An AAL2 based Framework for Efficient Transport of RTP Voice Streams

by

K. Dhananjaya Rao

B.E. University of Bombay, India, 1997

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

_____

Professor in Charge

_____

_____

Committee Members

_____

Date of Acceptance

Abstract

Asynchronous Transfer Mode has been used for a number of years to implement high-speed networks providing multi-gigabit services for multimedia applications. Of late, it is being widely deployed for high-bandwidth network access using new technologies such as DSL. The principal advantages for ATM are its reliability, fine-grained QoS mechanisms and the ability to efficiently transport integrated services, including voice.

With the rapid development of VoIP as a ubiquitous technology for voice over the Internet, it becomes necessary to efficiently transport VoIP packets, which are encapsulated in Real-Time Transport Protocol (RTP) headers, over ATM access links. The ATM Adaptation Layer 2 is a standard for transporting low bit rate, short and variable length voice packets. In this work, a framework is implemented for efficient transport of RTP voice streams using AAL2. The benefits of this scheme are discussed and the applicability of this framework is studied.

Acknowledgements

I would like to thank Dr. Joseph Evans, my advisor and committee chairman for his guidance and advice throughout my research work. It has been a pleasure working under him. I'm especially grateful to him for the freedom I have been given to explore new areas and develop my interests. I would also like to thank Dr. David Petr and Dr. Susan Gauch for being on my committee.

This project owes a lot to Vishal Moondhra who started it all and to Aarti Iyengar who worked with me on the initial AAL2 implementation. I am also deeply thankful to Sachin Sheth for his invaluable advice during my early days here and for his wonderful common sense.

I cannot forget to mention Brett, Roel and Mike for their immense help and for putting up with me for these two years. Special thanks to all those anonymous angels who kept the coffee pot full whenever I badly needed it. I would also like to thank my friends here at ITTC and KU, Ananth, Aarti, Deepak, Pramodh, Shirish, Harish, Sarav, Anu, Gowri, Giri, Sriram, Vijay, Phongsak, Ram and many others for making my stay here an enjoyable and learning experience.

All my endeavors so far have been due to the grace of God and the support of my parents and sister. My gratitude to them cannot be expressed in mere words.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The ever-increasing demand for faster access to information, data and communications has led to rising bandwidth requirements. Various broadband access technologies are being increasingly deployed to reduce last-mile costs while providing multiservice capabilities.

Multiservice networking is emerging as a strategically important issue for enterprise and public service provider infrastructures alike. The combination of all types of communications, all types of data, voice, and video over a single packet-cell-based infrastructure provides benefits of reduced operational costs, higher performance, greater flexibility, integration and control, and faster new application and service deployment.

Packetized voice is on the rise in carrier networks, limited corporate trials and next-generation Wide Area Network (WAN) access. Wide-area bandwidth remains a premium, so efficient utilization of available bandwidth is a prime area of concern. No single voice-over-packet technology fits all needs today. The major unification technologies - voice over IP (VoIP), voice over ATM (VoATM) and voice over Frame Relay (VoFR) - each have relative strengths and weaknesses, and there is no perfect technology fit for every problem.

VoIP is the latest voice-over-packet technology, which is rapidly heading towards becoming the most widespread in future. Its primary appeal is the reach of IP, which spans any underlying Layer 2 technology. Hence, voice can be converted to packets at any point in the network - within the carrier core, at the WAN boundary, in the campus or at the endpoint. VoIP especially targets the end-to-end approach, using network-attached phones, PCs and PC-like devices, and wireless devices. But even though IP has the

widest span, it traditionally provides a best-effort service and lacks the network reliability and QoS required for applications such as voice and video.

ATM has existed in carrier cores and campus backbones for many years and is being widely deployed for high-bandwidth network access using DSL in small offices and Synchronous Optical Network (SONET) in larger facilities. The principal advantages for ATM are in reliability, fine-grained QoS mechanisms and advanced network management. It has been demonstrated that ATM is an effective medium to transport integrated services, including voice.

The most common approach to encapsulating voice over ATM is to use Circuit Emulation Services (CES) based on the AAL1 encapsulation method. Unfortunately, CES doesn't offer the statistical gains required for maximizing network utilization, and it has a high overhead.

Many enterprise implementations use AAL5 LLC/SNAP encapsulation for voice services. Although this approach is less complex, it is also less efficient than another approach using AAL2. The ATM Forum has defined AAL2 for bandwidth efficient transmission of delay sensitive, low bit-rate voice services.

AAL2 is already in widespread use for trunking purposes as a viable alternative to AAL1 CE. With service providers increasingly looking towards voice as an enhanced revenue generating service over access technologies such as DSL (VoDSL), AAL2 can be used for this purpose. Access connections using this scheme can transport voice over the same facilities as data, minimizing the use of precious bandwidth.

At the same time, VoIP applications are increasingly being used to provide a cheaper and more flexible means of communication, in the enterprise LAN and also over the Internet. Voice packets over an IP network are usually encapsulated in Real-Time Transport Protocol (RTP) headers to provide end-to-end transport functionality required by voice, between the sources.

Hence it becomes important to determine if AAL2 can be deployed to efficiently transport this RTP encapsulated voice traffic over ATM. This thesis focuses on developing a framework for transport of RTP voice streams over an ATM network using AAL2. The salient points of interest are the efficiency in bandwidth utilization that can be obtained with this framework; the tradeoffs involved, such as any introduced delay; and the choices taken in this approach.

AAL2 provides inherent multiplexing capabilities within an ATM SVC, which make it simpler to use multiplexing mechanisms between two gateways or access routers and also increase the bandwidth efficiency as compared to other RTP over ATM approaches. Header compression is used here to further improve efficiency.

1.2 Organization of thesis

This document is organized in the following manner: Chapter 2 discusses some of the related work to this thesis. Chapter 3 gives an overlook about the different protocols and technologies used in this thesis. Chapter 4 discusses the design aspects of the proposed framework and explains some of the implementation details. Chapter 5 describes the set of tests that were conducted to determine the validity and performance of the implemented framework and presents some results. Chapter 6 contains the conclusions and future work.

Chapter 2

Related Work

This chapter discusses some of the ongoing work in the areas of efficient transport of real-time voice traffic in IP and ATM networks while providing the necessary quality of service. It includes efforts being carried out in the ITU, ATM Forum and IETF standards bodies in the context of RTP. Since the Real-Time Transport Protocol was published as an RFC [1], there has been growing interest in using RTP as one step to achieve interoperability among different implementations of network audio/video applications.

2.1 ITU Efforts

The ITU has developed a number of specifications for transport of multimedia including voice. The most important of these is the H.323 standards suite, which is the most widely deployed specification for voice over IP. The H.323 standard [3] enables real-time multimedia communications and conferencing over packet-based networks. It is a comprehensive standard that covers the selection of audio and video codecs, shared applications, call control, and system control, allowing vendors to develop interoperable products for LAN-based audio and video communications.

Figure 2.1 shows the different layers and protocols that are part of the H.323 specification. The H.323 protocol stack uses the H.245 standard for passing control information and H.225 for passing RAS (registration, admission and signaling) information. The media channel is carried using RTP. RTCP or Real-Time Control Protocol is used as the media control channel. Annex C of the ITU-T recommendation explains the usage of H.323 media on ATM.

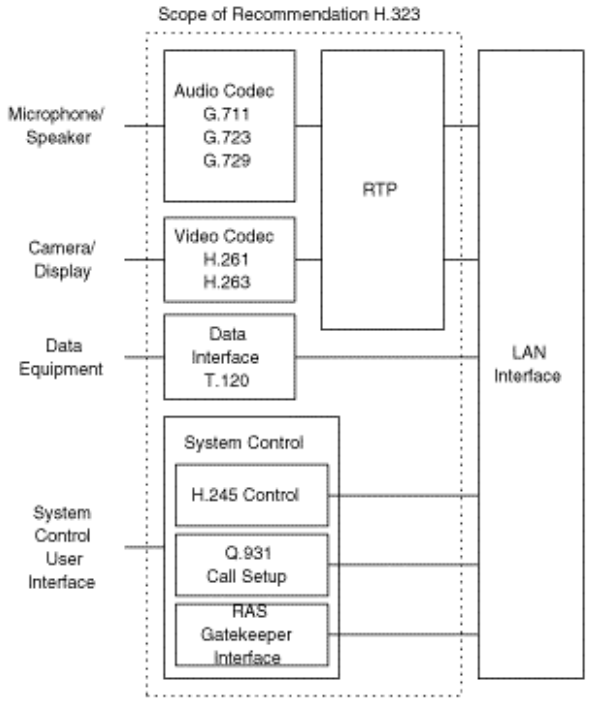


Figure 2.1 H.323 Protocol Suite

2.2 ATM Forum Efforts

There is work going on in the ATM Forum called RMOA (real-time multimedia over ATM) for H.323 transport over ATM [4]. The focus of RMOA work is on using RTP over ATM in an intelligent way by utilizing the QoS features of ATM. It describes the access to an ATM network using H.323. The H.323 terminal can be on a variety of network technologies, including non-native ATM IP-based (Ethernet, etc.), and native ATM. This differs from the ITU-T specification, which only targets native ATM terminals.

Carrying H.323 media streams over ATM will ensure that the media streams take advantage of the inherent quality of service of ATM. However, to ensure a suitable end-to-end quality of service, it is necessary that appropriate QoS mechanisms are applied outside the ATM network.

The specification proposes termination of both signaling and media streams on the gateways to the ATM network. With respect to control traffic between the endpoints, each of the gateways to the ATM backbone network serves as an H.323-to-H.323 gateway. With respect to media streams between the endpoints, each of the gateways terminates the media stream for the near endpoint and acts as a compressor/decompressor for the RTP media stream.

SVCs are set up for the transport of RTP media streams based on H.245 or H.225.0 control messages exchanged by the endpoints and the gateway. In the ATM network, the corresponding control messages are sent between the gateways using an IP over ATM method. This specification is discussed later in some more detail, as this thesis derives a number of issues from it.

2.3 IETF Efforts

There have been a lot of developments in designing new protocols and adding new functionality to the existing protocol stacks for efficient voice transport. A number of IETF working groups are focussing on the development of standards for VoIP, such as *avt* (Audio/Video Transport), *iptel* (IP Telephony), *mmusic* (Multiparty Multimedia Session Control) and *pint* (PSTN and Internet Internetworking).

The Audio/Video Transport Working Group was formed to specify a protocol for real-time transmission of audio and video over UDP and IP multicast. This is the Real-time Transport Protocol, RTP, together with its associated profile for audio/video conferences and payload format documents. The group is currently focussing on revision of the specification and profile, development of new payload formats and guidelines for developers.

A number of drafts have been submitted which are aimed at increasing the efficiency of RTP transport over low speed access links and reduce the protocol overhead. These

efforts include RTP multiplexing, tunneling and header compression techniques. This thesis also draws some of its features from these efforts.

The IETF has also been working diligently to develop specifications to enable real-time applications such as voice to work over IP networks, notably DiffServ and MPLS. The current trend indicates that the model of future IP-based networks will use IP to access the network, where DiffServ mechanisms will be in place to prioritize traffic according to application requirements. This will then be translated into MPLS mechanisms at the network ingress with the use of a label to be attached to the packet for transport across the backbone network. There has been recent effort to propose voice transport over IP using the MPLS protocol.

2.4 VoDSL

Voice over digital subscriber line (VoDSL) presents a tremendous opportunity for service providers in the converging data and voice communications market. Since 1996, high-speed Internet access has been the primary market for new service providers because of the ever-growing demand from both business and residential customers. In recent years, DSL has emerged as the most affordable method for carriers to serve both types of subscribers.

