Using Quality of Service can be simple: Arequipa with Renegotiable ATM connections [1]

Werner Almesberger[†], Leena Chandran-Wadia[†], Silvia Giordano[†]
Jean-Yves Le Boudec[†], Rolf Schmid[‡]

† *ICA (Institute for computer Communications and Applications), EPFL - Lausanne, Switzerland*
‡ *ASCOM Tech. Ltd, Bern, Switzerland*

Abstract

We demonstrate how the Quality of Service selection capability of ATM networks can be made available to IP applications in a simple way. To show this, we have modified the popular Mbone tool *Vic* (VIdeo Conferencing) to use *Arequipa* (Application REQuested IP over ATM). The latter enables applications and in particular *Vic*, to request a direct ATM connection for its exclusive use and to directly control the traffic parameters of this connection. We have also implemented ATM Forum's UNI4.0 signaling and ITU-T's connection modification recommendation Q.2963.1, on end-systems as well as on switches. This implementation, coupled with the *Arequipa* mechanism, allowed *Vic* to negotiate and renegotiate ATM bandwidth at will, at run time. We describe how *Vic* accomplished this over the ATM WAN of SWISSCOM, transferring live video from Lausanne to Basel and Zürich over switched, renegotiable ATM connections. To our knowledge, this was the first time any application had the capacity to tune ATM traffic parameters at run time.

Key words: IP, ATM, IP over ATM, Arequipa, QoS, signaling, renegotiation, Vic

1 Introduction

ATM networks [2,3] offer a rich set of features to support guaranteed quality of service (QoS), including several traffic and QoS [1] parameters which can be

[1] work done in collaboration with the ACTS EXPERT project - AC094 [1]

[1] We use QoS to refer to quality of service in general but sometimes also to ATM connection QoS parameters. The meaning will always be clear from the context.

set by the user. Despite this rich structure, the main body of applications today still use ATM only via the IP layer, and thus do not benefit from the possibilities for QoS selection offered by ATM.

We have been studying ATM networks with a view to making the QoS selection capability visible to the user, and to do this in as simple a way as possible. Among the many choices for the QoS parameters we believe that, from the users' point of view, it is appropriate to restrict that choice to just the bandwidth, or ATM peak cell rate (PCR). In this paper we describe how we can satisfy the requirements of a real time service for IP applications by running them directly over ATM. The application we have chosen as an example is the popular Mbone Tool *Vic* (Video Conferencing). We have modified *Vic* to make the QoS control available to the user, at the touch of a button, through a mechanism developed earlier by us namely *Arequipa*, by which IP applications can have the option of selecting QoS parameters.

The ATM Forum and the ITU [4,5] define a large number of ATM connection types, ranging from constant bit rate (CBR) to Available bit rate (ABR). In the work reported here, we focus on the use of *renegotiable CBR* connections, because it provides a simple means to offer a visible quality of service, under explicit control from the end-user. Renegotiable CBR connections are connections with a maximum peak cell rate, which can be modified by the user at any time (see Section 2).

Arequipa or Application REQuested IP over ATM [6,7] is a method for providing the quality of service of ATM to TCP/IP applications, assuming end-to-end ATM connectivity exists. The *Arequipa* mechanism does not require any changes in the network but two changes need to be made at the end systems. First, in order to implement *Arequipa* , some changes need to be made to the TCP/IP protocol stack (on the end systems). Second, applications using *Arequipa* to obtain QoS need to be modified slightly to make use of our simple extension to the socket interface. The latter consists of just four calls for setting up, tearing down and modifying the traffic parameters of connections. The applications then make use of the end-to-end ATM connectivity and *Arequipa* to set up a direct switched virtual channel connection (SVC) from the sender to the receiver application (Fig. 1).

