Implementing the Integrated Services QoS Model with IPv6 Over ATM Networks

David Fernández[1], David Larrabeiti[2], Ana B. García[1], Arturo Azcorra[2], Luis Bellido[1], Julio Berrocal[1]

[1] Dpto. Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación
Universidad Politécnica de Madrid, Ciudad Universitaria s/n
E-28040 Madrid, Spain
{david, abgarcia, lbt, berrocal}@dit.upm.es

[2] Área de Ingeniería Telemática
Universidad Carlos III de Madrid, Avda. Universidad, 30
E-28913 Madrid, Spain
{dlarra, azcorra}@it.uc3m.es

Abstract. This paper describes the experience gained and the results achieved developing and integrating a protocol stack for IPv6 over ATM networks with Quality of Service (QoS) support based on the Integrated Services model defined by IETF. This work, which constitutes one of the first implementations of such protocol stack over Windows NT operating system, was developed as part of the authors' contribution to European funded Broadband Trial Integration (BTI) ACTS Project[1]. The paper focuses on the description of the technologies selected in BTI to provide QoS over a residential access network, and on the main characteristics of the protocol stack developed as well as the problems faced and the technical solutions applied. Finally, some conclusions about the experience are presented.

1 Introduction

In order to cover the general demand for improved quality of Internet Services, the BTI [1] project pointed its objective to develop and demonstrate a concept for improved QoS based on the integration of IPv6 and ATM. The project focused on implementing QoS in an ATM based Passive Optical Network (APON) extended to the user by means of VDSL technology, supporting unicast and multicast with well-defined QoS control in terms of the controlled load service of Integrated Services Internet approach (IntServ from now on). Fig. 1 presents the detailed architecture of the access network used in BTI.

To explore QoS features offered by the integrated IPv6/ATM networking environment, a set of distant education applications were selected in the project, including tools to access digital video servers or to co-operate synchronously by means of vir-

[1] This work has been partially supported by the EU Commission under the ACTS project 362.

tual workspaces and video/audio conferencing tools. All these applications were migrated to work over IPv6 and modified to include QoS support, in order to allow the users to demand the network a QoS level through a simple user interface.

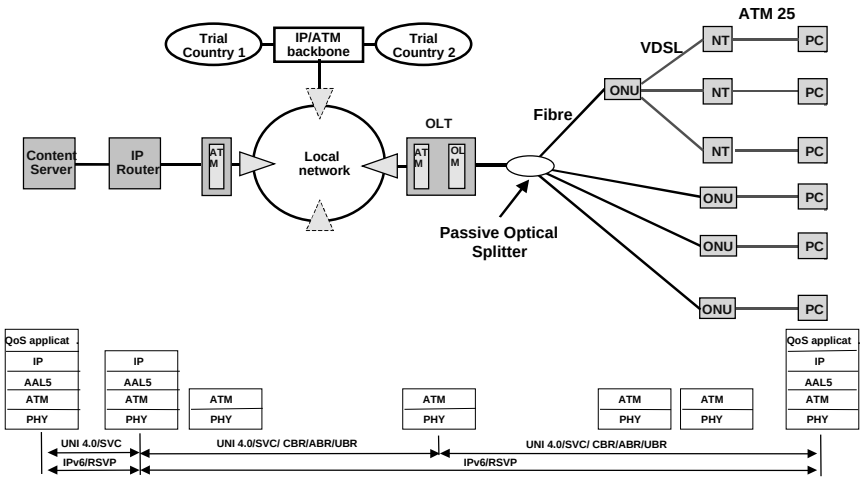

Fig. 1. BTI Network Architecture

Apart from the technical evaluation of the solutions used in BTI network, a program of structured usability testing was performed to evaluate the user perception of the QoS control and user interface. Students and teachers at universities and schools in Denmark, Poland, and Portugal participated in trial experiments over real access networks.

One of the main problems faced by the project was the unavailability of a complete protocol stack running over the target end-user's stations operating system (Windows NT) that implemented and integrated all the protocols used in BTI access networks: IPv6 over ATM with full multicast support based on multipoint ATM SVC's, and RSVP over IPv6 with traffic control functionality over ATM networks.

