# Arequipa: TCP/IP over ATM with QoS ... for the impatient

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

January 23, 1997

**Abstract:** Many applications need dependable quality of service on the Internet. The approaches currently being devised by IETF and ATM Forum will provide general solutions for this, but their availability will be delayed by the requirement for prior ubiquitous deployment. We propose a much simpler approach that offers similar services to users who are connected to ATM, and that is available instantly, because it does not need any modifications in the network. We describe how we implemented our approach, how we apply it to the Web, and finally, how we tested it on an European ATM WAN.

## 1 Introduction

In order to transfer multi-media data and other time-critical data over networks, a dependable Quality of Service (QoS) is needed. This can be achieved either by dimensioning the network such that traffic is very unlikely to ever exceed the available resources, or by explicitly reserving resources for specific data flows ([1]). Based on past experience in telephony networks, most contemporary efforts for providing dependable QoS are targeted at the reservation approach.

The current Internet does not provide reservations, but the Internet Engineering Task Force (IETF) has identified the need for supporting integrated services already years ago ([2], [3]) 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 [4]) and the corresponding mappings to specific link layers, which are currently still in draft status. Unfortunately, even when the protocol and the supporting framework are finalized, it will still take a considerable amount of time until they are implemented and deployed in sufficiently large portions of the Internet to be useful for the average user.

On the other hand, the design of ATM ([5, 6]) considered reservation mechanisms from the very beginning. ATM networks therefore offer very reliable reservation mechanisms and well-understood traffic management concepts. With ATM being widely deployed at least in the wide area, access to a network with excellent QoS support becomes feasible for an increasing number of users.

Applications that are specifically written to use ATM, so-called *native* ATM applications, can use these mechanisms already today. However, the majority of network applications are written for TCP/IP. They cannot directly benefit from ATM's capabilities, and it may be difficult to convert them to native ATM applications.

In this article, we present Application REQuested IP over ATM (Arequipa), a mechanism that allows applications to establish direct point-to-point end-to-end ATM connections with given QoS at the link level. These connections are used exclusively by the applications that requested them. After setup of the Arequipa connection (i.e. the ATM SVC that

[*]Contact author; werner.almesberger@lrc.di.epfl.ch

is used for Arequipa), the applications can use the standard TCP/IP protocol suite (and its APIs) to exchange data.

The ability to control ATM connections allows users to obtain exactly the service they need from the network (e.g. ranging for some minimum bandwidth requirements to achieve acceptable response times to "hard" guarantees which are required for highly timing-critical applications) and it also gives them the option of selecting better but probably also more expensive networks or connections than the Internet would provide.

We believe that, today, Arequipa offers the most practical way of enabling users and applications to benefit from ATM.

Also, 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 [7]. 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 setup ATM connections as needed. We thus view the existing Internet as the ideal application level signaling for an ATM network.

This 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 IETF or ATM Forum. In sections 3 and 4, we explain the concept of Arequipa and how it can be implemented in a Unix-like operating system. Section 5 introduces a way of using Arequipa with the World-Wide Web (WWW, [8]). In section 6, we describe how we tested Arequipa to transfer video data across Europe. Finally, we conclude with section 7.

## 2 Transmitting IP over ATM

IETF and 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 IETF developed for running IP over ATM is the so-called *classical IP over ATM*, defined mainly in RFC1577 [9], but see also RFC1483 [10], RFC1755 [11], and RFC1932 [12]. 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.

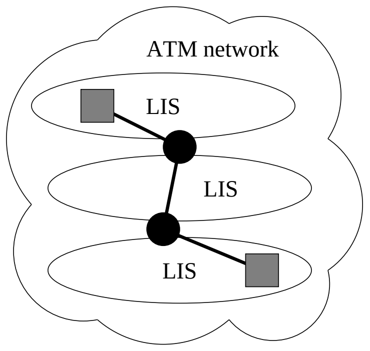

Figure 1: Classical IP over ATM has to use routers even if a direct ATM connection could be established between communicating hosts.