The first applications to be made Arequipa-capable were the Arena Web browser and the CERN httpd Web server [6,7]. These were used to transfer live video across a trans-european WAN from Lausanne to Helsinki with guaranteed QoS [8]. Since then we have partially implemented UNI4.0 signaling and the Q.2963.1 connection modification capability on the end systems and on switches. Simultaneously we have extended the *Arequipa* mechanism and API to include the connection modification capability. With these an application can now modify the QoS parameters at run time, *after* connection

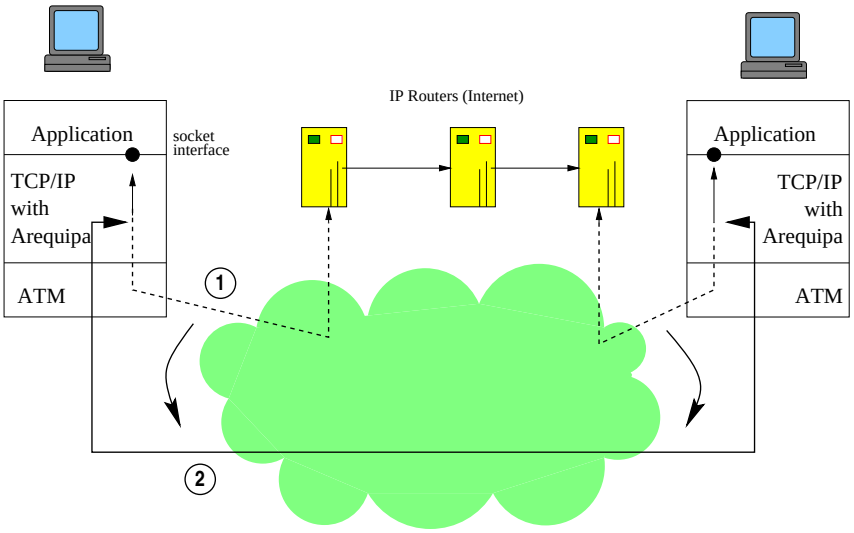

Fig. 1. *Arequipa* capable applications: data transmission is switched from the default IP path (1) to a dedicated ATM connection(2). An *Arequipa* connection between two ATM attached hosts bypasses intermediate IP routers completely

setup.

We demonstrated this in the case of *Vic* which was used [9] to video conference with QoS, in point-to-point mode, between Basel and Lausanne and also between Zürich and Lausanne. We describe details of this demo which was done over the ATM WAN of SWISSCOM. To our knowledge this was the first time any application had the ability to tune ATM traffic parameters at run time.

Arequipa only works in situations where end-to-end ATM connectivity exists, but this assumption is not unrealistic in Europe where many countries have extensive ATM already deployed, not just in the backbones but also in the LANs.

The following section describes the main features of UNI4.0 signaling and Q.2963.1 connection modification capabilities. Sections 3 and 4 describe *Arequipa* and *Vic* respectively, while sections 5 and 6 describe details of implementation issues and the demo.

2 Renegotiable CBR Connections

ATM signaling is used by applications to set up SVCs with prescribed QoS. As mentioned in the introduction, we focus here on ATM connections of the CBR class with renegotiation, which we call renegotiable CBR. We refer to renegotiation as the connection modification capability defined in the ITU recommendation Q.2963.1. The latter relates to modifying the traffic parameter

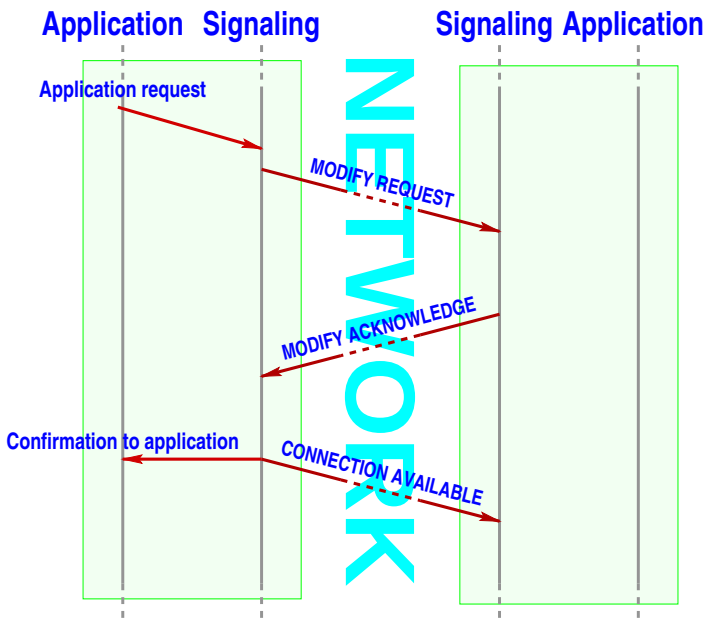

Fig. 2. Message sequence during bandwidth renegotiation

of an already active CBR connection. The only connection characteristic that can be modified according to Q.2963.1 is the peak cell rate (PCR).