In principle, BTI planed to reuse some existing solution available in the market or, more likely, in the research arena; with the aim of concentrating the project effort on the adaptation of applications. However, as time progressed and no complete solution was available, the project decided to develop and integrate its own protocol stack, starting from existing protocol blocks available on the network (mainly RSVP and IPv6 implementations with source code available) and adding the missing parts (mainly IPv6 over ATM drivers and all QoS traffic control functionality). All the components, existing and new, had to be integrated to offer applications the QoS demanded through standard programming interfaces.

The rest of the paper is organized as follows. Section 2 describes the main technologies involved in BTI access network, focusing on how IPv6 works over ATM and how IntServ QoS approach is implemented over this scenario. Section 3 briefly presents the three applications selected in BTI, emphasizing on the requirements they impose on the network. Later, Section 4 describes the integrated protocol stack devel-

oped, bringing out all the difficulties found during the design, implementation and testing phases, and presenting the main technical solutions adopted. Finally, section 5 summarizes some conclusions about the work presented.

2 IntServ over ATM Networks with IPv6

This section presents how the main technologies used in BTI access network -IPv6, ATM and IntServ- were integrated to provide QoS support. First, we discuss how IPv6 works over ATM networks for *Best Effort* traffic; then we present how IntServ model has been implemented over ATM networks to provide end-to-end QoS support.

2.1 IPv6 over ATM Networks

Most ATM-based solutions implemented today in IP access networks make use of permanent virtual connections (PVC). In these scenarios, IP-over-ATM devices at customer premises communicate with routers at provider premises by means of pre-configured PVC's, either directly from hosts equipped with ATM cards or through a local router or bridge. This setup guarantees an agreed quality of service –ATM connections used are typically CBR or VBR- for the whole traffic coming out or going to the client, but does neither allow any adaptation to dynamic usage requirements nor any differentiation among data flows sent by clients. Furthermore, this approach does not take full advantage of subnetwork facilities like direct ATM connections between clients, or the ATM multipoint service, in the case of IP multicast traffic.

Carrying IP packets over ATM networks using PVCs is relatively simple; the only decision to make is the adaptation layer and encapsulation to be used -by default AAL5 and LLC/SNAP, respectively [2]- and configuring IP routing to work over these static point-to-point channels.

However, the situation is much more complex if switched virtual circuits (SVC) are used. For example, the lack of a mechanism to broadcast packets to all nodes belonging to a logical IP Subnet (LIS) on ATM prevents the usage of simple protocols, like ARP, for link-level address resolution.

Moreover, IP multicast support over ATM is rather complicated. Although multicast services exist in ATM networks -in the way of multipoint SVC's-, it is not possible to directly map IP multicast addresses to multicast subnet addresses, as it is done in LAN. Multicast model in ATM networks does not include multicast addresses at all, so an external entity to manage the association between IP multicast addresses and the set of ATM unicast addresses of hosts that belong to the group is needed. In the data plane, a multipoint ATM connection must be established to all those endpoints that have joined the group prior to sending any multicast IP packet.

In the case of multicast, the connectionless (IP) to connection-oriented (ATM) gap is greatly enlarged by the highly dynamic nature of IP multicast membership. This problem is partially simplified if UNI 4.0 ATM signaling version is used, thanks to the new leaf-initiated join capability.

In IPv4 two new protocol entities were defined when the Classical IP over ATM model (CLIP) was designed: the ATM ARP server initially defined in RFC1577 [3] and the MARS (Multicast Address Resolution Server) defined in RFC2022 [4].

The ATM ARP server is in charge of address resolution function over the ATM subnet. It is basically a client-server version of ARP that, given a destination IP address, supplies the corresponding ATM endpoint address. Likewise, MARS is a client-server protocol that resolves IP multicast addresses to the set of ATM addresses of the nodes in the LIS that have joined the target group.

The usage of these protocols is conceptually simple. Nodes in a LIS register themselves in the servers. When a node needs to forward a datagram, firstly computes the next-hop address and issues a request to the server to find out the ATM address (*addresses* in the multicast case) associated to that next-hop if it is not available in its neighbors cache. Once the ATM address is known, the source node can proceed to set up an ATM connection to the target node and transmit the datagrams over the SVC to that next-hop.