Within the transport, a number of technologies are supported to transfer both voice and data. While data transport has been optimized, the simultaneous transfer of toll quality voice needs to be addressed.

Initial VoIP initiatives were not successful because the quality of service could not be maintained throughout the network. Also, H.323 maintains too many overheads if used for supporting only voice.

Recently, after much discussion, the ADSL Forum determined in favor of using ATM technologies as the basis for the first phase standard approach to VoDSL service. An

estimated 90 percent of the installed DSLAM (Digital Subscriber Line Access Multiplexer) equipment uses the ATM transport method back to the switch. Though older techniques use AAL1, the ADSL Forum is anticipated to adopt the more efficient AAL2 for voice services.

Chapter 3

Background

This chapter gives a brief description about the various protocols and technologies that are part of this thesis work. We start with a basic outline of ATM, its different Adaptation Layers, and the various signaling entities present in the system. The Adaptation Layer 2, which is of relevance to this thesis and the Real-time Transport Protocol (RTP) are then discussed in some detail. Finally, some salient features of the Robust Audio Tool and different speech coding techniques are explained.

3.1 Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) has rapidly emerged as a protocol of choice for the demands made by multimedia networks [5]. ATM networks have many distinctive features that help maintain its edge over other network protocols, especially in the area of high speed networking carrying different kinds of data. ATM transfers data between network elements using fixed sized 'packets', or cells of 53 bytes.

3.1.1 ATM Adaptation Layers

Above the ATM Layer lies the Adaptation layer, which provides for the transformation of the higher-layer service - voice, video, data - into a form suitable for transmission over the ATM infrastructure. The AAL preserves timing relationships for traffic requiring it (voice, for example). The ATM Forum defines the following AAL types:

1. *AAL1* is used to support Constant Bit Rate (CBR) connections across the ATM network. Incoming data is placed in an ATM cell along with a 3-bit sequence number and a 4 bit CRC. The remaining bit in the first byte is used over a series of cells to indicate, among other things, timing recovery information and whether or not the AAL1 connection is structured.

2. *AAL2* is used to multiplex many low bandwidth channels over a single VC. Each channel has a 3 byte channel overhead, including a Channel Id Number, and a length indicator. Up to 255 channels can be multiplexed over a VC, and individually setup and torn down. AAL2 needs a separate AAL2 Negotiation Protocol (ANP), as is discussed in section 3.4.8.

3. *AAL3/4* is an obsolete adaptation standard used to deliver connectionless and connection- oriented data over the ATM network. This AAL has a substantial overhead in terms of sequence numbers and multiplexing indicators, and is rarely used.

4. *AAL5* is an outgrowth of the data communication industry. It is optimized for data transport. The PDU is broken up into ATM cell segments, and the last ATM cell carries an indication in the PTI. The last cell is padded out to 48 bytes. It contains a CRC over the entire PDU, and the length of the PDU.

3.1.2 ATM Signaling

When two nodes in an ATM network want to communicate, they first need to establish a virtual connection [6]. These connections can be either provisioned (in which case they are called Permanent Virtual Circuits or PVCs) or established on demand (in which case they are called Switched Virtual Circuits or SVCs). PVCs are analogous to leased lines in a phone network, while SVCs are analogous to making a call over a phone network. SVCs require signaling support on the originating node, the switches that lie along the path, and the on terminating node. ATM networks have dedicated Signaling Channels, which implement connection setup and tear down between hosts.

The signaling is of two kinds – User-Network Interface, or UNI, and Network-Network Interface, or NNI. When a user on a host wants to setup a connection, the UNI entity on the host sends a setup message to the network. The network, which is collection of

switches, will use NNI to build a route between the destination, and the final leg between network and the destination host will again use UNI to setup the connection.

The ATM Forum, a group of industrial representatives and academicians, are responsible for establishing and standardizing the various aspects of the ATM protocol. The Forum has currently published Version 4.0 of UNI signaling, and has recently published the PNNI specification, which is Public NNI. Before PNNI was standardized, ATM networks implemented the Interim Inter-switch Signaling Protocol, or IISP, as an interim solution. The work on this project has used and extended UNI 3.1 on the hosts.

3.2 Switching in ATM

An ATM switch contains a set of input ports and output ports, through which it is interconnected to users, other switches, and other network elements. It might also have other interfaces to exchange control and management information with special purpose networks. Theoretically, the switch is only assumed to perform cell relay and support of control and management functions. However, in practice, it performs some inter-networking functions to support services such as SMDS or frame relay. It is useful to examine the switching functions in the context of the three planes of the B-ISDN model [7].

3.2.1 User Plane

The main function of an ATM switch is to relay user data cells from input ports to the appropriate output ports. The switch processes only the cell headers and the payload is carried transparently. As soon as the cell comes in through the input port, the Virtual Path Identifier/Virtual Channel Identifier (VPI/VCI) information is derived and used to route the cells to the appropriate output ports. This function can be divided into three functional blocks: the input module at the input port, the cell switch fabric (sometimes referred to as switch matrix) that performs the actual routing, and the output modules at the output ports.

3.2.2 Control Plane

This plane represents functions related to the establishment and control of the VP/VC connections. Unlike the user data cells, information in the control cells payload is not transparent to the network. The switch identifies signaling cells, and even generates some itself. The Connection Admission Control (CAC) carries out the major signaling functions required. Signaling information may or may not pass through the cell switch fabric, or maybe exchanged through a signaling network such as SS7.

3.2.3 Management Plane

The management plane is concerned with monitoring and controlling the network to ensure its correct and efficient operation. These operations can be subdivided as fault management functions, performance management functions, configuration management functions, security management functions, accounting management and traffic management. These functions can be represented as being performed by the functional block Switch Management. The Switch Management is responsible for supporting the ATM layer Operations and Maintenance (OAM) procedures. OAM cells may be recognized and processed by the ATM switch. The switch must identify and process OAM cells, maybe resulting in generating OAM cells. As with signaling cells, OAM cells may/may not pass through cell switch fabric. Switch Management also supports the interim local management interface (ILMI) of the UNI. The Switch Management contains, for each UNI, a UNI management entity (UME), which may use SNMP.

3.3 Signaling on the AAL2 Gateways

The gateway systems exist on a PC platform, running Linux. Networking support on Linux exists in the form of the BSD socket interface. The Linux ATM driver is patched in to provide ATM Network and Link Layer support. A standard ATM Network Interface Card (NIC) is installed to obtain a fiber channel to conventional ATM switches.

Signaling on the system is performed by a set of daemons that are part of the ATM driver for Linux. There are essentially three important signaling daemons.

3.3.1 ATMSIGD

atmsigd is the UNI signaling daemon for the Linux ATM [8]. It implements the UNI 3.1 signaling stack for a single ATM interface on the gateway. The signaling daemon implements the UNI signaling complexity as part of user space, while a simple protocol to support ATM signaling resides in the kernel. The user process communicates with the kernel using a simple Internal Signaling Protocol, which relies on the well-ordered nature of the system to manage the signaling. The ISP uses synchronous communication based on BSD sockets.

3.3.2 ILMID

ilmid is the Interim Local Management Interface daemon that is used to manage the system status and configuration, perform address registration and update at the switch. It uses existing SNMP standard, and defines a new ATM UNI Management Information Base (MIB) to perform VC status and management, operational measurement as required, diagnostics, etc [6].

3.3.3 ATMARPD and ATMARP

atmarp is used to resolve IP addresses over an ATM subnet. The ATMARP requests and replies are sent using AAL5. The ATMARP structure consists of an ARP server whose ATM address is well known within the subnet. Hosts wanting to resolve IP addresses to their respective ATM addresses send queries to the ARP server, which looks up its data base and responds with the required address. The ARP server updates it's data base when it receives ARP requests by sending inverse ARP or InARP REQUEST to the originating host for each logical IP subnet the server is configured to serve. In the event that the server is unable to find the corresponding entry in its table, it returns an ARP NAK to the

host. The ARP client is responsible for contacting the ARP Server with its own IP address to register itself. This usually happens at boot-up time. It is also responsible for initiating and maintaining a VC to the ARP server, responding to ARP and InARP requests, and generating and transmitting ARP REQUEST when required by applications wishing to make connections.

3.4 ATM Adaptation Layer 2

AAL2 is an adaptation layer that is used to multiplex more than one low bit rate user information stream on a single ATM virtual connection [9]. This AAL provides for bandwidth efficient transmission of low-rate, short, and variable length packets in delay sensitive applications. In situations where multiple low bit rate data streams need to be connected on end systems, a lot of precious bandwidth is wasted in setting up individual VCs for each of the connections. Moreover, most network carriers charge based on the number of open Virtual Connections, hence it is efficient both in terms of bandwidth and cost to multiplex as many of these as possible on a single connection. A preliminary standard published by the International Telecommunication Union (ITU-T) can be found in [9]. Drafts related to AAL2 SSCS and AAL2 signaling have recently been published.

3.4.1 General Framework of AAL2

The AAL type 2 is subdivided into the Common Part Sublayer (CPS) and Service Specific Convergence Sublayer (SSCS) as shown in Figure 3.1. Different SSCS protocols may be defined to support specific AAL2 user services or groups of services. The SSCS may also be null, providing merely for the required mapping between the CPS and higher layers. AAL2 provides the capabilities to transfer AAL-SDUs from one AAL-SAP to another through the ATM network. Multiple AAL2 connections may utilize a single underlying ATM connection. The multiplexing and de-multiplexing of connections occurs at the CPS.

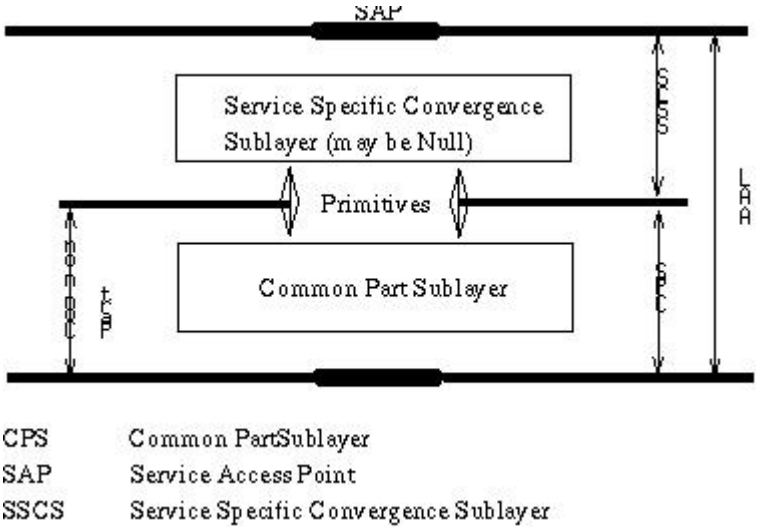


Figure 3.1 AAL2 Structure

3.4.2 CPS to ATM data interface

The CPS hands a 48-byte ATM payload to the ATM layer below it, a 1-bit ATM User to ATM User (AUU) indication, and a loss priority (called the Submitted Loss Priority). SLP is used by the ATM layer to set it's own CLP bit. CPS also receives from the ATM layer a 48 byte SDU, and a loss priority bit (called the Received Loss Priority). The RLP may differ from SLP in case the network changed CLP along the way.

3.4.3 CPS to SSCS data interface

The CPS hands CPS-Interface data packets to the SSCS (1 to 64 bytes). The format and actual length of the data are determined at setup time. The CPS also hands a 5 bit User to User Indication to the SSCS. This is data used optionally by the SSCS entity to decide the destination of the PDU. The CPS also receives the same two units from the SSCS entity.