In order to change the PCR of an active connection without renegotiation, the only possibilities are to (1) open a new connection and to close the old one once the new connection is available, or (2) to close the old connection first and to open the new one afterwards. In (1), the sum of the old and the new bandwidth is allocated for a moment, which may cause the modification to be rejected, although the new PCR alone would be acceptable. Also, data can reach the destination on two distinct paths, so the sequence of cells is no longer guaranteed and some synchronisation is required. In case (2), connectivity is interrupted for a short moment, and, if the new PCR cannot be supported by the ATM network, the connection may be lost entirely.

Renegotiation has neither of those drawbacks and also has shorter latency and less processing, as the modification messages are small and the ATM network only needs to perform connection admission control but no addressing, routing or other processing required for call establishment.

The exchange of signaling messages during bandwidth renegotiation between two applications is shown in Fig. 2. When an application requests a modification, a MODIFY REQUEST is sent out to the called user provided local resources are available. Similarly a MODIFY ACKNOWLEDGE message is returned to the calling user only if resources are available at the called user. The local resources at the calling user are rechecked before the modification is considered to be agreed upon. Then a confirmation is sent to the application and the peak cell rate is changed. A renegotiation requesting a rate increase

may fail, in which case the peak cell rate remains at its previous value.

Connection modification is going to be an important capability in future commercial broadband networks, where users will want to control the speed and quality of the data stream because they will be charged for network usage (e.g. bandwidth, time). Modification is especially suited for interactive multimedia applications where bandwidth usage is likely to vary considerably over time. Renegotiable CBR service can be used when there are infrequent and coarse-grained scale changes in the bandwidth requirements during the connection life time. In a video-conferencing scenario a typical example is the situation where the video moves from the image of a speaker to the contents of transparencies.

Note that renegotiation is different from the initial QoS negotiation. The latter capability (defined in UNI 4.0) takes place at call setup time, and consists in allowing network nodes, or the called user, to negotiate the traffic descriptor requested by the calling user. Contrary to negotiation, re-negotiation occurs after the connection is set up.

Renegotiable CBR connections also differ from Available Bit Rate (ABR) connections [10]. With ABR connections, the maximum cell rate (allowed cell rate) is also variable, but here the value of this rate is dictated by the network, not the user. Also, with ABR connections, there is a concept of minimum cell rate. Extension of our work to ABR connections would be interesting but still remains to be done. At the same time renegotiable CBR service differs from VBR services. VBR is better suited for bursty traffic while renegotiable CBR service is useful when there are a small number of drastic changes in the QoS.

3 Arequipa

3.1 Overview of Arequipa

The first step in running IP over ATM is to have a means to carry IP packets on ATM. This is mainly an encapsulation issue, defined in RFC 1483 [11]. With this alone, IP can be run over ATM using PVCs. For SVCs, a way to resolve IP addresses to ATM addresses is needed. The IETF currently uses an approach called "classical IP over ATM" (CLIP) that is based on an extension of ARP called ATMARP [12]. The ATM Forum has defined a similar service called "LAN emulation" (LANE) [13] which tries to provide exactly the functionality one would obtain from a LAN, say an Ethernet. Neither CLIP nor LANE are designed to allow applications to benefit from the inherent QoS selection features offered by the underlying ATM network [14].

As described in section 1, *Arequipa* is a method for allowing ATM-attached hosts that have direct ATM connectivity to set up end-to-end IP over ATM connections, within the reachable ATM cloud, on request from applications and for the exclusive use by the requesting application. These applications use both an IP and an ATM stack to obtain direct ATM connections (SVCs) with guaranteed QoS. The QoS is guaranteed by the fact that each of these SVCs is used exclusively for one IP flow identified by a pair of sockets (eg. a TCP connection or a UDP stream). Currently *Arequipa* only supports unicast (point-to-point) operation. Extending Arequipa to support multicast would be straightforward, but it would burden the application with session management tasks if ATM multicast modes other than leaf-initiated join were to be supported.