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, [13]) is ATM Forum's equivalent to classical IP over ATM. Like the latter, it limits direct ATM connections to comparably a small cloud of systems, the so-called Emulated LAN (ELAN). The main differences to classical IP over ATM are that LANE uses IEEE 802 [14] MAC addresses instead of IP addresses, and that is 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, [15]). This protocol tries to resolve ATM addresses for hosts which are not in the same LIS. Having 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.

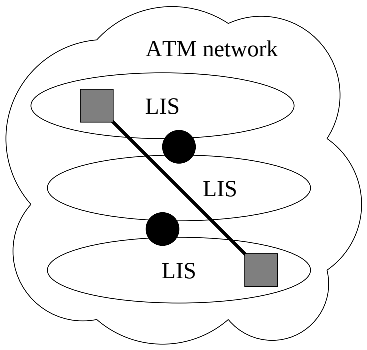

Figure 2: NHRP can establish direct end-to-end ATM connections between hosts.

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, [16]) merges protocols developed by IETF and 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 [17].

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 [18]) approaches the QoS issues by directing flows with QoS requirements over dedicated WAN connections (e.g. ISDN, 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 (i.e. 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. This is due to mismatched protocol layering. Since ATM is used below the IP layer and IP has no notion of connections or QoS, the interoperation mechanisms must hide those properties of ATM.

NHRP/MPOA and RSVP palliate that 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 a lot of complexity and suffers from the following restrictions:

- In order for NHRP and RSVP to be effective, they need to be deployed on all the nodes between communicating hosts. If this is not the case, NHRP will not be able to create an end-to-end connection between the hosts and RSVP will not be able to guarantee reservations on the entire path. 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's guaranteed QoS.

- 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, we show in the following section an approach that, by being slightly less general, avoids the aforementioned obstacles.

## 3 Arequipa

Arequipa is a mechanism which 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.

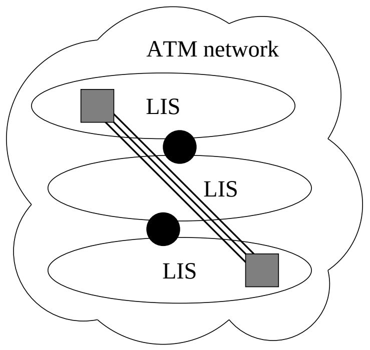

Figure 3: End-to-end Arequipa connections for three applications with QoS requirements.

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.

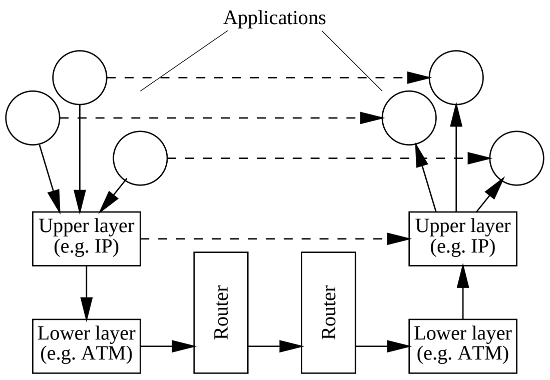

Figure 4: Communication without Arequipa.

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.

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:

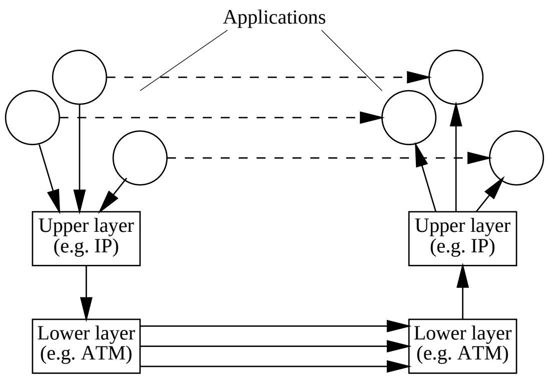

Figure 5: Communication with Arequipa.

- 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 endpoints (or they share at least a useful common subset of reachable endpoints)

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 endpoints

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.