In the case of MARS, clients connect to the MARS server to register themselves and to issue multicast group joining/leaving updates. Once registered, nodes become leaves of a cluster control multipoint VC rooted at the MARS, which is used to multicast membership updates to all registered LIS nodes and usually also to respond to IP-multicast address to ATM-addresses resolution requests. Knowing of this information is required by clients to keep up the ATM multipoint SVC to the right IP group members, issuing the necessary ATM leaf add/drops.

At the time of sending multicast data, MARS allows two possible scenarios:

- *VC mesh.* Every sender to a group issues a request to the MARS to find out the ATM addresses of the group members and, once known, places a multipoint call to all of them as leaves.
- *MCS (Multicast Server).* In this case, the MARS instructs its sender clients to setup the connection to a single leaf -the MCS entity-, where a multipoint connection to all group members is rooted and shared by all senders to the group. The MCS serializes and forwards all packets received from group senders to group members via the multipoint connection.

MARS RFC [4] discusses advantages and disadvantages of each approach. Just let us outline that, although both configurations were feasible in BTI scenario, in principle the MCS solution was more suitable, due to the tree topology of the access network as well as for scalability reasons (less signaling overhead and less number of SVCs).

However, VC mesh option was chosen instead because MCS does not provide the QoS support. In particular, it is not possible to delegate on the MCS a multipoint connection to a subset of group members, as it is an essential requirement in the IntServ scenario to differentiate users who has made a reservation from the rest. Therefore, the choice for the VC mesh for the IntServ scenario implemented was a must, as our design principle was just using standardized solutions.

The way IPv6 works over ATM networks is similar to the solutions described for IPv4, but includes some important differences that make it simpler. For example, IPv4 stacks that work over LAN and ATM networks duplicate at some extend the address resolution functions, as they are both included in LAN and ATM drivers (Fig. 2).

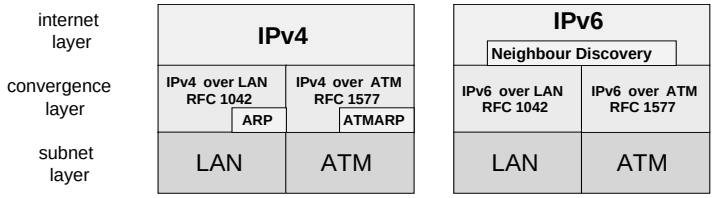

Fig. 2. IPv6 vs. IPv4 Address Resolution Architecture

IPv6 architectural model solves this problem in a simple an elegant way. In IPv6, Neighbour Discovery protocol is defined independently of the underlying subnet, and it assumes that all subnets provide multicast link-layer services to allow the multicast of Neighbour Discovery messages. In this way, address resolution function is common to all subnets (Fig. 2), and the ATMARP server is not needed. The disadvantage of this approach comes from the fact that multicast must be always supported, and therefore a MARS server must be available in the ATM subnet when using IPv6.

2.2. IntServ over ATM

The potential interest of implementing IntServ over ATM is obvious, specially if we think that flows requesting a given quality of service can take advantage of ATM short-cuts by-passing and unloading routers from packet scheduling and signaling, delegating real QoS implementation and even flow aggregation, on ATM. Nevertheless, implementing IntServ QoS approach over ATM networks it is not easy at all, due to the important differences between QoS support models of RSVP and ATM networks. To be mentioned:

- RSVP is receiver-based and ATM is sender-based. This can be partially solved using signaling at layers above ATM and, for the multicast case, by the leaf-initiated functions included in UNI 4.0.
- Resource allocation in RSVP is unidirectional. However, in ATM it is bidirectional for unicast circuits (although reservations can be asymmetric and even void in one direction), and unidirectional for multicast ones.
- RSVP allows dynamic changes of QoS reservations; however, ATM obliges to close a circuit and reestablish it again to change QoS parameters.
- RSVP allows receiver heterogeneity in multicast reservations; however, ATM imposes uniform QoS for all receivers of a multipoint connection. This implies that multicast can be performed very efficiently at the ATM layer with a guaranteed QoS only if reservations are coordinated at the application layer in such a way that a single mutipoint connection can be shared by many receivers.