3.4.4 The Common Part Sublayer

AAL2 CPS offers the following peer to peer operation:

- Data transfer of CPS-SDUs of up to 45[1] (default) or 64 bytes.
- Multiplexing and de-multiplexing of multiple AAL2 channels.
- Maintains the CPS-SDU sequence integrity on each AAL2 channel
- Unassured operation, i.e. lost CPS-SDUs are not retransmitted
- Bi-directional virtual channel connection, using the same VC number in either direction.
- The VC can be permanent or switched.

The CPS interacts with both the management layer and the control layer. The control layer establishes the VC as required.

3.4.5 Format and Encoding of CPS Packet

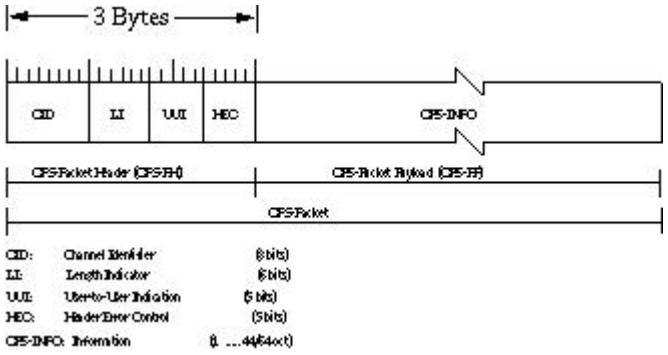


Figure 3.2 AAL2 CPS Packet Format

A CPS Packet consists of a 3 byte Packet Header (CPS-PH), followed by up to 64 bytes of Packet Payload (CPS-PP). CPS Packets are the data exchange mechanism between CPS and SSCS. Figure 3.2 shows the field lengths and format.

[1] There are some references that specify the maximum AAL2 CPS-SDU length as 44 bytes which is a logical value, since additional 3 byte CPS header and 1 byte STF fill an ATM PDU perfectly. The value of 45 has been taken from the original ITU-T I.363.2 specification [9].

- *Channel Identifier (CID)* identifies the AAL2 channel user. The AAL2 channel is a bi-directional-medium, and both directions use the same value of CID.

CID value	Use
0	Not used
1 2…7	Reserved for layer management peer-to-peer operations Reserved
8…255	Identification of SSCS entity

Table 3.1 AAL2 CID Values

- *Length Indicator (LI)* is a binary encoded value that corresponds to the length of the payload of the CPS-Packet. The default maximum length is 45 bytes. It can be set to a maximum of 64 bytes. The maximum channel length needs to be negotiated at setup time. LI cannot exceed the maximum negotiated value. Each channel can individually negotiate its maximum value. Maximum lengths between 45 and 64 are not allowed.

- *User-to-User Indication (UUI)* serves two specific purposes:
  - To convey specific information to SSCS entities transparently through the CPS.
  - To distinguish between the SSCS entities and Layer Management users of the CPS

  The 5 bit UUI field is handed without change by CPS to the SSCS entity. Its usage by the SSCS entity is optional.

- *Header Error Control (HEC)* is the reminder (modulo 2) of the division, by generator polynomial $X^5 + X^2 + 1$, of the product of X^5 and the contents of the first 19 bits of the CPS-PH. The receiver uses the HEC field to detect errors in the CPS-PH.

3.4.6 Format and Encoding of CPS-PDU

The CPS-PDU consists of a one-byte start field (STF), and 47-byte payload. The 48-byte CPS-PDU is the ATM cell SDU (Figure 3.4). A CPS-PDU may carry 0, one or more full or partial CPS-Packets. The packets may overlap over more than one PDUs. Any unused space in the PDU is padded with 0s. The CPS-Packet may be partitioned anywhere along it's length (Figure 3.3). The start field values are:

- Offset Field (OSF): This field carries the binary value of the offset, measured in number of bytes, of the first start of a CPS-Packet or, in the absence of a start of a CPS-Packet, to the beginning of the PAD field. A value of 47 indicates that the packet has overlapped up to the next cell.

- Sequence Number (SN): This 1-bit field is a modulo 2 sequence number of the stream of CPS-PDUs Parity (P) To detect errors in the STF, a 1 bit odd parity is set as the last bit of the STF.

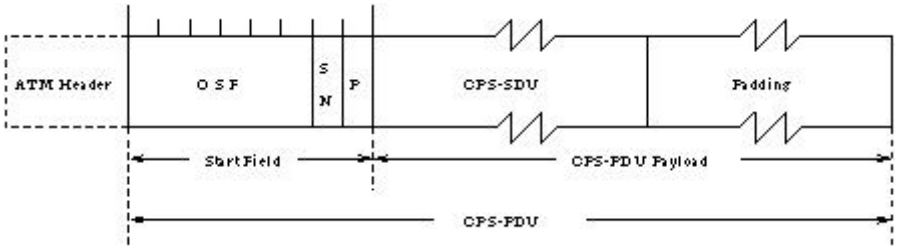


Figure 3.3 AAL2 CPS PDU Format

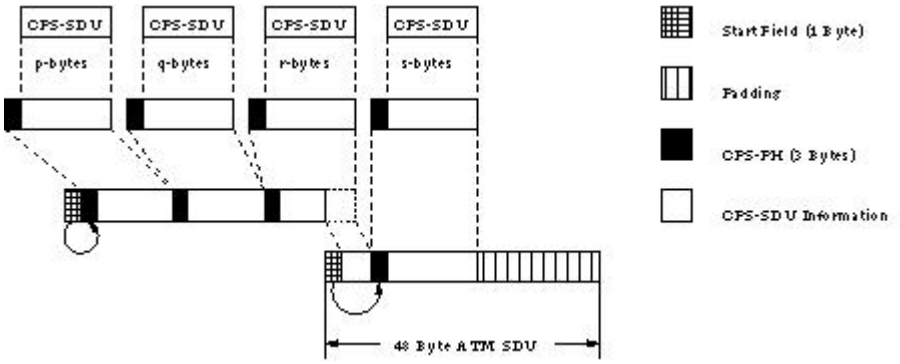


Figure 3.4 Translating CPS SDUs to ATM SDUs

3.4.7 AAL2 CPS Procedure

The CPS consists of distinct state machines for transmission and reception that function independent of each other. The transmission state machine multiplexes the various channels into as few ATM SDUs as possible, while still maintaining the time requirements of the CBR traffic, while the reception state machine demultiplex channels that can be spread over multiple ATM SDUs. The detailed explanation of the two state machines can be found in [10].

3.4.8 AAL2 Negotiation Protocol (ANP)

The ATM signaling protocol as defined in UNI 3.1 does not cater for setting up and tearing down individual channels across a switched network. Further, each channel is an entity by itself, requiring a complete negotiation process like the setup of SVCs over a switched network. Hence, a separate negotiation protocol is required that can manage channels on individual VCs. The AAL2 Negotiation Procedures was defined as an annex to the ITU-T I.363.2 recommendation for dynamic allocation of AAL2 channels.

An implementation of the ANP is described in [10]. The ANP was implemented in the form of a daemon that runs on the end systems. The setting up and deleting of individual

channels is transparent to the switched virtual network. Since no signaling support is available on any of the component systems, channel setup must work within the framework of the existing protocol. Some rudimentary form of bandwidth negotiation is supported so that, channels can be guaranteed the bit rate they started out with. The ANP restricts additional channels until it can allocate the requested bandwidth. The ANP does not allocate channel numbers itself, instead well known channel numbers are requested by the users on both sides, and arriving connections are accepted by the ANP if a process is listening on that channel number.

The following points outline the functionality of the ANP:

- Channel Setup: Each time a process wants to setup a new channel, the ANP daemon is contacted. It is provided a channel number by the process. It then contacts its peer on the called party, and negotiates the channel setup.

- Bandwidth Negotiation: The ANP daemon also watches upon the allocated bandwidth on the end system. The total bandwidth allocated cannot exceed the practical limit of the link rate or the PCR for the VC.

- Data Transfer: The ANP provides a transparent path for the data transfer on a channel.

- Channel Release: When the party at either end wishes to release the connection, the ANP at that end signifies it's peer of an end of transmission so that both ends can release the resources dedicated to the channel.

3.5 Real-time Transport Protocol - RTP

RTP is the Internet-standard protocol for the transport of real-time data, including audio and video. It has been designed within the Internet Engineering Task Force (IETF) [1]. It can be used for media-on-demand as well as interactive services such as Internet

Telephony. RTP consists of a data and a control part. The latter is called RTCP. If RTP packets are carried in UDP datagrams, data and control packets use two consecutive ports, with the data port always being the lower one. If other protocols serve underneath RTP (e.g. RTP directly over ATM AAL5), other schemes have to be used.

While UDP/IP is its initial target networking environment, efforts have been made to make RTP transport-independent so that it could be used, say, over CLNP, IPX or other protocols. RTP is currently also in experimental use directly over AAL5/ATM.

RTP does not address the issue of resource reservation or quality of service control; instead, it relies on resource reservation protocols such as RSVP.

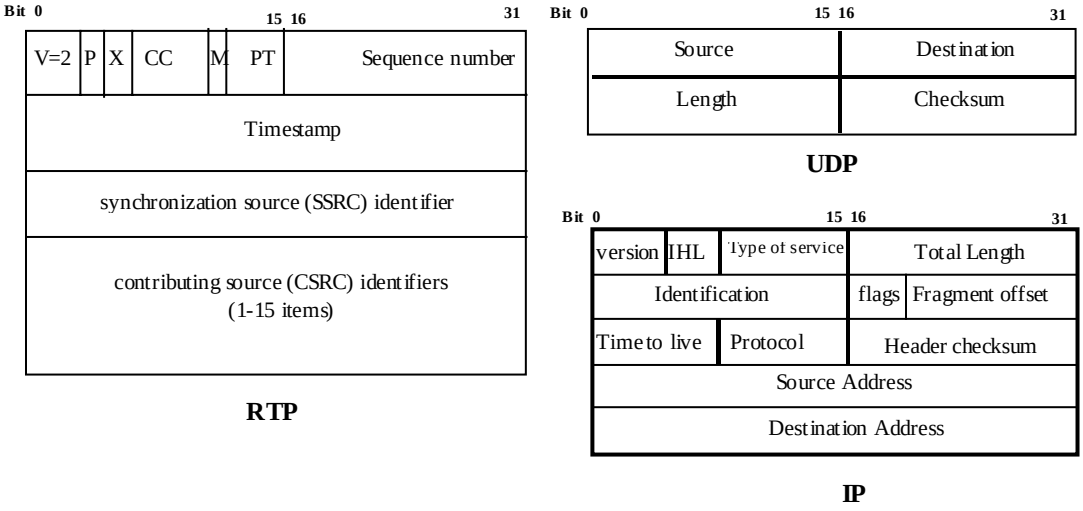


Figure 3.5 Header Structures: RTP, UDP, IP

3.5.1 RTP Data Packets

The data part of RTP is a thin protocol providing support for applications with real-time properties such as continuous media (e.g., audio and video), including timing reconstruction, loss detection, security and content identification.

The RTP header format is shown in Figure 3.5. RTP data packets consist of a 12-byte header followed by the payload e.g. a video frame or a sequence of audio samples. The payload may be wrapped again into an encoding-specific layer. The header contains the following information:

- *Payload type*: A one-byte payload type identifies the kind of payload contained in the packets, e.g. JPEG video or GSM audio.