Arequipa does not require any modifications in the networks (routers, switches etc.) but as mentioned earlier, two important changes need to be made at the end-systems. Some changes need to be made to the TCP/IP stacks at the end systems (discussed briefly in section 5.3 and in detail in [7]) and applications need to be modified to use the *Arequipa* socket extensions. It is important to note that if an application is to be QoS aware at all, some code needs to be added somewhere, either in the applications themselves or in some proxies or gateways. We argue that modifying *Vic* to use *Arequipa* is not more complex than modifying it for RSVP [15,16], and certainly less so than modifying it for native ATM [17].

Arequipa coexists with "normal" use of the networking stacks so that applications not requiring *Arequipa* need not be modified and continue to function as normal.

3.2 Functionality

Arequipa adds four simple new functionalities at the socket layer. Applications need to be modified to use just the following four calls in order to set-up, tear-down or modify their ATM SVCs.

- `arequipa_preset(socket, atmaddr, qos)`: establishing or preparing establishment of a new link-layer ATM connection to a given address with a given ATM service and QoS, to make sure that further data sent on the specified socket, and only data sent on that socket, will use the new ATM connection. arequipa_preset sets up a bidirectional VCC, symmetric or asymmetric, and is only applicable to connection oriented sockets (eg. TCP or connected UDP sockets).
- `arequipa_expect(socket, {true, false})`: preparing a socket to use an incoming *Arequipa* connection for all its outgoing traffic. When a socket

receives data from an *Arequipa* connection and arequipa_expect has been set to true, the socket is set to send all its data over the *Arequipa* connection. Again, `arequipa_expect` is only applicable to connected sockets.

- `arequipa_close(socket)`: implicit or explicit closing of *Arequipa* connections. An *Arequipa* connection can be explicitly closed using `arequipa_close` or implicitly closed when the corresponding socket is closed.
- `arequipa_renegotiate(socket, newqos)`: renegotiation of existing *Arequipa* connections. The QoS of an *Arequipa* connection can be modified (increased or decreased) using `arequipa_renegotiate`. The QoS is not modified until the modification is agreed upon.

The `arequipa_preset` and `arequipa_expect` calls are usually made as soon as the application opens sockets for network I/O. The `arequipa_renegotiate` call needs to be made every time the traffic parameters of the *Arequipa* connection are modified.

4 VIC

The Mbone [18,19] VIdeo Conferencing tool *Vic* [20] has a flexible system architecture characterised, among other things, by network layer independence and an extensible user interface. We have used these two features in order to enable *Vic* to use *Arequipa*. The user interface has been modified to include elements for *Arequipa* control and a new network module has been added to set up the appropriate *Arequipa* connections when necessary.

The *Vic* distribution [17] contains three separate network modules for normal IP, native ATM and RTIP (Real-Time IP). Only one of these can be linked at a time, in the current version of *Vic*. The new "*Arequipa*" network module we have added defaults to the normal IP module if *Arequipa* fails for any reason.

As mentioned earlier, currently *Arequipa* only supports unicast communication using standard unicast IP addresses and connected UDP sockets. Thus we only run *Vic* in the point-to-point mode, even though it was primarily intended as a multi-party conferencing application on the Mbone. For such point-to-point video transfer, both the sender and receiver of video need to know each others' IP address and to agree on a port number. This is normally settled out of band, just as in the case of the IP network module.

4.1 User Interface

The main window of *Vic* containing thumbnail views of the outgoing (loop-back) video as well as the incoming video is shown on the left side of Fig. 3. Each thumbnail picture is accompanied by identification text, frame and bit rate statistics, and a loss indicator (in parenthesis). The latter is inferred from sequence numbers of the incoming packets. Packets are lost either due to network drops or due to local socket buffer overflows resulting from CPU saturation. Fig. 3 is taken in Basel, using the new "*Arequipa*" network module but before starting *Arequipa*. It shows a loss rate of the incoming video from Lausanne (over the Internet) to be well over 50%, even at relatively low bit rates.