In the past years a lot of effort have been invested to have a set of standards that define how to map RSVP reservation requests into ATM connections. In outline, the alternatives for each service category, as given by RFC2381 [5], are the following: the Guaranteed Service should be implemented as a CBR or rt-VBR connection, Con-

trolled Load reservations as CBR, nrt-VBR or ABR including a minimum cell rate, and Best Effort as UBR or ABR connections.

In the BTI scenario, the Controlled Load Service was realized by means of CBR (BCOB-A) connections in order to guarantee the portability to any ATM device, since this service is always present in all implementations. However, the best theoretical match, corresponds to nrt-VBR where the mapping to ATM traffic descriptors is as simple as mapping the receiver TSpec's peak rate to PCR (Peak Cell Rate), the token bucket rate to SCR (Sustained Cell Rate), and the bucket size to MBS (Maximum Burst Size). Usually, these traffic translations must be complemented with buffering edge devices to achieve a seamless coupling between the two models.

It is important to note in this context that IP nodes linked by ATM networks must establish bidirectional best-effort connections to exchange RSVP messages. These connections, together with the necessary connections for MARS signaling and the fact that currently no standard way of aggregating sessions over a single ATM connection exist, add up complexity to this solution in terms of number of VCs.

3. Application Scenarios

In order to explore QoS features offered by the integrated IPv6/ATM networking environment, a set of distant education applications was specified including Digital Video Library, Virtual Workspace and Video/Audio Conferencing tools [6].

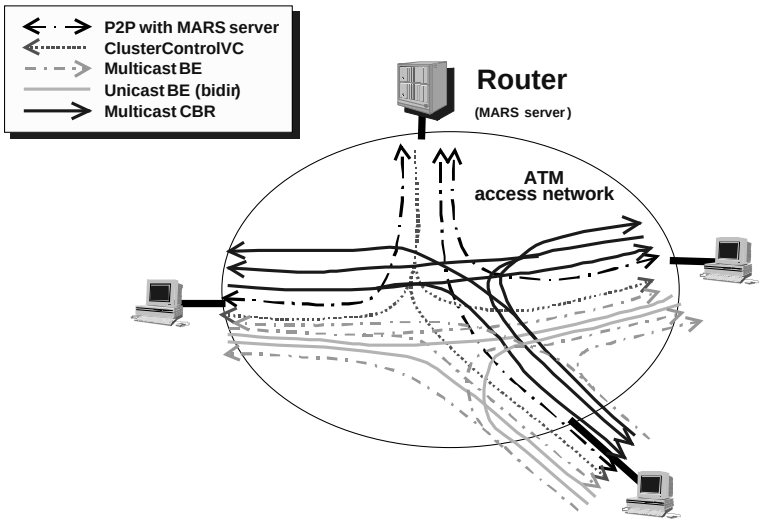

Fig. 3. A multipoint to multipoint videoconference

The Virtual Workspace is made of a set of integrated data-conferencing applications that support collaborative education. It was basically reused from the environ-

ment developed in LEVERAGE EU funded ACTS 109 project [7], and it allows users to create virtual meetings and to interact in real time by sharing documents. The Digital Video Library consists of a video-streaming engine, database, and content manager server. User interfaces for video retrieval, playback, uploading, and QoS control are embedded in a standard web browser.

The Video/Audio Conferencing tool allows the users to establish point to point and multipoint videoconferences. It was based on well-known VIC (Videoconference) and RAT (Robust Audio Tool) multicast tools used in Internet Mbone. All these applications were adapted to work over IPv6 and modified to include a simple and friendly interface to allow users to control the QoS, basically by selecting the desired quality level.

All this target applications were supported by BTI platform. However, each application had different requirements from the network, in terms of resources and functionality demanded. For example, Virtual Workspace did not required multicast, as it was implemented at application level; Digital Video Library did not required multicast services from clients, only from video server.

As expected, the most complex and resource-consuming[2] application was multipoint videoconferencing. To illustrate that, Fig. 3 shows the scenario of an audioconference with tree terminals using multicast. In this "simple" scenario each client has to maintain 10 different ATM circuits, in order to send and receive all traffic types: control traffic with the MARS server, unicast and multicast best effort traffic and multicast traffic with guaranteed QoS. In addition, if the conference uses video as well (sent to a different multicast address than the one used for audio), 6 more circuits are added to the list. As already mentioned, a centralized approach (MCS like) would have minimized the amount of SVCs, but this is not currently feasible in a fully standard way.