- *Timestamp*: A 32-bit timestamp describes the generation instant of the data contained in the packet. The timestamp frequency depends on the payload type.

- *Sequence number*: A 16-bit packet number allows loss detection and sequence within a series of packets with the same timestamp.

- *Marker bit*: The interpretation of a marker bit depends on the payload type. For video, it marks the end of a frame, for audio, it marks the beginning of a talkspurt.

- *Synchronization source (SSRC) identifier*: A randomly generated 32-bit scalar that uniquely identifies the source within a session.

Some additional bit fields are not described here for brevity.

3.5.2 RTP Control Functionality

RTP offers a control protocol called RTCP that supports the protocol functionality. An RTCP message consists of a number of stackable packets, each with its own type code and length indication. Their format is fairly similar to data packets, the type indication in particular, is at the same location. RTCP packets are multicast periodically to the same multicast group as data packets. Thus, they also serve as a liveness indicator of session members, even in the absence of transmitting media data. RTCP is scalable and provides

support for real-time conferencing of groups of any size. RTCP provides the following functionality:

- QoS monitoring and congestion control
- Inter-media synchronization
- Identification
- Session size estimation and scaling

3.5.3 RTP Header Compression

Protocol header compression has been an active research area for the past couple of years especially after the maturity of the protocols and standards that drive audio and video streaming over the Internet. Besides IP and UDP, researchers have also investigated RTP header compression. Jacobson and Casner [11] proposed an approach for compressing RTP, UDP, and IP headers to be used over low-speed serial connections to the Internet.

This approach seems ideally suited to better bandwidth utilization of the media streams since the protocol overhead is significantly reduced. A few companies are currently working on the deployment of this approach inside the network interface infrastructure in order to improve multimedia conferencing over the Internet for terminals connected via low speed links.

3.6 Robust Audio Tool – RAT

RAT is a network audio tool that allows users to participate in audio conferences over the Internet. These can be between two participants directly, or between a group of participants on a common multicast group. No special features are required to use RAT in point-to-point mode, but to use the multicast conferencing facilities of RAT, a connection to the Mbone, or a similar multicast capable network, is required. RAT is based on IETF standards, using RTP [1] above UDP/IP as its transport protocol, and conforming to the RTP profile for audio and video conference with minimal control. In addition to the

features provided by other Mbone audio conferencing tools, such as vat, RAT offers the following additional functionality:

- Sender based repair of damaged audio streams
- FEC in the form of redundant packet transmission
- Support for interleaved audio
- Received based repair of damaged audio streams
- Adaptive scheduling protection
- Secure conferencing
- Improved statistics and diagnostic features
- Conference coordination bus
- Transcoder operation

It is supported on a range of platforms: FreeBSD, HP-UX, IRIX, Linux, Solaris, SunOS, and Windows 95/NT. The source code is publicly available for porting to other platforms and for modification by others.

Chapter 4

Design and Implementation

This chapter explains the important features of the proposed framework. The first section describes the implementation architecture and explains some of the design choices. The next section touches upon the salient points in the actual implementation.

4.1 Design

The following important issues are involved in the design of the proposed framework.

- Signaling required for setting up the voice flows
- Establishment of SVCs/AAL2 channels
- Data path for the transport of voice packets
- Header compression and
- Effect of the choice of coding used on transport efficiency

4.1.1 Architecture

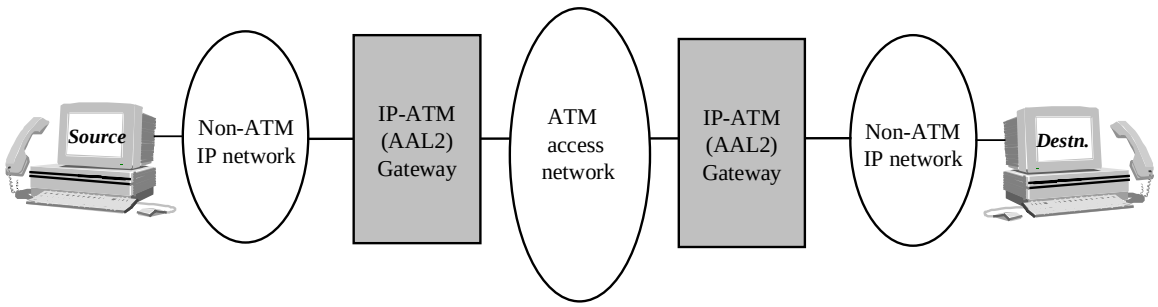


Figure 4.1 Reference Configuration – RTP over AAL2 framework

Figure 4.1 represents the reference configuration for the proposed framework. The framework exists on gateways on the periphery of an ATM access network. Voice flows, encapsulated in RTP headers, arrive at the gateways from an IP based network. These

flows are then sent through the ATM link in separate AAL2 channels, multiplexed over a single ATM SVC, after removing the IP/UDP headers and compressing the RTP header. At the other end, the IP packets are formed again using a table lookup and sent over the IP network to their destination.

The two main issues in the proposed architecture are:

- Signaling for channel setup and mapping
- Data path for the media streams

4.1.2 End-to-End Signaling

Signaling has been used for most voice applications since the early circuit based telephony systems, to set up the connection and a channel for voice flows, as well as to provide the necessary QoS resources along the path. This signaling extends to the packet telephony systems to establish paths between the different media gateways. ATM is a connection-oriented technology where signaling is used to establish virtual connections before any data flow is sent. Even though IP is based on a connectionless paradigm, the requirements for transporting voice have led to the development and usage of telephony/multimedia signaling protocols such as H.323 and SIP.

4.1.2.1 Signaling issues and requirements

H.323 as explained before, is a comprehensive suite of protocols for multimedia communications over packet networks. The use of H.323 over IP-ATM internetworks is being standardized by the ATM Forum (RMoA), which has proposed approaches for signaling through ATM networks, use of AAL5 SVCs between the two ATM gateways for efficient bandwidth utilization and QoS, and transport of compressed RTP streams over AAL5.

The RMoA approach proposes termination of the media streams on the gateways, along with the control streams. With media termination, the RTP packets are directed to local

interfaces on the gateways to simplify forwarding over the SVCs, for example by preventing the gateways from having to inspect all IP traffic crossing them.

The problem with this approach is the edge box now has to do some application level processing instead of being a pure layer 3/2 device. One option is to stay at layer 3, and establish VCs based on mechanisms such as RSVP or DiffServ.

Using H.323 for signaling implies an explicit association between the SVC setup and the "call/connection" at the H.225.0/H.245 level, which is not always necessary. It would be required for an end to end ATM case.

But for a network that has IP on the periphery, with an ATM cloud in the middle, other protocols and technologies could be used for signaling. When the signaling (RSVP for example) hits the ATM cloud, it causes a SVC to be set up. Incoming RTP packets are sent through this SVC, possibly after compression, and then completely reproduced, with all headers at the far end. They are then carried over normal IP to the destination. The existence of the SVC is completely transparent to the end users.

The advantage of using RSVP or DiffServ as a trigger for SVC establishment is that it is application independent. By making the setup and QoS mechanism H.323 specific, the scope of the service that can be provided is narrowed, the edge box is made more complex, and in general the end to end model is broken.

The H.245 method has the advantage of being very precise on what type of connection will be required at the SVC level. But it does have the disadvantage that the gateway has to maintain a substantial amount of software (the H.323 stack). As such, it would be a relatively complex device.

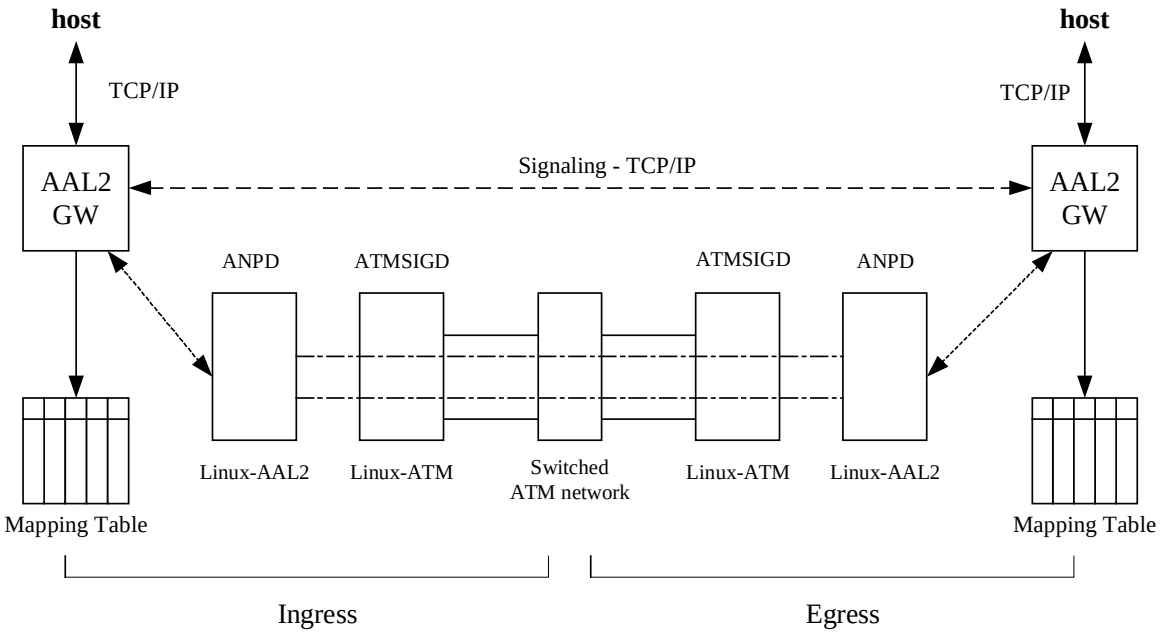
The emphasis in the current work is on the framework for transporting voice packets efficiently. Any appropriate signaling protocol can be used for end-to-end path establishment. With minor modifications, all these mechanisms can be made to interwork

with the current framework. The major functionality lies in the gateway modules, which do the actual SVC/channel setup and establish the necessary mappings. These modules just need a mechanism to communicate with hosts to receive requests and send responses, and with their peer gateways.

For the current framework, a simple end-to-end signaling protocol has been designed that is used to determine the parameters of the voice flow by the two sides. Some necessary parameters are destination port, packet length, and payload type (coding technique used). The messages also include the source and destination IP addresses. This model has been roughly adapted from the H.323 signaling model. However, the framework for data flow is not restricted to H.323 media only.

This signaling is used to set up the appropriate SVC/channels between the 2 gateways and to map the SVCs to the appropriate voice channels. This is done prior to the arrival of the actual media stream. An alternative to this is detecting a voice flow based on the source/destination address and port numbers and then setting up the SVC/channel and mapping. However, there is a certain delay in this process and packets have to be queued until the setup is completed. Also, the SVC and channel setup involves signaling which can be combined and integrated with any call signaling taking place between the end points.

The signaling architecture for the AAL2-RTP framework is as shown in Figure 4.2.



Signaling Architecture

Figure 4.2 Signaling Architecture

The source endpoint initiates signaling to establish a voice channel to the destination. Signaling here takes place in the following steps.

1. The source initiates a connection to the gateway nearest it, on the path to the destination. This gateway, called the *ingress gateway*, is the entry point into the ATM network. When the connection is established, the source sends the flow request parameters to the gateway. The signaling module on the gateway then determines the identity and address of its peer gateway for that particular destination and forwards the request to it. This process continues at the peer gateway, which in this case is the gateway at the other edge of the ATM network. The ATM network may either be one link or may pass through multiple ATM switches. The two gateways form the two ends of the network.