## 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 Protocol changes

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 [11], 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 ([19]).

### 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.

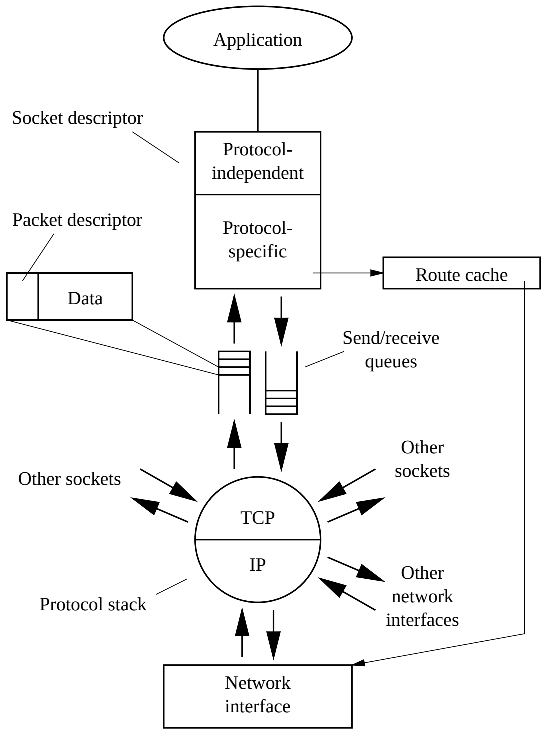

Figure 6: Kernel data structures of a TCP socket (simplified).

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.

## Data structures for incoming Arequipa

When using Arequipa, incoming packets are handled like when 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]

If the socket is not yet using Arequipa for sending (i.e. if it has no associated Arequipa VC) and if it expects incoming Arequipa traffic (i.e. 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.

The following subtle race condition needs to be considered: if a packet is received over an Arequipa VC, that VC may no longer exist 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).

This can be implemented as follows:

- all incoming Arequipa VCs are registered in a list (while they are dangling, i.e. while 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.

[1]The term "pseudo-interface" is used to make it clear that the Arequipa interface does not correspond to a physical network interface (i.e. hardware) although the protocol stack interacts with it as if it did.

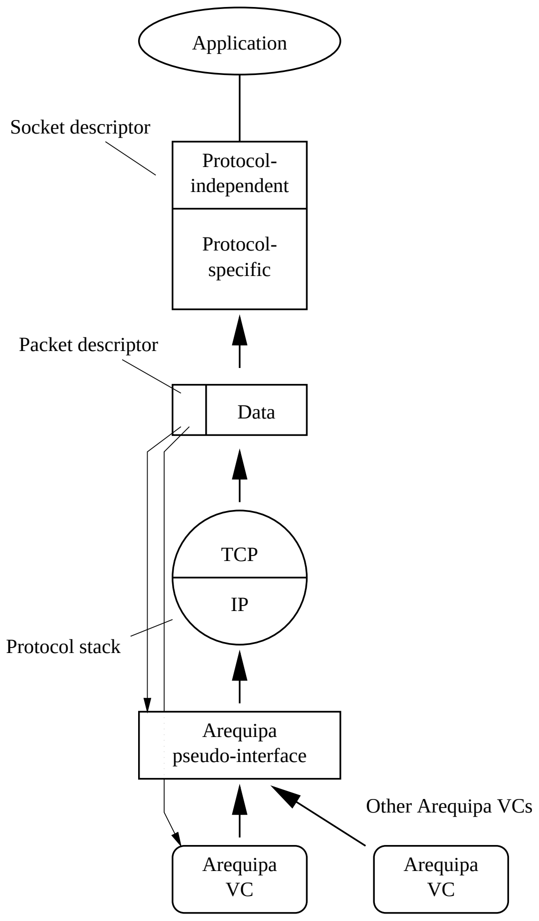

Figure 7: Arequipa for incoming data.