4. Integrated Protocol Stack

DIT-UPM and IT-UC3M main effort in BTI project was devoted to develop an integrated protocol stack named PATAM (IPv6 over ATm Adaptation Module with RSVP support, [8]). PATAM is a Winsock2 compatible protocol stack running on Windows NT, the operating system chosen by BTI application developers and trial organizations.

PATAM includes an IPv6 stack able to run over Ethernet and ATM subnetworks with full multicast support, as well as an RSVP over IPv6 implementation with traffic control support over ATM interfaces.

IPv6 support of PATAM was based on a modified version of Microsoft Research's IPv6 implementation for Windows NT [9] -that only supports IPv6 over Ethernet interfaces- with the addition of a completely new IPv6 over ATM driver developed by the authors. RSVP for IPv6 support was based on the well-known ISI's RSVP imple-

[2] From the point of view of control resources (number of circuits to establish, signaling messages, etc). In terms of bandwidth, the Digital Video Library is the most resource-consuming application clearly.

mentation for UNIX operating system [10], which has been migrated to Windows NT and adapted to offer a Winsock2 interface and to interact with MSR's IPv6 stack.

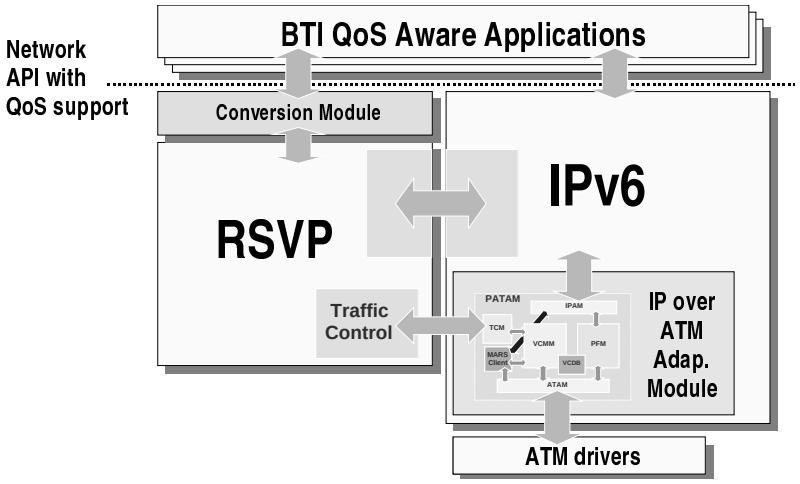

Fig. 4. Protocol Stack Architecture

The whole stack integrates under the Winsock2 based networking architecture of Windows NT. QoS aware applications use the standard Winsock2 API to access either IPv6 or RSVP services offered by the stack. Fig. 4 depicts the general architecture of PATAM stack. The next two subsections describe with more detail the two main parts of our contribution: the IPv6 over ATM driver (PATAM) and the RSVP protocol daemon.

4.1. IPv6 over ATM driver

Fig. 5 shows the detailed architecture of the IPv6 over ATM driver, including its relation with other modules of the integrated stack. PATAM is a user-mode multi-threaded driver that implements all the necessary functions to carry IPv6 packets over ATM networks using dynamic circuits (SVCs) with full multicast support. The driver is made of several components:

- *Flows Database*, which manages all the information about the Best Effort (BE) and Controled Load (CL) active IPv6 flows. Each time a new IP flow is created, either unicast or multicast; BE or CL, a new entry in the Flows Database is created, storing all the information necessary to later classify and schedule the sending of packets belonging to that flow.
- *IPv6 Access Module (IPAM)*, which manages the communications between the driver and the IPv6 stack. Each time an IPv6 packet is received through any of the ATM circuits, IPAM passes it to the IPv6 stack; and each time the IPv6 stack has a packet directed to the ATM interface, it is received by IPAM, that delivers it to the classifier and scheduler modules.

- *Packet Forwarding Module*, which is in charge of sending and receiving IPv6 packets to and from ATM network. It includes all the functions needed to classify IPv6 packets according to the different flows in the database and schedule their transmission according to QoS reservations.