2. The peer ATM gateway, called the *egress gateway*, then establishes a connection with the destination host and passes the flow request to it. The destination responds with

the accepted parameters and also provides the destination port to which the source should direct the flow. The egress gateway returns this reply to its peer, along with its own ATM NSAP address. It also initiates a passive open for a channel through the ATM network from its peer.

3. The ingress gateway checks to see if an ATM SVC is already open to the other end. If so, it checks to see if there is enough bandwidth on that VC to satisfy this request. If there is not, then an ATM VC is opened by signaling. Once a VC has been established or if there is enough bandwidth available on an existing VC, the gateway initiates the opening of an AAL2 channel on this VC.

4. If an AAL2 channel for the flow is successfully set up, then the ingress gateway sends a message confirming the setup to the source that requested the flow along with the agreed parameters; else it returns a negative message to the source. The two gateways also add the flow parameters to the mapping table maintained in the kernel.

It is assumed that the egress gateway is determined by the signaling approach used, either using some IP gateway location protocol, through H.323 mechanisms or by RSVP means. Both the IP address and the equivalent ATM NSAP address of the peer gateway are thus known.

4.1.2.2 SVC/Channel setup and teardown

The setup and teardown of AAL2 channels is carried out by the AAL2 Negotiation Procedures daemon (*anpd*). The gateway module requests the *anpd* to open up a channel, passing the destination ATM NSAP address and channel negotiation parameters to it. The *anpd* sets up a channel transparently through the switched ATM network by communicating with its peer.

When a signaling request for a new flow arrives at the ingress, it checks if a SVC for the egress already exists. If it does not, then a new SVC is setup as explained above. If it finds an entry for the gateway, it checks if enough bandwidth is available on the SVC and

opens another channel for this flow. The gateway module then adds an entry into the mapping table.

Signaling is also used for terminating a connection. At the ATM gateways, the termination messages are used for tearing down the channel and deleting the corresponding entry in the mapping table. Appropriate AAL2 techniques are used for channel setup, teardown and maintenance.

4.1.3 Data path for voice streams

The voice streams pass through the framework from the IP layer and are handed off to the AAL2 layer after lookup, header compression and re-packetization.

4.1.3.1 Mechanisms for transport over ATM

The usual way of transporting IP packets through an ATM network is using IP over ATM technologies like CLIP [21] or LANE [22]. Though this approach works fine for variable size data packets, it is sub-optimal when used to transport voice packets which are usually very small. It is highly bandwidth inefficient because of the significant overhead introduced by using RTP (at least 12 octets), UDP (8 octets) and IP (at least 20 octets). The multiprotocol encapsulation header for CLIP (RFC 1483) adds another 8 octets. LANE also adds a header with its own overhead. It also does not utilize the inherent Quality of Service capabilities of ATM.

To reduce protocol overhead and to increase the bandwidth utilization/efficiency while providing the voice stream with necessary QoS, an RTP over ATM approach has been suggested in H.323/Annex C. It gets rid of the UDP/IP overhead by using AAL5 directly to transport media streams on RTP. However, it is still quite bandwidth inefficient for voice sessions because of the significant overhead of the RTP protocol. Also, it is intended for use only in an end-to-end ATM network.

The signaling architecture defined by the ATM Forum terminates the UDP/IP protocols on the media gateways, removing the need to transport these headers end-to-end. Both gateways maintain a mapping of RTP flows to the ATM SVC and also the necessary transport address information for re-creating the packet at the destination end of the ATM network. Since the UDP/IP headers are not sent over the ATM SVC, only RTP header compression has to be used. This results in a greater efficiency as compared to the earlier cases. However, this approach too has a few shortcomings, which are touched upon in some of the following sections.

AAL2 provides inherent multiplexing capabilities within an ATM SVC, which make it simpler to use multiplexing mechanisms between 2 gateways/access routers and also increase the bandwidth efficiency as compared to the compressed RTP over ATM approach. It also benefits more than AAL5 from RTP header compression as shown later.

4.1.3.2 Flow identification

As mentioned above, the media stream from the source is terminated on the ATM gateway. Hence the voice packets have to go up to the application level and then back down the protocol stack in the two gateways. This results in a lot of processing overhead in the protocol layers. In addition, this approach assumes the existence of a media gateway with application level signaling such as H.323 and was originally intended for an ATM backbone network.

However, the framework proposed here can be used with any signaling method and need not always go through a media gateway. Also, its usage is much more oriented towards low-speed access networks. It would also be much more beneficial if this application level processing was avoided.

The alternative is to identify flows at the network layer in the kernel protocol stack. Typically, this is done by some combination of source IP address and port number, destination IP address and port number, and protocol type.

For RTP and VoIP traffic classification, vendors use a fixed range of UDP port numbers. For example, Cisco uses UDP port numbers from 16384 to 32767 for RTP flows. In addition, classification is increasingly being done in hardware after which, the packets are sent to the appropriate module via an internal bus; hence this is a very fast operation.

The other option to identify RTP flows here is to assign a separate protocol ID for it at the link layer, so that such packets can be directly handed off to the framework. But this has not been done, as it would require modifications to the protocol stack at all transmitting and receiving hosts to set the protocol ID. Hence the source and destination addresses and port numbers are used to identify flows.

4.1.3.3 Mapping RTP flows to AAL2 channels

Mapping tables are maintained at both the ingress and egress gateways. They contain information to identify a flow when packets arrive and to send them over the appropriate AAL2 channel. Since, the packets are transferred from the IP to the ATM network and back again directly in the kernel, each packet has the entire IP/UDP header overhead. Since the headers are not required for transport over ATM, they are stripped off. Their presence, in fact, increases the packet size beyond the range of AAL2 CPS packet sizes. The headers are required, however, on the other end for transport over the IP network. Hence, when the first packet in a flow arrives, the IP/UDP headers are stored in the mapping table on both sides. For the remaining packets, the headers are not sent. Instead, the complete packets are generated at the other end using the stored headers and some additional information sent with the data.

4.1.4 RTP Header Compression

The RTP header is not stored in the mapping table. Instead it is sent along with each packet. This is because the RTP header contains the sequence number and timestamp, which change from packet to packet. These are important for the receiver to maintain

synchronization with the sender. Hence even after removing the IP and UDP headers, there is still an overhead of 12 bytes per voice packet due to the RTP header.

RTP header compression plays an important role in reducing this overhead. The RTP header compression technique described in [11] has been partially implemented here, with modifications to suit it to AAL2.

The big gain in this compression comes from the observation that, although several header fields change in every packet, the difference from packet to packet is often constant and therefore the second order difference is zero. By maintaining both the uncompressed header and the first order differences in the session state shared between the compressor and the decompressor, all that must be communicated is an indication that the second order difference was zero. In that case, the decompressor can reconstruct the original header without any loss of information simply by adding the first order differences to the saved uncompressed header as each compressed packet is received.

In the RTP header, the SSRC identifier is constant in a given context since it is part of what identifies a particular context. For most packets, only the sequence number and the timestamp will change from packet to packet. If packets are not lost, the sequence number will increment by one for each packet. For audio packets of constant duration, the timestamp will increment by the number of sample periods conveyed in each packet. The first order differences of these fields is transmitted to the decompressor along-with the SSRC identifier.

Note that in each of these cases the second-order difference of the sequence number and timestamp fields is zero. So the next packet header can be constructed from the previous packet header by adding the first-order differences for these fields that are stored in the session context along with the previous uncompressed header. When the second-order difference is not zero, the magnitude of the change is usually much smaller than the full number of bits in the field, so the size can be reduced by encoding the new first-order difference and transmitting it rather than the absolute value.

The M bit will be set on the first packet of an audio talkspurt. If it were treated as a constant field such that each change required sending the full RTP header, this would reduce the compression significantly. Therefore, one bit in the compressed header will carry the M bit explicitly.

If the audio packets are flowing through an RTP mixer, then the CSRC list and CC count will also change. Changes to the CSRC list and CC count need to be communicated explicitly, hence 1st order differences need not be maintained. In cases where RTP mixing is not used, these fields will not have any effect on the compression.

4.1.4.1 The compression protocol

The RTP header compression protocol consists of two finite state-machines, one for the transmitter side shown in Figure 4.3 and the other for the receiver side shown in Figure 4.4. One instance of each state-machine will exist for each channel on both sides on the link.

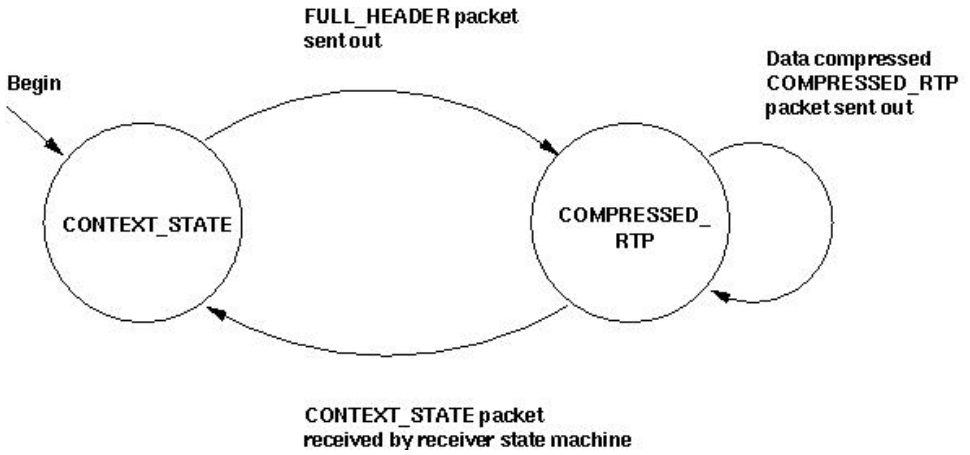


Figure 4.3 RTP Header Compression: Transmitter

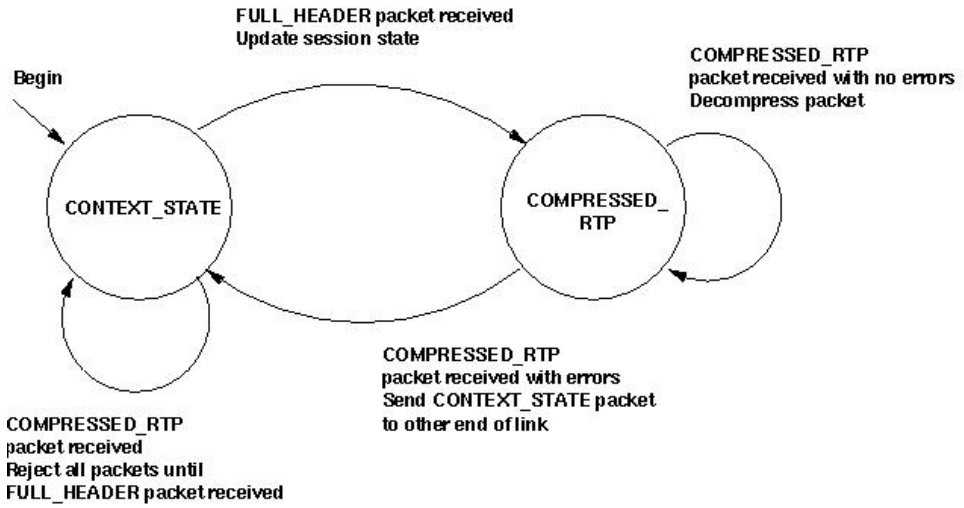


Figure 4.4 RTP Header Compression: Receiver

The compression protocol must maintain a collection of shared information in a consistent state between the compressor and decompressor. There is a separate context for each RTP packet stream as defined by the RTP SSRC field.