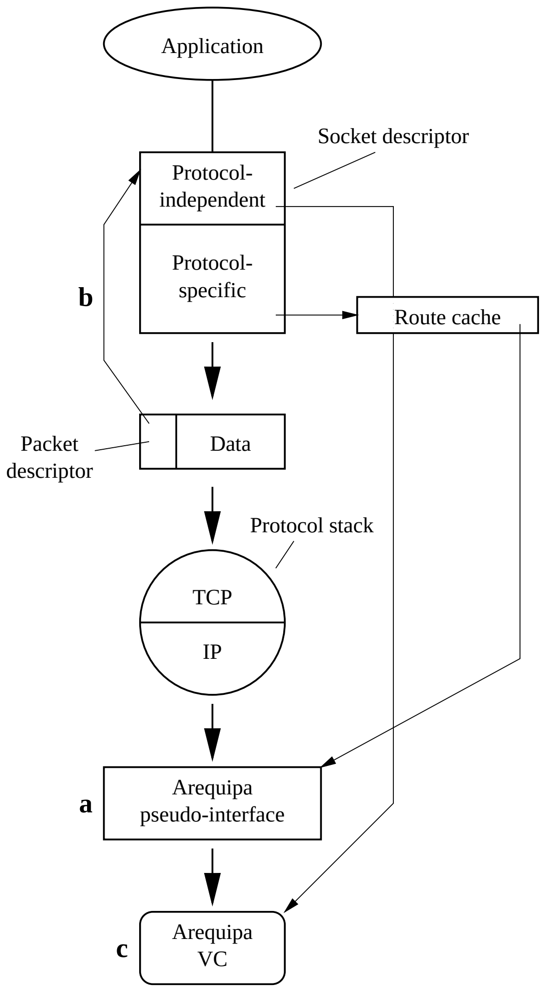

Figure 8: Arequipa for outgoing data.

## 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.

## 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 (i.e. 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 [20], 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.

### 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 ([21]), that MSS will have to be kept even if Arequipa is later used for that socket (see RFC1122 [22], 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 sides 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 [23].

## 5 Transmitting Web documents with guaranteed QoS

One of the most popular application on IP networks is the World Wide Web (WWW, [8]). 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 conveniently be sent as a pragma in the client's request ([24]). 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.

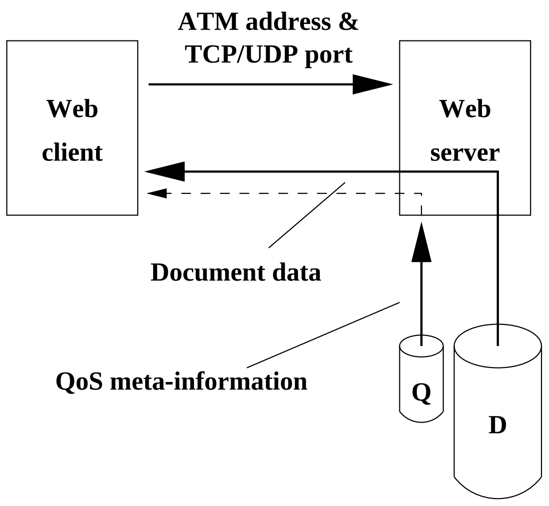

Figure 9: General information flow when using Arequipa with the Web.

Although not strictly required, the QoS information is also useful for 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, independent of if the document in question is eligible for being transferred with Arequipa or not. 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 [25] 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 endpoints. 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.