Details of the *Vic* control panel (obtained by clicking on the menu button in the main window) are shown on the right side of Fig. 3. We have extended the menu by adding a section on top for *Arequipa* control. By design only the sender of video can initiate an *Arequipa* session. This is done by selecting the service category as well as the bandwidth (converted internally into PCR) and clicking on the *Arequipa* button [2].

In line with the spirit of keeping the visible QoS simple the only user specifiable traffic parameter provided is the bandwidth, and CBR/UBR the only choice of service categories. Once an *Arequipa* connection is in place, it it not possible to change the service category (e.g. from CBR to UBR), but only to change the traffic parameter. Each time the content of the bandwidth entry is modified by the user, a call to `arequipa_renegotiate` is made automatically.

The capture (and encode) frame and bit rates are displayed above the QoS settings. The close button only closes the *Arequipa* connection so video transmission continues via the normal IP path. As in the unmodified *Vic* the release button has to be used to stop video transmission.

Arequipa can be started any time during (IP) video transmission, e.g. when picture quality deteriorates. If normal IP transmission has not already begun when *Arequipa* is started then *Vic* first starts it before switching over to *Arequipa*. Then, if *Arequipa* fails for any reason, transmission reverts to the normal Internet path.

[2] note that the *Arequipa* mechanism itself does not carry any constraint as to which side can initiate *Arequipa*.

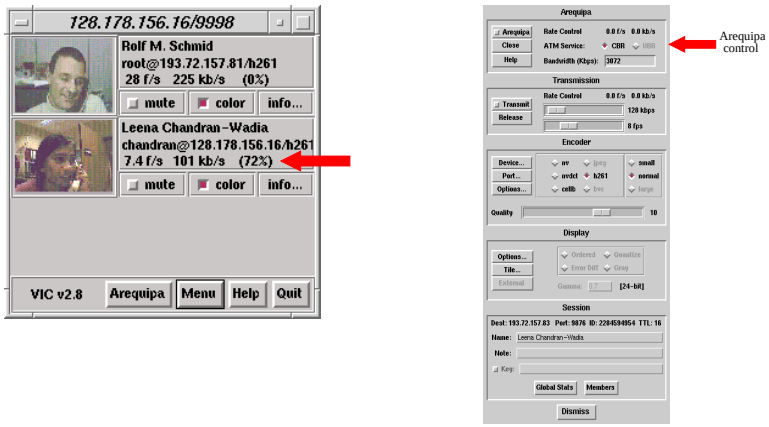

Fig. 3. Main window and control panel of *Arequipa* capable *Vic*

4.2 Networking

As described in detail in [20] *Vic* uses the Real-time Transport Protocol, RTP [21], which is realised completely within *Vic* itself. RTP is divided into two components: the data delivery protocol and the control protocol RTCP. The former handles the actual media transport and the latter manages control information like sender identification, receiver feedback and cross-media synchronisation.

We have added a new network module which contains, apart from the normal IP module, procedures for the exchange of ATM addresses, for opening, closing and renegotiating *Arequipa* connections. Note that in order to make the connection, the `arequipa_preset` call requires the knowledge of the ATM address of the peer machine (section 3.2. Details of the implementation of a procedure for the exchange of ATM addresses depends very much on the application itself e.g., in the case of a Web browser and a Web server we used HTTP for the address-exchange, see [7].). In the demo version of *Vic* this exchange was done out of band. A newer version contains a generic address-exchange module in which the exchange is performed over TCP when *Arequipa* is first requested. So far *Arequipa* connections have only been set up for the data channel. We decided against using an *Arequipa* connection for control traffic too because the latter, being low volume, would typically only benefit from an *Arequipa* connection in extreme high-loss situations, say caused by severe congestion, which should be rare.

5 Implementation

ATM signaling support, including *Arequipa* and the connection modification capability at the end systems has been implemented for PC's running the

Linux operating system [22]. Signaling and connection modification has also been implemented on the ATMLightRing switch of ASCOM.

5.1 ATMLight Ring (ASCOM)

The ATMLightRing is a campus and metropolitan area backbone developed by ASCOM. Physically, it consists of a dual-fiber ring interconnecting a number of Access Nodes across a campus or city area. Each node provides standard ATM and non-ATM user interfaces to which various communication equipment with standard interfaces can be connected.

