitema C, Deering S, Ferrari D, Reservations or No Reservations, `ftp://ftp.parc.xerox.com/pub/net-research/infocom95.html`, *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, April 1995, panel-discussion slides.