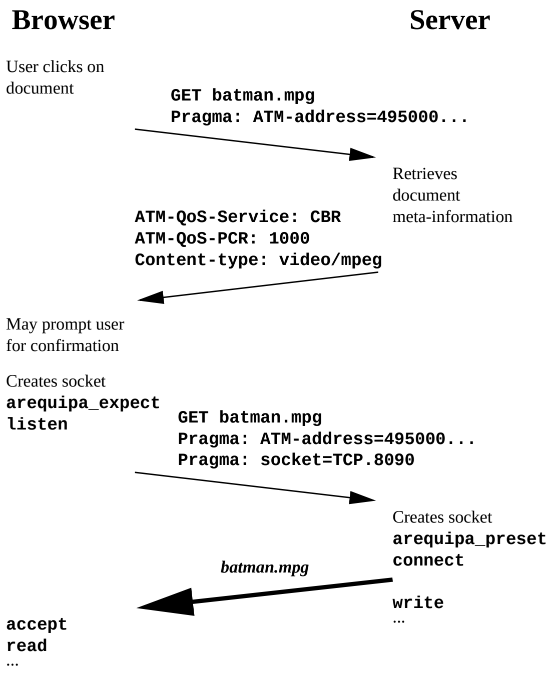

Figure 10: Example request/response flow when using Arequipa on the Web.

## 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.

## 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 (i.e. the typical situation in a LAN), it is sufficient if Arequipa is used only over the – possibly congested – WAN.

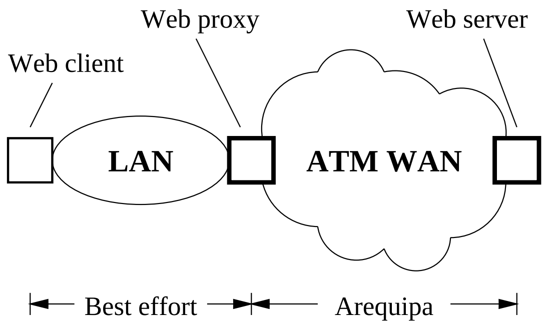

Figure 11: Arequipa with a proxy Web server.

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.

## 6 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 [26], which also comprises the work on Arequipa.

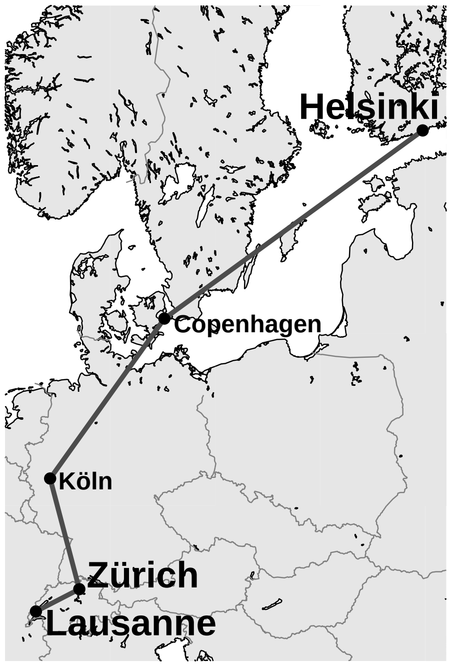

Figure 12: Arequipa test in European WAN.

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.

### 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/

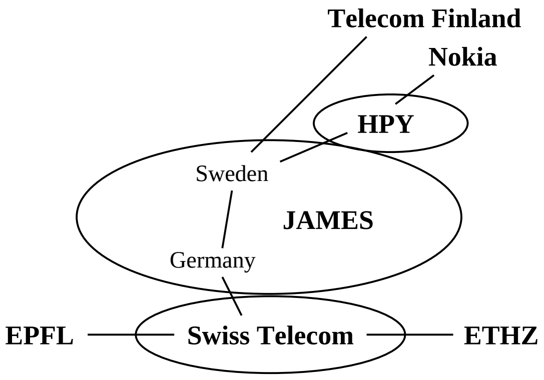

Figure 13: Schematic overview of the network structure.

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 Mbps). The connection with ETH was a virtual path with a bandwidth of approximately 34 Mbps.

Like 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 Mbps, a throughput of approximately 25 Mbps was obtained for video traffic from Finland.

Also, the throughput for TCP over Arequipa without application overhead was tested on the 34 Mbps virtual path with ETH. This benchmark was done with `ttcp`, a program that sends/receives to/from memory without doing any further data processing. With traffic shaping set to 33.3 Mbps, we obtained a throughput of up to 33.0 Mbps.

Arequipa enables one given application to receive a specified throughput. 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 predict with the normal Internet.