Logically, an ATMLightRing is a high-speed transport backbone providing full connectivity at each port. It appears to the network as a single distributed switch, thereby greatly reducing system management complexity. It allows interconnection of switches, routers, hubs, concentrators, servers, PBXs, workstations and WAN access devices.

For the experiments with renegotiable ATM connections, the control software of the ATMLightRing was enhanced to support the UNI4.0 signalling capabilities together with the Q.2963.1 connection characteristics negotiation and modification.

5.2 ATM on Linux

ATM on Linux is a comprehensive implementation of ATM-related protocols, including the latest signaling as specified in ATM Forum UNI 4.0. This platform is also used for the reference implementation of *Arequipa* (see below) and for experiments with renegotiation as specified in Q.2963.1.

The ATM on Linux distribution containing full source code for kernel changes, system programs and test application is available publicly [22].

5.3 Arequipa

Arequipa has also been implemented in the Linux operating system and is part of the ATM on Linux distribution [23] of ICA. For the establishment and the release of ATM connections the signaling parts of classical IP over ATM have been reused. A new virtual network device for *Arequipa* has been created and a few modifications have been made to the socket layer.

The implementation, detailed in [24] builds upon the route cache entry in the socket descriptors. This entry stores a pointer to the interface to which all data sent from a socket has to be forwarded. This is normally done to avoid doing an IP route look-up every time a datagram is sent. By setting and locking this route cache to point to the *Arequipa* device it is ensured that any further data sent from the socket goes to the *Arequipa* device. An additional field has been added to the socket descriptor to store a pointer to the VCC on which the data should be transmitted. When it receives a datagram from IP, the *Arequipa* device simply sends the datagram on the VCC indicated in the socket descriptor.

`arequipa_preset` is implemented as a library function which does the following. First it asks the signalling demon to establish a VCC with a given destination and QoS. Then it enters a pointer to that VCC in the socket descriptor and makes the IP route cache of the socket point to the *Arequipa* device.

`arequipa_expect` simply sets a variable to indicate whether the application wants to use an incoming *Arequipa* connection for its outgoing traffic. Every time a datagram is received on an *Arequipa* VCC, the variable is tested. If it is set, the route cache is set to point to the *Arequipa* device.

`arequipa_renegotiate` requests renegotiation for the VCC attached to the specified socket, using the common native ATM procedures. It blocks until the renegotiation completes (with or without success). Other processes or threads can continue sending and receiving on the socket while `arequipa_renegotiate` is in process. Note that there is no explicit notification for renegotiation initiated by the peer.

6 Demonstration over an ATM WAN

Vic was used to transfer live video from Lausanne to Basel and Zürich over SWISSCOM's public ATM network. The demo was a cooperative effort between the Web over ATM project [25] of the EPFL and the ACTS-EXPERT project funded by the European Commission, and was conducted on 24th October 1997.

6.1 Setup

The demonstration network consisted of three ATM equipped Linux terminals located in Lausanne, Basel and Zürich, connected to a (two node) ATM-LightRing System at the EXPERT testbed in Basel. The three sites were