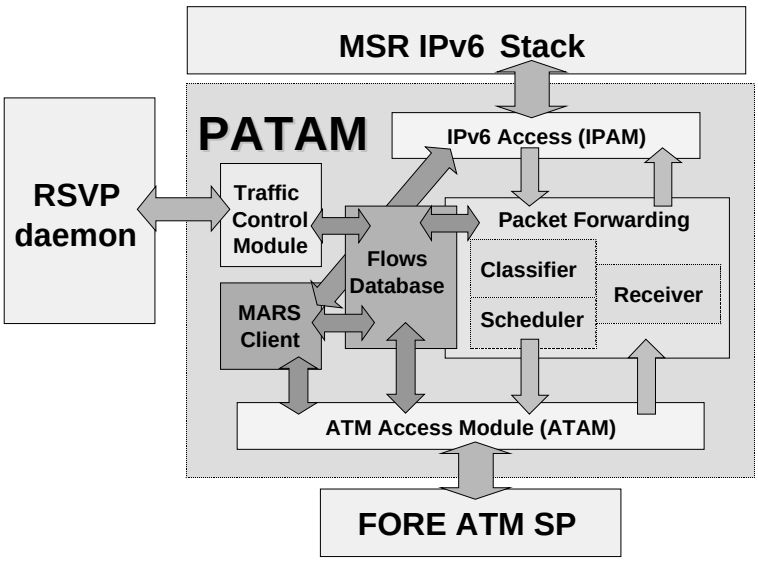

Fig. 5. IPv6 over ATM driver Architecture

- *ATM Access Module (ATAM),* which manages all the ATM circuits associated with IPv6 flows. It is in charge of creating and releasing SVCs, adding or deleting leafs to multipoint circuits and, in general, reporting other modules about any event related to ATM circuits. It accesses ATM network services using the standard Winsock2 API defined for ATM. This interface allows the creation of UBR and CBR point-to-point and multipoint circuits, however, no support for ABR was available.
- *MARS Client Module,* which implements MARS Client functionality according to [4]. All requests to send and receive to or from IPv6 multicast addresses and all the communications with the MARS server are managed by this module. This module was developed starting from a public LINUX implementation developed by NIST. This implementation was modified to support IPv6 and later migrated to Windows NT and adapted to work over PATAM architecture.
- *Traffic Control Module,* which manages the communications with the RSVP daemon for the creation and release of Controlled Load flows and their correspondent CBR ATM multipoint circuits. It is described with more detail below.

4.2. RSVP daemon

The RSVP functionality developed by the authors for BTI project includes a complete RSVP engine according to current standards [11] [12]. The main features are:

- Standard Winsock2 API.
- IPv6 support (no IPv4 support is provided), using native IPv6 encapsulation.
- Support for both Ethernet and ATM interfaces.
- Interaction with PATAM driver to offer Traffic Control (TC) implementation over ATM subnetworks, supporting FF and SE styles, and IntServ's Controlled Load reservations.
- Host (not router) implementation.

As already mentioned, the development was started from the well-known RSVP implementation of ISI [10], which works on UNIX platforms. This daemon was migrated to Windows NT and adapted to the Windows-specific asynchronous event notification, and later was completed in order to provide actual Traffic control support for ATM subnetworks. Figure 6 shows the specific architecture of the RSVP module.

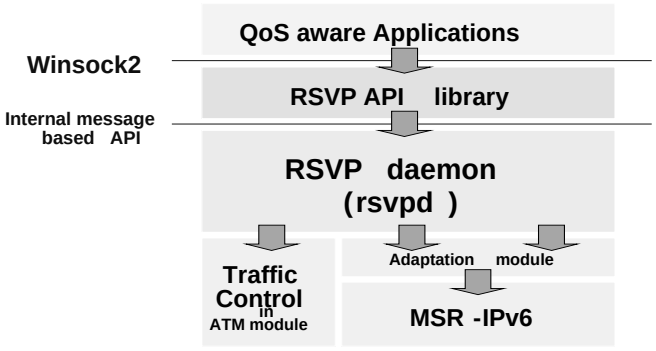

Fig. 6. RSVP Architecture in PATAM

The main tasks carried out in the developing of RSVP module are briefly discussed:

- *Adaptation to Winsock2 architecture.* To offer the standard Winsock2 RSVP API to applications, a standard WS2 Service Provider (SP) Library was developed. This library performs the translation between WS2 interface offered to applications and an internal interface with the core RSVP processing, which has been kept the same as it is in ISI's implementation.