## 7 Conclusions

We have illustrated the need for dependable quality of service in the Internet and summarized current efforts by IETF and ATM Forum which aim to provide a long-term solution for this.

We have presented Arequipa, a more specialized but functionally similar approach that has less implementation complexity and that does not require deployment of a supporting server infrastructure in the network, thereby allowing users the immediate use of QoS for Internet traffic. We have also shown how Arequipa can be implemented on Unix systems.

We have described a way of using Arequipa with the Web without sacrificing compatibility with standard Web browsers and servers, and have indicated how Arequipa can be of use even for hosts not directly connected to ATM. Finally, we have presented the results of a test of Arequipa in an European WAN setup.

## 8 Available software

An implementation of the Arequipa mechanisms is part of the ATM on Linux ([27]) 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

[1] Zhang, Lixia; Shenker, Scott; Clark, Dave; Huitema, Christian; Deering, Steve; Ferrari, Domenico. *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.

[2] 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.

[3] RFC1633; Braden, Bob; Clark, David; Shenker, Scott. *Integrated Services in the Internet Architecture: an Overview.*, IETF, June 1994.

[4] Braden, Bob; Zhang, Lixia; Berson, Steve; Herzog, Shai; Jamin, Sugih. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification* (work in progress), Internet Draft `draft-ietf-rsvp-spec-13.ps`, August 1996.

[5] Le Boudec, Jean-Yves. *The Asynchronous Transfer Mode: a tutorial*, Computer Networks and ISDN Systems, Volume 24, Number 4, 1992.

[6] 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.

[7] RACE Project MAGIC. *Commission of the European Communities*, Final report, 1992.

[8] Berners-Lee, Tim. *World-Wide Web - Summary*, `http://www.w3.org/pub/WWW/Summary.html`, ≈ 1991.

[9] RFC1577; Laubach, Mark. *Classical IP and ARP over ATM*, IETF, 1994.

[10] RFC1483; Heinanen, Juha. *Multiprotocol Encapsulation over ATM Adaptation Layer 5*, IETF, 1993.

[11] RFC1755; Perez, Maryann; Liaw, Fong-Ching; Mankin, Allison; Hoffman, Eric; Grossman, Dan; Malis, Andrew. *ATM Signaling Support for IP over ATM*, IETF, 1995.

[12] RFC1932; Cole, Robert G.; Shur, David H.; Villamizar, Curtis. *IP over ATM: A Framework Document*, IETF, April 1996.

[13] 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.

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

[15] Luciani, James V.; Katz, Dave; Piscitello, David; Cole, Bruce. *NBMA Next Hop Resolution Protocol (NHRP)* (work in progress), Internet Draft `draft-ietf-rolc-nhrp-10.txt`, October 1996.

[16] The ATM Forum, Multiprotocol Sub-Working Group. *MPOA Baseline Version 1*, `ftp://ftp.atmforum.com/pub/mpoa/baseline.ps`, September 1996.

[17] RFC2022; Armitage, Grenville. *Support for Multicast over UNI 3.0/3.1 based ATM Networks*, IETF, November 1996.

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

[19] Cox, Alan. *Network Buffers and Memory Management*, Linux Journal, issue 30, October 1996.

[20] RFC *to appear*; Almesberger, Werner; Le Boudec, Jean-Yves; Oechslin, Philippe. *Application REQuested IP over ATM (AREQUIPA)*, IETF, October 1996.

[21] RFC1626; Atkinson, Randall J. *Default IP MTU for use over ATM AAL5*, IETF, 1994.

[22] RFC1122; Braden, R. *Requirements for Internet Hosts – Communication Layers*, IETF, October 1989.

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

[24] RFC1945; Berners-Lee, Tim; Fielding, Roy T.; Frystyk Nielsen, Henrik. *Hypertext Transfer Protocol – HTTP/1.0*, IETF, May 1996.

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

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

[27] Almesberger, Werner. *ATM on Linux*, `ftp://lrcftp.epfl.ch/pub/linux/atm/papers/atm_on_linux.ps.gz`, EPFL, March 1996.