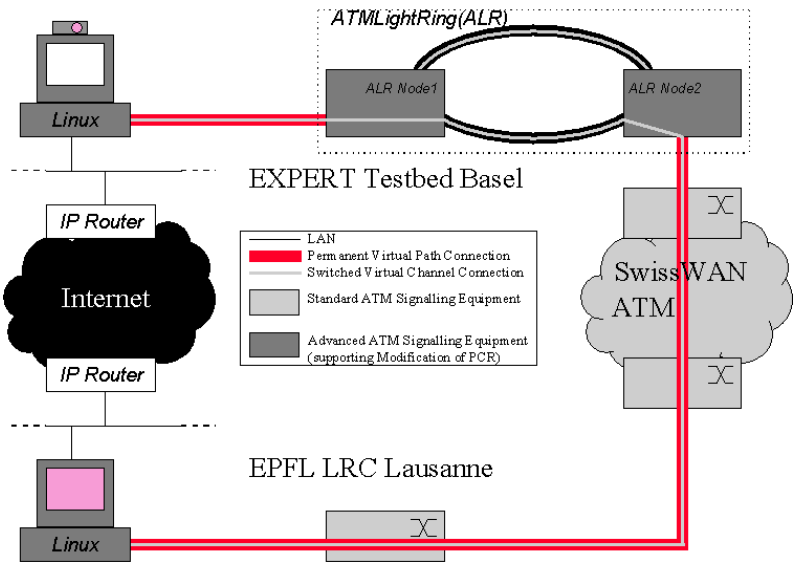

Fig. 4. Demo network configuration

interconnected by the VP connection provided by SWISSCOM's public network. The demonstration network between Basel and Lausanne is illustrated in Fig. 4. None of the intermediate switches except the ATMLightRing supported the UNI4.0 and the Q.2963.1 signaling capabilities. The VP connection from SWISSCOM provided seamless connectivity between the end systems and the ATMLightRing.

6.2 Observations

As shown by the loss indicator of the incoming video in Fig. 3, the loss rates over the Internet can often be well over 50% even at relatively low bandwidth. Switching over to *Arequipa* at such times results in a dramatic change in the quality of the video, because then network losses in congested routers no longer occur. At low bandwidths the loss rate drops to zero immediately. At much higher bandwidths losses make their appearance once again, this time due to CPU saturation. The threshold at which losses appear depend on the system configuration of course, but also on the type of encoding being used.

The ATMLightRing was used to test the connection-renegotiation feature several times during the Demo. There is a visible and distinct change in the quality of the picture when the bandwidth of the ATM SVC is tuned, even in a low movement conferencing scenario. This change, though less dramatic than the change observed when *Arequipa* is first turned on and trafffic is switched from the internet into ATM, is remarkable nevertheless. As mentioned earlier, we believe that the renegotiation feature will really come into its own for applications which require infrequent but sizeable changes in the quantity of bandwidth required from time to time.

7 Discussion and Summary

We have shown that in areas where end-to-end ATM connectivity already exists, there is a relatively simple way for IP applications to use the QoS of ATM, namely, by using *Arequipa* and its simple API. We demonstrated this in the case of *Vic* which also became the first instance of an application to tune bandwidth at run time.

We argue that in some cases applications can benefit from renegotiable CBR service, especially when a single connection is used for video transfer in which distinct sequences with different QoS requirements are identifiable. For instance a tele-teaching session where the image of the speaker is followed by illustrations. On the other hand if large changes in resources requirements occur frequently, then renegotiable CBR is probably not the appropriate service, from a pure billing point-of-view.

With end-to-end ATM connectivity, there is an alternate way of providing hard QoS guarantees to applications. This is to specifically enable them to use ATM, resulting in a so-called *native* ATM application. In the case of *Vic*, this has been done and *Vic* can run on ATM using the FORE SPANS API. In general, the effort involved in converting an IP application into a native ATM one can often be significant. In the case of *Vic* this was certainly so. The native ATM network module is completely different from the normal IP module whereas the *Arequipa* network module is merely an extension of it, using the same sockets and so on.

The IETF way of providing quality of service (QoS) guarantees to applications in IP networks today is to use the ReSerVation Protocol (RSVP) [15]. As with *Arequipa* and with native ATM, applications need to be modified in order to make them QoS sensitive. *Vic* [20] has also been RSVP enabled [16] and can provide soft guarantees on the QoS depending on the extent of RSVP deployment on the intermediate routers. With the work reported here, we believe to have demonstrated that presenting explicit quality of service to the application and their user is indeed simple and can be deployed as soon as networks exist which support reservations.

Some other issues related to what we presented in this paper are a matter of ongoing work. Among them:

(1) implicit QoS negotiation versus explicit QoS negotiation. In the implicit QoS negotiation context we also deal with policing and admission control issues as well as multicast [26,27]
(2) solutions for QoS renegotiation over heterogeneous networks (e.g networks using ATM, RSVP or SRP protocols) involving horizontal and vertical mapping among the different protocols used to obtain QoS [28]

Acknowledgement

We would like to thank Rainer Burki, Paul Hurley, Philippe Oechslin and Anne Possoz for many interesting and useful discussions during the course of this work. We are grateful to Alan Cox, Mireille Goud, Paco Hope, Jörg Liebherr, Koji Okamura and Petar Vrdoljak for help with aspects of the demo. We would also like to thank SWISSCOM for the use of the WAN and ASPA Basel for support and use of the EXPERT testbed.