- *Interfaces with the IPv6 stack: routing and I/O.* RSVP module has to access IP functionality at a lower level than a standard application. Regarding pure input/output (I/O), at least raw IPv6 access must be provided. It is also necessary for RSVP to know, for instance, what interfaces the system has, or through what interfaces a PATH should be forwarded according to IP routing information. Although advanced APIs [13] are being defined for this purpose, they were not available in

the IPv6 protocol stack used, so operating system specific interfaces were used instead. This fact implied some important modifications to the MSR's IPv6 stack.

- *Traffic Control for ATM subnetworks.* ISI's implementation introduces an intermediate layer between core RSVP processing and the actual Traffic Control module: the Link Layer Dependent Adaptation Layer (or LLDAL). It also provides a LLDAL and an (almost) empty TC implementation suitable for Ethernet interfaces. In BTI new ATM-specific LLDAL (pertaining to the daemon) and ATM TC module (actually implemented within PATAM) have been developed. The interface offered by LLDAL has been maintained, making it possible for the daemon to work both over Ethernet and ATM interfaces applying the correct Traffic Control to each one.

  The ATM TC offers to the ATM LLDAL a simplified and adapted-to-ATM version of the TC interface specified in RFC 2205 [11]. This simplified interface basically allows the daemon to open and close CBR circuits (or leafs of circuits) when reservations must be placed or torn down. Fig. 7 summarizes the modules involved in the TC for the two types of interfaces supported. BTI work has been focused on the remarked modules.

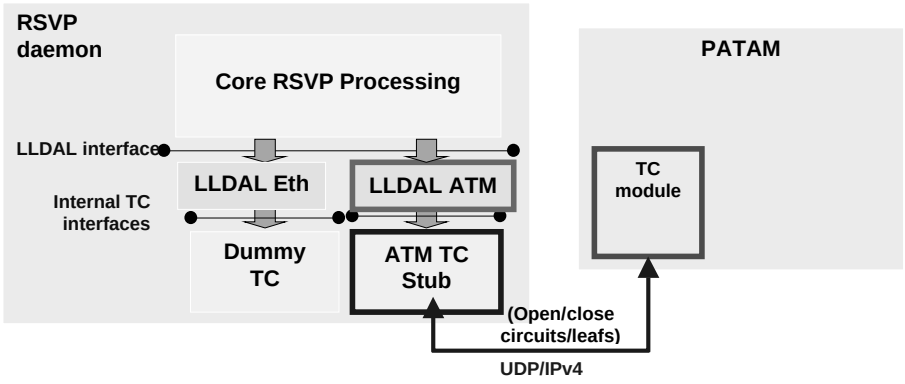

Fig. 7.: Traffic Control Architecture in PATAM

5. Conclusions

Although the timeframe for the development and integration of the protocol stack was very tight (in fact, the whole development was made in less than 8 months), the effort dedicated to the task concluded with a successfully running solution. That solution fulfilled the requirements imposed by BTI network scenario and applications running on Windows NT (videoserver, videoconference and data applications), and made possible the experimentation with the whole BTI system during usability testing phase.

With respect to the practical problems found during protocol development, in summary, most of them were due to the use of state-of-the-art implementations of the technologies being integrated in BTI project. Even some technologies we thought at the beginning that should be stable enough were found to be unstable.

Many conclusions can be drawn from the experience acquired by the development of the protocol stack and the usage trials over the broadband access scenario. Let us outline the most important ones. On the positive side:

- It has been demonstrated the feasibility of implementing dynamically provided end-to-end QoS using IPv6 over ATM, with the added complexity of an underlying VDSL access network. This solution provides a tight way of controlling network resources in an access network.
- IPv6 implementations for Windows NT (from Microsoft Research), Ericsson-Telebit and Solaris proved to be stable and interworked properly. Basic IPv6 functionality could be used in production networks. However, the lack of advanced functions made the integration of the ATM driver and the RSVP daemon difficult and costly. Advance functionalities need to be work out and stabilized before it can be used in production environments.
- The way Neighbor Discovery is organized in IPv6 compared to how it is done in IPv4 has simplified driver development. Instead of having to write an ATM ARP client for address resolution, as it should have been the case for IPv4, we only have to add multicast support to the driver in order to have ND functions working. Some small modifications were needed to the IPv6 protocol stack, for example, to cope with bigger subnet addresses (20 octets for ATM addresses compared to 6 octets MAC addresses). But basically, as all ND functions were located inside the protocol stack, the driver development was simplified.