The context information for a flow consists of the previous RTP header, the first order difference of the Timestamp field and the link sequence number (C/D). The 4-bit link sequence number is used to detect packet loss between the compressor and decompressor. Each context has its own separate sequence number space.

RTP Header
First order difference for the RTP time stamp field
Link Sequence number value

Figure 4.5 RTP Compression Call Context

In order to communicate packets in the various uncompressed and compressed forms, this protocol introduces four new packet formats. As with the IETF and ATM Forum

approaches, the RTP-only compression relies on the link layer (AAL2, here) being able to provide an indication of the different packet formats.

1. RTP_FULL_HEADER - communicates the uncompressed RTP header and data to establish the uncompressed header state in the decompressor for a particular context.

2. RTP_DELTA_HEADER - communicates the new 1st order differences for the RTP header fields indicated since they incurred a non-zero 2nd order difference. The header for this type of packet contains a 4-bit flag field to indicate which fields in the RTP header changed, followed by the respective new 1st order differences. This 4-bit flag field is named the MSTC sequence.

   The M bit carries the original M bit of the RTP header. The S and the T bits signal the new 1st order differences for the sequence number and the time stamp, respectively. The C bit indicates changes to the RTP CSRC list. Note that the RTP sequence number is expected to always increase by one and therefore if the S bit is set, the decompressor uses the new delta only once.

3. RTP_COMPRESSED_HEADER - indicates that the RTP header has been fully compressed, i.e., all changing fields have actually had a 2nd order difference of zero.

   The use of a packet type to indicate a fully compressed packet is intended to improve the performance of the common case; it is expected that null 2nd order differences will be frequent.

4. RTP_CONTEXT_STATE - is a special packet sent from the de-compressor to the compressor indicating that the context associated with the SVC may have been invalidated. The compressor is expected to send the next packet as a RTP_FULL_HEADER packet.

4.1.4.2 RTP header compression transport on AAL2

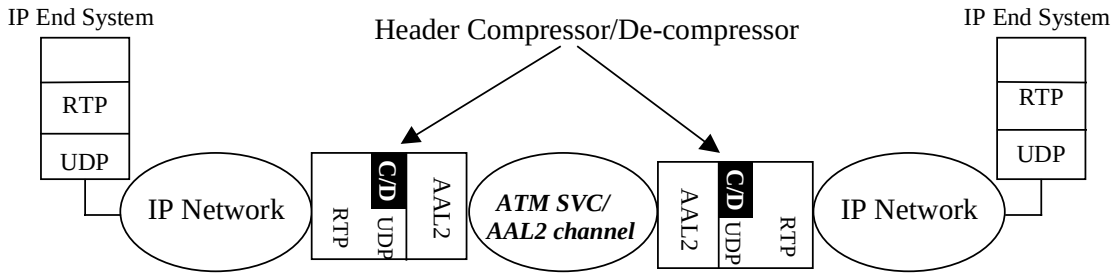


Figure 4.6 RTP Header Compression on AAL2

In the case of ATM, because every channel is uniquely identified by the combination of VPI/VCI and the AAL2 CID, it is not necessary to maintain a separate context id for each flow.

The AAL2 CPS header has a 5-bit UUI field that is used to carry SSCS information between 2 AAL2 peer entities. The SSCS field can also be null. Since, this field is not being used by SSCS entities in the current setup, it can be used to provide information about the RTP packet type contained in that CPS packet. The 3 LSB bits of the UUI field are used to provide the information.

The different packet types are:

Packet Type	**UUI field bits: 4 3 2 1 0**
RTP_FULL_HEADER	1 0 0
RTP_CONTEXT_STATE	0 1 1
RTP_DELTA_HEADER	0 1 0
RTP_COMPRESSED_HEADER	0 0 1
NORMAL AAL2	0 0 0

The AAL2 CPS-PDU contains the RTP header extension, if any, and the RTP payload.

One byte (the first) in the CPS PDU is used to convey information about the fields that changed and also the link sequence number. The MS four bits are used to indicate changes in the M, S, T and C fields and the LS four bits are used to carry the link sequence number. The link sequence number should increment by 1 for each consecutive packet.

The other fields of the RTP header (version, P bit, X bit, payload type and SSRC identifier) are assumed to remain relatively constant. In particular, the SSRC identifier is defined to be constant for a given context because it is one of the factors selecting the context. If any of the other fields change, the uncompressed RTP header will be sent.

4.1.4.3 Error Recovery

Whenever the 4-bit sequence number for a particular context increments by other than 1, the decompressor invalidates that context and sends a CONTEXT STATE packet back to the compressor indicating that the context has been invalidated. All packets for the invalid context are discarded until a FULL HEADER packet is received for that context to re-establish consistent state. If multiple compressed packets arrive in the interim, the decompressor does not retransmit the CONTEXT STATE packet for every compressed packet received.

4.1.4.4 Compression of RTCP Control Packets

RTP convention dictates that data is carried on lower port number and the corresponding RTCP packets are carried on the next higher port. For RTCP, the compression could apply, but would involve maintaining more state. The RTP protocol suggests that the RTCP packet interval be scaled so that the aggregate RTCP bandwidth used by all participants in a session will not be more than 5% of the session bandwidth. Hence there is not much to be gained from RTCP compression. The current work does not perform RTCP compression.

4.1.5 IP/UDP headers

By not sending IP and UDP headers over the AAL2 channels, a lot of protocol overhead is eliminated. However, there are some changing fields in both these headers that have to be considered.

In the IPv4 header, only the total length, packet ID, and header checksum fields will normally change. The total length is redundant, as the AAL2 layer provides it. The packet ID is used for handling fragmentation, which would not occur for the small size voice packets under consideration. If the source sending the voice packets does not insert a new ID for every packet, then the packet ID can be ignored. Usually, the packet ID increments by one or a small number for each packet. To maintain lossless compression and also to avoid checksum errors at the other gateway, the changes in the packet ID are transmitted.

The checksum is not transmitted. Instead it is calculated at the other end, after the IP packet has been re-created.

In the UDP header, the length field is redundant due to reasons given above. The UDP checksum field will be a constant zero if the source elects not to generate UDP checksums. Otherwise, the checksum must be communicated intact in order to preserve the lossless compression and to maintain end-to-end error detection capabilities.

Since a source will typically include checksums on all packets of a session or none of them, its presence is determined at the start of the flow and then included in all subsequent packets, if necessary.

4.1.6 Codecs and packet sizes

Audio codecs normally considered for voice transport over IP are G.711 (PCM), G.722 (SB-ADPCM), G.723.1 (MP-MLQ/ACELP), G.728 (LD-CELP), and G.729 (CS-ACELP). Maximum packet sizes are to be chosen for these audio codecs. The important

implications of this choice are transport efficiency, end-to-end delay and possible voice quality degradation caused by packet losses.

The focus here is on transporting RTP voice packets efficiently over ATM using RTP header compression, hence greater emphasis is provided on the impact of maximum packet sizes on transport efficiency.

Since H.323 is the prevalent standard for IP voice transport, the guidelines provided by the ATM Forum are used here while considering the capabilities of AAL2.

The default packetization interval for the voice traffic is 20 ms for any voice codec, unless the codec itself cannot accommodate this value (e.g., for G.723.1 which has a voice frame size of 30 ms by definition). This means that the default voice packet size (in octets) varies for every voice codec. However, this size can be negotiated during signaling.

The packet overhead decreases as the number of frames per packet increases. Intuitively, high bit-rate audio has slightly less overhead when compared to low bit-rate packets. As the number of audio frames per packet increases, the packet overhead decreases. The overhead decrease is explained by more information bytes (audio frames) being included in a packet with a fixed header.

The latency increases as the number of frames per packet increases. This is expected since more audio frames require more time to be captured and buffered. Hence, a tradeoff exists between packet overhead and local latency for audio packet transfer.

A variety of frame sizes and packet sizes can be negotiated and chosen for the different codecs in use. However, as mentioned above, there is a tradeoff between protocol overhead and delay in packetization.

4.1.6.1 A Comparative Analysis of AAL5 and AAL2 for different codecs

The table below shows the different packet sizes for various codecs supported by H.323 for their default packetization interval [4].

Encoding	**G.711**	**G.722**	**G.723.1**		**G.728**	**G.729**
	PCM	**SB-ADPCM**	**MP-MLQ/ACELP**		**LD-CELP**	**CS-ACELP**
Rate (kbit/s)	64	64	6.3	5.3	16	8
Frame size (ms)	1	1	30	30	2.5	10
Frame size (octet)	8	8	24	20	5	10
Default packet size (ms)	20	20	30	30	20	20
Default Frames per packet	20	20	1	1	8	2
Default payload size (octet)	160	160	24	20	40	20

Table 4.1 Default voice packet sizes for standard codecs

The high bit rate codecs generate packets which are larger than the packet sizes supported by AAL2 (default maximum SDU size is 45 bytes; if greater, only 64 bytes is allowed) Hence only the low bit rate codecs are considered for the comparison.

Consider the G.723.1 encoder, the preferred low bit rate codec for H.323 transport.
The G.723.1 encoder provides one frame of audio every 30 milliseconds. The audio frame size is 20 bytes for the low rate (5.3 kb/s) and 24 bytes for the high rate (6.4 kb/s). A fixed number of frames are sent per audio packet, the default for G.723.1 being 1. Increasing the number of frames per packet improves bandwidth utilization and reduces network packet overhead. However it also introduces added delay for playback.

Each codec packet is then appended to an RTP header, incurring a minimum of 12 octets of overhead. Over an ATM network, if direct AAL5 is used, at least another 8 octets of overhead are added due to the AAL5 trailer. The AAL5 payload can be further padded

with unused octets to make the AAL5 frame an integral multiple of 48 bytes for segmentation into ATM cells.

Naturally, normal IP over ATM (CLIP) transport incurs more protocol overhead due to the presence of the IP and UDP headers.

The table below presents a comparison between AAL5 and AAL2 for the resulting bit rates when the default packet sizes shown above are used with and without RTP header compression.

Encoding - G.723.1 (5.3 kbps)	**No RTP Header Compression**		**RTP Header Compression**	
	AAL5	**AAL2**	**AAL5**	**AAL2**
Default packet size (ms)	30	30	30	30
Default packet size (octet)	20	20	20	20
RTP header bytes	12	12	0	1
AAL overhead (octet) *	8	3	8	3
Total bytes required	40	35	28	24
Cells (Bytes required)	1 (53)	1 (40)	1 (53)	1 (29)
Unused bytes	8	0	20	0
Packets/sec	33.33	33.33	33.33	33.33
Bandwidth required(kbps)	14.131	10.665	14.131	7.733

* AAL5 Trailer/ AAL2 CPS Header

Table 4.2 Bandwidth Requirements for G.723.1 codec – AAL5/AAL2

It should be noted that the values given above are for the ideal case assuming that no padding results for AAL2. The actual bandwidth used may be more depending on the efficiency of AAL2 multiplexing and the number of flows present. The 1-byte STF field for AAL2 has also not been considered here because the overhead caused by it is divided among the different packets that make up the cell.

Also, the minimum RTP overhead is considered here. But still the table serves to show the inherent efficiency in AAL2 for transporting small size packets. It also indicates that even though the same RTP header compression can be used for AAL5 as well, it is not as effective as it is for AAL2.

Similar comparisons for the other low bit rate codecs are given in the tables below.