References

[1] EXPERT Consortium, *EXPERT AC094 home page*, 1997. http://www.elec.qmw.ac.uk/expert/intro.html.

[2] J.-Y. Le Boudec, "The Asynchronous Transfer Mode : A Tutorial," *Computer Networks and ISDN Systems*, vol. 24 (4), pp. 279–309, May 1992.

[3] A. Alles, "Interworking with ATM," *InterOp proceedings*, 1995.

[4] The ATM Forum, *ATM User-Network Interface (UNI) Signalling Specification, Version 4.0*, 1996. ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.000.ps.

[5] ITU Telecommunication Standardization Sector - Study group 13, *ITU Recommandation Q.2931, Broadband Integrated Services Digital Network (B-ISDN) – Digital subscriber signalling system no. 2 (DSS 2) – User-network interface (UNI) – Layer 3 specification for basic call/connection control*, 1995.

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

[7] W. Almesberger, J-Y. Le Boudec, Ph. Oechslin, "Arequipa: TCP/IP over ATM with QoS ... for the impatient," Technical Report 97/225, DI-EPFL, CH-1015 Lausanne, Switzerland, January 1997. ftp://lrcftp.epfl.ch/pub/arequipa/impatient.ps.gz.

[8] P. Oechslin-(Editor), "Web over ATM," Tech. Rep. 96/209, DI-EPFL, October 1996.

[9] LRC-EPFL, L. Chandran (Editor), *Web over ATM: Intermediate Report*, 1997. http://lrcwww.epfl.ch/WebOverATM/finaldemo.html.

[10] The ATM Forum, *Traffic Management ABR Addendum*, 1997. ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0077.000.ps.

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

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

[13] The ATM Forum, *LAN Emulation Over ATM, Version 1.0*, January 1995. ftp://ftp.atmforum.com/pub/approved-specs/af-lane-0021.000.ps.

[14] S. Giordano, R. Schmid, R. Beeler, H. Flinck, J.-Y. Le Boudec, "IP and ATM - current evolution for integrated services ," Technical Report 98/2, DI-EPFL, CH-1015 Lausanne, Switzerland, January 1998.

[15] R. Braden and L. Zhang and S. Berson and S. Herzog and S. Jamin, *RFC2205: Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.* IETF, September 1997.

[16] S. Berson, "RSVP enabled Vic," , 1996. ftp://ftp.isi.edu/rsvp/release.

[17] Vic-Distribution, *VIdeo Conferencing*, 1997. ftp://ee.lbl.gov/conferencing/vic.

[18] H. Eriksson, "Mbone: The multicast backbone," *Commun. of the ACM*, pp. 54–60, Aug. 1994.

[19] V. Kumar, *MBone: Interactive Multimedia on the Internet.* Indianapolis, IN: New Riders, 1996.

[20] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," *ACM Multimedia*, 1995.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RFC1889: RTP: A Transport Protocol for Real-Time Applications.* IETF, 1996.

[22] W. Almesberger, *ATM on Linux Distribution.* EPFL, 1997. ftp://lrcftp.epfl.ch/pub/linux/atm/dist.

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

[24] W. Almesberger, "Arequipa: Design and Implementation," Technical Report 96/213, DI-EPFL, CH-1015 Lausanne, Switzerland, November 1996.

[25] P. Oechslin, *Web over ATM.* EPFL, 1997. http://lrcwww.epfl.ch/WebOverATM.

[26] W. Almesberger, T. Ferrari, J.-Y. Le Boudec, "SRP: a Scalable Reservation Protocol for the Internet," Technical Report 98/009, DI-EPFL, CH-1015 Lausanne, Switzerland, March 1998. ftp://lrcftp.epfl.ch/pub/people/almesber/srp/srpMar98.ps.gz.

[27] W. Almesberger, S. Giordano, P. Hurlay, J.-Y. Le Boudec, "From Arequipa to SRP," technical report, DI-EPFL, CH-1015 Lausanne, Switzerland, 1998.

[28] DIANA Consortium, "DIANA Technical Description," technical report, 1998. http://www.telscom.ch/Diana.