On the negative side, let us address the following problems and proposed solutions:

- Too many ATM circuits must be setup in order to support IPv6 multicast over ATM using multipoint switched QoS VCs in a full standard way. Solutions to this problem could be: multiplexing different flows over the same ATM circuit, improving multicast signaling over ATM by using only MARS signaling and not the ICMPv6 Multicast Listener Discovery procedure (for example, to avoid the multipoint circuits used only for sending MLD report packets in receiver-only clients), and finally the possibility to order multipoint connections with a given QoS at the MCS over a given subset of group subscribers.
- As stated above, the way IPv6 is conceived streamlines the development of new network drivers for new media if multicast support is one of the targets. Otherwise, for a non-broadcast medium, the implementation of neighbor discovery is costly and treating ATM as a static point to point single LIS link can be worth the extra effort required to simulate multicast at the link layer.
- Lack of integrated QoS APIs. The protocol stack provides two different interfaces: one is used to access IPv6 stack to send and receive data, and the other to access RSVP services to make reservations. Although the interfaces are not com-

plex, the applications are in charge of coordinating the activity between them, and that has been demonstrated problematic for application developers. The use of an integrated network API with QoS support, like the one defined for Winsock2 in [14], will simplify greatly the development or adaptation of QoS aware applications.

Acknowledgements

This work was been partly supported by the EU Commission under the ACTS project 362 BTI. DIT-UPM and IT-UC3M would like to thank all the partners involved in BTI project and also the ACTS Project Manager for their positive collaboration and contribution to the project success.

References

1. Andersen, N., Azcorra, A., Bertelsen, E., Carapinha, J., Dittmann, L., Fernandez, D., Kjaergaard, J., McKay, I., Maliszewski, J., Papir, Z.: Applying QoS Control through Integration of IP and ATM. IEEE Communications Magazine, July 2000.
2. Ginsburg, D.: ATM. Solutions for Enterprise Internetworking. Second Edition. Addison Wesley, 1999.
3. Laubach, M.: Classical IP and ARP over ATM. Request for Comments 1577. IETF Proposed Standard, January 1994.
4. Armitage, G.: Support for Multicast over UNI 3.0/3.1 based ATM Networks. Request for Comments 2022. IETF Proposed Standard, November 1996.
5. Garrett, M., Borden, M.: Interoperation of Controlled-Load Service and Guaranteed Service with ATM. Request for Comments 2381. IETF Proposed Standard, August 1998.
6. Fernández, D., García, A. B., Larrabeiti, D., Azcorra, A., Pacyna, P., Papir, Z.: Bouquet of Multimedia Services for Distant Work & Education in IP/ATM Environment. Pending publication, June 2000.
7. Fernández, D., Bellido, L., Pastor, E.: Session Management and Collaboration in LEVERAGE. First LEVERAGE Conference on Broadband Communications in Education and Training, Cambridge, January 1998. Available at: http://www.dit.upm.es/leverage.
8. PATAM Protocol Stack v0.9, March 2000. Available at: http://www.dit.upm.es/bti.
9. IPv6 implementation for Windows NT. Microsoft Research. Available at: http://research.microsoft.com/msripv6.
10. RSVP daemon. Information Science Institute. University of Southern California. Available at: http://www.isi.edu/rsvp.
11. Braden, R., (ed.): Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. Request for Comments 2205. IETF Proposed Standard, September 1997.
12. Wroclawski, J.: The Use of RSVP with IETF Integrated Services. Request for Comments 2210. IETF Proposed Standard, September 1997.
13. Stevens, W., Thomas, M.: Advanced Sockets API for IPv6. Request for Comments 2292. February 1998.
14. Bernet, Y., Stewart, J., Yavatkar, R., Andersen, D., Tai, C., Quinn, B., Lee, K.: Winsock Generic QoS Mapping. Windows Networking Group.