Encoding	**G.723.1 (6.3 kbps)**		**G.728**		**G.729**	
RTP Header Compression	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
Default packet size (ms)	30	30	20	20	20	20
Default packet size (octet)	24	24	40	40	20	20
RTP header bytes	12	0	12	0	12	0
AAL5 overhead (octet)	8	8	8	8	8	8
Total bytes	44	32	60	48	40	28
Cells (Bytes required)	1 (53)	1 (53)	2 (106)	1 (53)	1 (53)	1 (53)
Unused bytes	4	16	36	0	8	20
Packets/sec	33.33	33.33	100	50	50	50
Bandwidth required(kbps)	14.131	14.131	42.40	21.20	21.20	21.20

Table 4.3 Bandwidth Requirements for AAL5 with/without RTP Header Compression

It is only for the G.728 codec that RTP header compression results in a bandwidth reduction for AAL5. Further, even for this case, the DELTA_RTP packets would require the same number of cells as FULL_HEADER packets.

Encoding	**G.723.1 (6.3 kbps)**		**G.728**		**G.729**	
RTP Header Compression	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
Default packet size (ms)	30	30	20	20	20	20
Default packet size (octet)	24	24	40	40	20	20
RTP header bytes	12	1	12	1	12	1
AAL2 overhead (octet)	3	3	3	3	3	3
Total bytes required	39	28	55	44	35	24
Cells (Bytes required)	1 (39)	1 (28)	2 (55)	1 (44)	1 (35)	1 (24)
Unused bytes	0	0	0	0	0	0
Packets/sec	33.33	33.33	50	50	50	50
Bandwidth required(kbps)	10.398	7.466	22.00	17.6	14.0	9.6

Table 4.4 Bandwidth Requirements for AAL2 with/without RTP Header Compression

In the table for AAL2, the G.728 codec payload is greater than the default maximum SDU size in the absence of compression. So, its inclusion may not be wholly appropriate but is shown only for comparison.

As can be seen, the default sizes may not be suitable for transport over ATM, especially AAL5 because of a fixed small cell size. If AAL5 is used as the adaptation layer over ATM, it is seen from the table that only G.728 codec can be used with the default values. The other codecs need some changes to be done to the default frame size or the number of frames per packet to reduce the protocol overhead even in the absence of RTP.

If the packet size is too small, then the remaining part of the cell is wasted in padding. If it exceeds a cell size, then the packet spills over into two cells and again the second cell is padded (up to a maximum of 47 bytes).

But changing the default values in order to better match the protocol overhead characteristics of ATM may lead to increased latency at the host to fill up the packet.

If AAL2 is used instead, it can be seen from the table that the default packet sizes for most codecs fall within the one cell size limit (except G.728). Because of the multiplexing capabilities of AAL2, more than one packet can be sent in the same cell with possible overlap between cells. Hence even with the default values specified, AAL2 will provide lesser protocol overhead and hence better bandwidth utilization while maintaining the original delay characteristics of the codec at the source.

One important issue here is the effect of the AAL2 timer_CU, which introduces additional delay in the network. Studies [16] have shown that the optimum value for the timer is 1-2 ms. Previous tests have shown that the average delay in the AAL2 transmitter is also in the range of 1-2 ms. This value is within acceptable limits. Hence, it is advantageous to use AAL2 in the described context.

With the introduction of RTP, additional 12 bytes of header are added to the calculations. However, it must be noted that, with RTP header compression, most of the voice packets will have headers fully compressed, i.e., no RTP header is sent. Hence the above observations hold good in this case too.

4.2 Implementation

This section describes the implementation details of the AAL2-RTP framework. The first subsection explains in brief some of the modifications carried out for providing real-time capabilities to the ATM Adaptation Layer 2 (AAL2) support in the Linux kernel. The next subsection describes the RTP over AAL2 framework in detail.

4.2.1 Real-Time Modifications to AAL2

4.2.1.1 Temporal Modifications

A timer, *timer_CU*, is associated with every AAL2 VC. On expiry of this timer, an unfilled cell is transmitted with padding, instead of waiting for more CPS packets to arrive, thus limiting the experienced delay for existing packets in the cell. The optimum values for this timer are in the range of 1-2 ms.

In the Linux-ATM protocol stack, the *struct atm_vcc* data structure consists of all the elements and data pointers needed to set up, maintain and modify an ATM virtual connection. A number of additional elements have been added to this structure to support AAL2, one of which is the *struct timer_list.*.

If any subsystem of the Linux kernel needs asynchronous notification, it creates a timer and adds it to a list maintained by the kernel. The structure of a timer list is given below.

```
struct timer_list {
     struct timer_list *next;
      struct timer_list *prev;
      unsigned long expires;
      unsigned long data;
      void (*function)(unsigned long);
}
```

Each timer has an *expires* field and a *function* field. The *expires* field specifies when the asynchronous notification represented by the timer is to happen and the *function* field specifies the function that should be called when the timer expires. Each timer also has a *data* value that is passed to the function when it is called. The timers are kept in a doubly linked list sorted in ascending order of time. Functions are provided to add and delete timers from this list.

UTIME [12], developed here at ITTC, improves the temporal resolution of the Linux kernel. The standard Linux timing mechanism provides scheduling resolution on the order of 10 milliseconds, called a *jiffy*. This resolution is not suitable for real-time applications like voice. UTIME modifies Linux to provide a microsecond-resolution time sense, without significantly increasing the overhead of the software clock. Hence it increases the utility of Linux for soft real-time applications.

With UTIME, an additional member *usec* is added to the timer structure. This member specifies the microsecond within the *jiffy* at which the timer will expire. Also, a global variable, *jiffies_u* is used to maintain the microsecond within the current jiffy.

These variables were used as given below, their usage determined by experimentation.

```
struct atm_vcc {
  :
  :
  struct timer_list timer_cu;
  :
};

/* Initialize timer values */
  vcc->timer_cu.expires = jiffies;
  vcc->timer_cu.usec = jiffies_u + 2000; /* 2 msec */
/* Add timer to the list */
  add_timer(&vcc->timer_cu);
```

4.2.1.2 Additional Modifications

With enhanced timer resolution, synchronization problems arose with the original transmitter algorithm implementation. Because the timer was an asynchronous event, it expired at arbitrary points in the transmitter code, leading to certain critical variables and states being set to incorrect values. Hence, cells were filled erroneously at the transmitter. This led to improper operation at the transmitter as well as at the receiver, where packets were dropped erroneously.

The transmitter state machine implementation was modified for it to function without errors and synchronization problems. In addition, semaphores were used for some critical execution sequences.

4.2.1.3 Receiver Modifications

In the AAL2 receiver, the flow of data to the various channels is controlled through the signaling channel, which receives the cells arriving on that VC, and de-multiplexes them to retrieve individual packets. It then assigns the packets to the appropriate channels. In the original implementation, this took place under the control of the *anpd* through a user level read operation. The read, performed by the *anpd* every time a cell was received on its VC, would initiate the receiver operation, where channel de-multiplexing was done on the arrived cell. The individual channels receiving packets were then signaled to receive the packet. Each channel, controlled by a user level read, would strip off the CPS header and read the actual voice data. Packets meant for the *anpd* would be read directly by it.

Because this process was invoked by a user level system call, it was slow and caused a delay in reading the data by the individual channels, leading to a drop in both individual and overall system performance. The operation would be much faster if the control remained in the kernel.

Hence, a driver level *aal2_push* function was implemented, specific to VCs opened by the AAL2 *anpd*. This function, which is called by the ATM driver whenever a cell for the *anpd* VCC arrives, initiates the AAL2 receiver directly in the kernel. As packets are de-multiplexed from the cell, the individual channels, including the *anpd* control/signaling channel are signaled to read the voice packets meant for them. This has led to a much faster processing of the cell and improved performance.

In order to avoid a lot of this processing being done in the driver, the receiver is called asynchronously by the driver by means of a task queue, every time a cell is received for this VCC.

```
queue_packet(vcc->cell_queue,packet);
queue_task(&vcc->rqueue, &tq_scheduler); /*invoke task queue */
```

4.2.2 AAL2 Gateway Module

As mentioned earlier, mapping tables are maintained at both the ingress and egress gateways. The mapping table format is as shown in the figure below.

Source		Destination		MAP ID		Payload Type	Channel Length	IP/UDP Headers	RTP Header Comp.
IP Address	UDP Port	IP Address	UDP Port	ATM VCC	AAL2 CID				

Figure 4.7 RTP Flow - AAL2 Channel Mapping

Flow classification can be done using some or all of the source and destination information present in the table. Currently for this implementation, the destination IP address and UDP ports are used to identify voice flows. Each flow/voice stream is mapped onto an AAL2 channel on a particular ATM VC. Hence the combination of the two serve to uniquely map the incoming AAL2 packets to the corresponding RTP flow at the egress. The channel length gives an indication of the maximum SDU size expected on the AAL2 channel. It is used for validation purposes in the AAL2 receiver. The IP and UDP headers are maintained on both sides. At the egress, after de-multiplexing in the AAL2 receiver and decompression of the RTP header, the IP/UDP headers for that flow are attached to the data portion to re-create the IP packet. Context state information is maintained for RTP header compression, including the last instance of the RTP header and the relevant first order differences. These are used to detect changes in the RTP header at the ingress and to recreate the RTP header at the egress.

4.2.2.1 Signaling

The source initiates call signaling by opening a TCP/IP socket to the gateway and issues a OPEN_CHANNEL request, passing the destination IP address and other parameters. The gateway module consists of a server that constantly listens on two sockets for incoming requests – one from hosts and another from peer gateways. It spawns appropriate processes to service those requests. On receiving the OPEN_CHANNEL message from a source, the gateway opens a TCP/IP connection to the peer gateway for that call, and forwards the message. The two gateways are connected over the ATM network, through an ATM PVC. The peer gateway, i.e. the egress, receives this request message on its peer socket and forwards it to the destination address mentioned in the message.

The destination host or the receiver responds with an OPEN_CHANNEL_ACK or OPEN_CHANNEL_REJECT message. If it can set up the voice call, it sends back its UDP port number to which the source should send the flow.

The egress returns this message to its peer. If it is an OPEN_CHANNEL_ACK, then it also sends the ATM NSAP address for the interface on which the ingress sent the request. The ingress, on getting this reply, opens an AAL2 channel for this flow. If successful, it adds the entry into the mapping table in the kernel and returns the OPEN_CHANNEL_ACK to the source along with the destination port number, else it returns an OPEN_CHANNEL_REJECT.

The important functions of the gateway modules are:

1. Obtain the necessary flow parameters from the end-to-end signaling messages. The parameters include the source/destination addresses and ports, payload type and packet length.
2. Initiate the establishment of an ATM SVC/AAL2 channel. This is done by opening an AAL2 socket, setting the relevant AAL2 socket options and finally issuing a *connect*

system call on the socket at the ingress gateway. At the egress, a corresponding *accept* call is issued.

Then, the channel is set up via the ANP negotiation procedures [] as described earlier. If for any reason the channel cannot be set up, *connect* fails and an OPEN_CHANNEL_REJECT message is returned to the source in response.

3. Add the corresponding entry into the AAL2_MAP table in the kernel. This is done by means of an implemented *ioctl* system call. The file descriptor for the opened AAL2 socket is stored for later reference, along with the flow identifiers.

4. When the source terminates a flow/call, it sends a CLOSE_CHANNEL message. The ingress gateway forwards this message to its peer and at the same time, initiates a *close* system call on the AAL2 socket for that channel. It retrieves the previously stored socket file descriptor based on the flow identifiers passed in the request.

5. The peer gateway passes along the message to the destination. It also initiates a *close* on the AAL2 socket at its end.

6. Both gateways then delete the corresponding entries in the kernel AAL2_MAP table, by issuing an *ioctl*.

4.2.2.2 Modifications to the ANP daemon

In the existing AAL2 Negotiation Procedures daemon, the channel IDs were set by the applications that open AAL2 sockets to some destination, by issuing a *setsockopt* system call. The ANP daemon checks whether a channel with that ID exists and returns an error if it does. This requires the presence of a channel at the other end, which has set the same channel ID. However, in the present framework, the gateway modules on both sides open AAL2 sockets whenever a request for that particular path is received. They are not aware if an ATM SVC to that destination exists or what channels are free on that SVC.

Hence, the ANP daemon was modified to maintain a list of possible channel IDs on each SVC opened and to keep track of available channel IDs. Whenever a *channel_open* request is made, the ANP daemon scans through the list and returns a free channel ID. This channel ID is also passed to the peer ANP daemon to assign it to the corresponding channel opened by the peer gateway module. To enable identification of the correct channel on the other side, additional parameters relating to flow identification were added to the messages exchanged by the ANP daemons.

4.2.2.3 Media Transport

This section describes some of the implementation details for the data path of the voice flows and briefly traces their passage through the protocol stack at both the ingress and egress gateways.

The figure below shows the schematic flow for packets through the protocol layers.

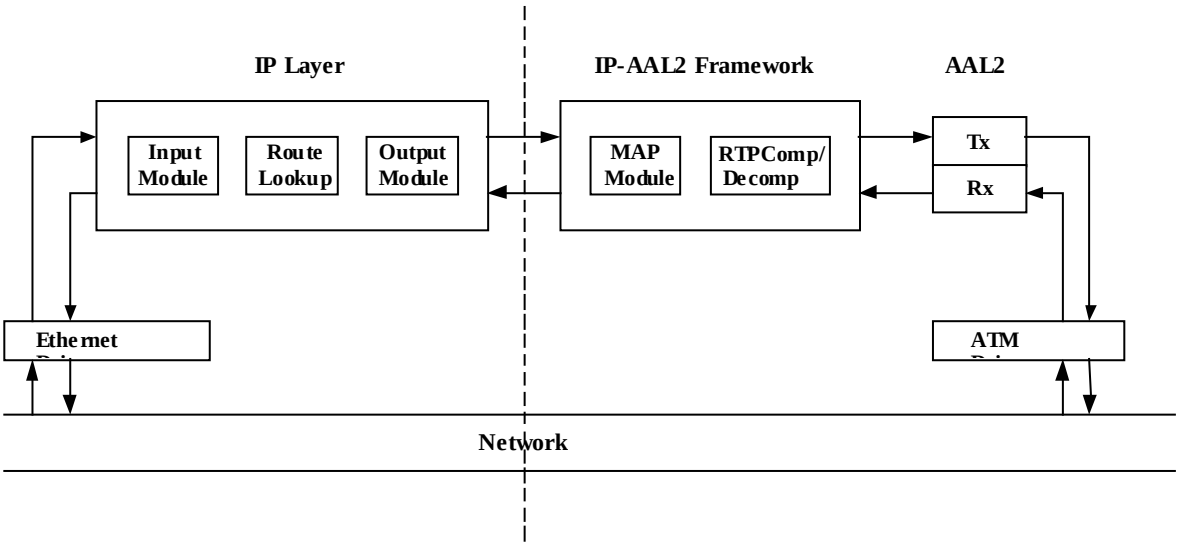


Figure 4.8 Data flow for RTP packets through framework

When the RTP packets for the voice flow arrive at the source gateway, say over an Ethernet connection, the Ethernet driver hands over the packet to the IP layer via the network bottom-half routine.

In the IP input module, the protocol header is checked for correctness by doing a checksum. If the header has any options, they are processed. Next, a check is done to determine if the packet is meant for the local host (i.e. for itself). If the packet is not meant for itself, it is passed on to the forwarding module.

In the forwarding module, the packet's TTL is checked and decremented. Then a routing table lookup is done to determine the next hop and the output interface for the packet. Fragmentation, if present, is handled. The packet is then sent off to the network device associated with the output interface. This is where the RTP over AAL2 framework comes into the picture.

If the output interface is the ATM interface over which the two gateways are connected by an ATM PVC, the network device would be the CLIP device set up at the interface. When the packet arrives at this point, a quick check is done to see if it is a UDP packet and if the port is an even port. If this check succeeds, then a further lookup is done to determine if the packet is an RTP flow for which a mapping has been established.

If the packet is identified to be a constituent of an RTP flow, and if it is the first packet for that flow, then the UDP and IP headers are stored in the mapping table. Since the other gateway also needs a copy of the two headers, one or more AAL2 CPS packets are constructed, as necessary, with the two headers making up the payload. They are then sent to the peer over the channel set up for the flow.

The IP and UDP headers are then stripped off, and the packet given to the RTP Compression/Decompression module. For the first packet in the flow, the full RTP header has to be sent for the receiver to establish context state. The full header is also

stored in the compressor. The link sequence number and other state information are initialized. A packet type of RTP_FULL_HEADER is indicated.

For the remaining packets in the flow, after header removal, the RTP compressor checks the header fields for changes. Depending on changes to the M, S and T fields, a RTP_DELTA_HEADER or RTP_COMPRESSED_HEADER type is sent. An indication of what fields changed is sent (MSTC) in the most significant four bits of the first byte in the CPS payload. The link sequence number is incremented by one for each consecutive packet and sent in the least significant four bits. The change in the RTP sequence number is encoded in a single byte as it would be a small number (greater than one). Differences in the timestamp field are encoded in two bytes and are stored in the context if they are different from the previous state. If the difference is too large, the context state is reinitialized by sending the full RTP header. The full header is also sent if any changes occur in the remaining header fields. The context is updated by storing the current header.

A couple of optional data fields in the AAL2 CPS packet are the changes in the IP ID field, which takes a byte, and the 16-bit UDP checksum. These are sent only if the source machine includes them in the packets it generates.

The AAL2 CPS PDU is then assembled. The indicated RTP packet type is put in the UUI header field. The AAL2 CID and the LI fields are also filled and HEC is calculated. The packet is then given to the AAL2 transmitter for insertion into a cell and transmission.

At the egress, the ATM driver initiates the AAL2 receiver by means of a *task queue*[2] in the *aal2_push* operation, and hands over arriving cells to it. The AAL2 receiver breaks down the cell and retrieves the constituent packets. Overlapping packets are stored until the next cell arrives with the remaining bytes. The packets are then put into individual channel queues and handed over to the IP-AAL2 framework module by means of another task queue.

[2] A *task queue* is a structure available in the Linux kernel that is used to store tasks to run at a later time. It can be used in different modes, which determines when the scheduler invokes it.

The AAL2 receiver had to be slightly modified here to handle a kernel-level forwarding operation. The original receiver had to hand over the packets to channels that were processes reading from open AAL2 type sockets. In that case, the packets were put on to the individual packet queues, and the processes woken up, who then retrieved the packets via user-level read system calls.

In the present framework, the packets had to be sent to the mapping module for table lookup and transmission over the IP network. This operation could have been done in the receiver itself. However, it is a lengthy process, right from header decompression to formation of the IP packet and transmission. It would delay the processing of the cells by the AAL2 receiver. Hence the receiver uses task queues in *tq_scheduler* mode to schedule the mapping function at the instant the scheduler next runs. The other modes of this task queue operation cannot be used, as they are not suitable for the timing constraints present here.

At the mapping module, the correct entry is located using the ATM VC and AAL2 CID. The RTP packet type is determined from the UUI field. The IP ID difference value and the UDP checksum are retrieved if present.

If the packet contains the full RTP header, then the context state is built and the state variables are initialized. For packets containing the compressed header, the changed fields are found out from the MSTC field. The link sequence number is checked to detect packet loss. If there are any 2^{nd} order changes, then the encoded differences are retrieved and the state updated. The RTP header is reconstructed from the previously stored header and current indicated changes. If there any error occurs during this process, an RTP_CONTEXT_STATE type packet is sent to the compressor for that context so that a full header packet can be sent by the ingress.

After the RTP header is reconstructed, the IP and UDP headers are taken from the mapping table and a complete IP packet is reconstructed. The new IP ID value is

calculated from the difference sent by the ingress and a new checksum is generated. These values are put into the IP header. The UDP checksum, if present, is updated with the value obtained from the AAL2 packet. In Linux, the *struct sk_buff* structure is used to move the packet among the various layers in the protocol stack. The relevant fields and pointers in this structure are updated as necessary and the packet is handed off to the IP layer.

In Linux, the entire forwarding operation of the IP layer takes place in the network *bottom half*. The scheduler regularly services the network bottom half among others, such that its execution takes place in a timely manner. All layer 2 protocol drivers invoke the network bottom half by calling the *netif_rx()* function and pass it the packet in the form of the *sk_buff* structure. This procedure is also followed by CLIP since it simplifies the operation and provides a single uniform interface to the IP layer. Hence the framework also invokes the *netif_rx()* function to hand over the packet to the IP layer, from where it is forwarded to the destination as a normal IP packet.

The AAL2_MAP table in the kernel is implemented in the form of a linked list of structures, of type *struct aal2_map_table*.

```
struct aal2_map_table {
      :
      :                  /*Flow parameters and headers–as
                         explained in table[]*/

  int hdr_flag;    /* Flags required in the kernel for
  int in_flag;                checking state */
      :
      :
  struct aal2_map_table *next; /* pointer to next node in
                                  list */

};
```

The gateway module needs to add and delete entries from the AAL2_MAP table in the kernel. Hence, two *ioctl*[3] types – ATMAAL2_MAP_ADD and ATMAAL2_MAP_DEL -

[3] An *ioctl()* is a function by which user-level processes can communicate with and manipulate the underlying device specific parameters of special files or devices.

were added to the ATM device ioctl list and their corresponding functions in the kernel were implemented.

Chapter 5

Evaluation

This chapter describes the tests carried out to evaluate the implementation and performance of the framework. The first section outlines some of the results for the real-time modifications to AAL2 in Linux. The next section explains in detail the evaluation of the RTP over AAL2 framework.

5.1 Modifications to AAL2 in Linux:

5.1.1 Test setup

The setup consisted of two end hosts interconnected by an ATM switch. The hosts were 400 MHz Pentium II based boxes running Linux kernel version 2.2.13 and atm-0.59. The ATM switch was a ForeRunner ASX-200WG switch running UNI-3.1 with OC-3 interfaces. The hosts had ENI-155 Mbps ATM cards. The two end systems were running ATM signaling daemons and the ANP daemon.

The ATM VCs used were of type CBR. Each user (channel) source transmits at a given rate (e.g. 32 kbps), sending data in chunks of payload size with a pacing determined by the bit rate desired. The CPS packet size from each user is fixed depending on the sending rate and packetization delay. The bit rates considered in the experiments are 32 kbps, 24 kbps and 16 kbps. The CPS packet size is selected such that the packetization delay remains constant (8 ms). Here it is 32 bytes and 24 bytes for 32 kbps and 24 kbps coding rates respectively.

5.1.2 Receiver Modifications

Details of the tests carried out and their results can be found in [13]. One significant benefit derived from the modifications has been the ability to achieve realistic values of 1-2 ms for the timer_CU, thus making the implementation more practical. Another

important enhancement has been the ATM driver-initiated receiver module, which has improved the performance greatly for increasing number of users. The following plots represent this fact.

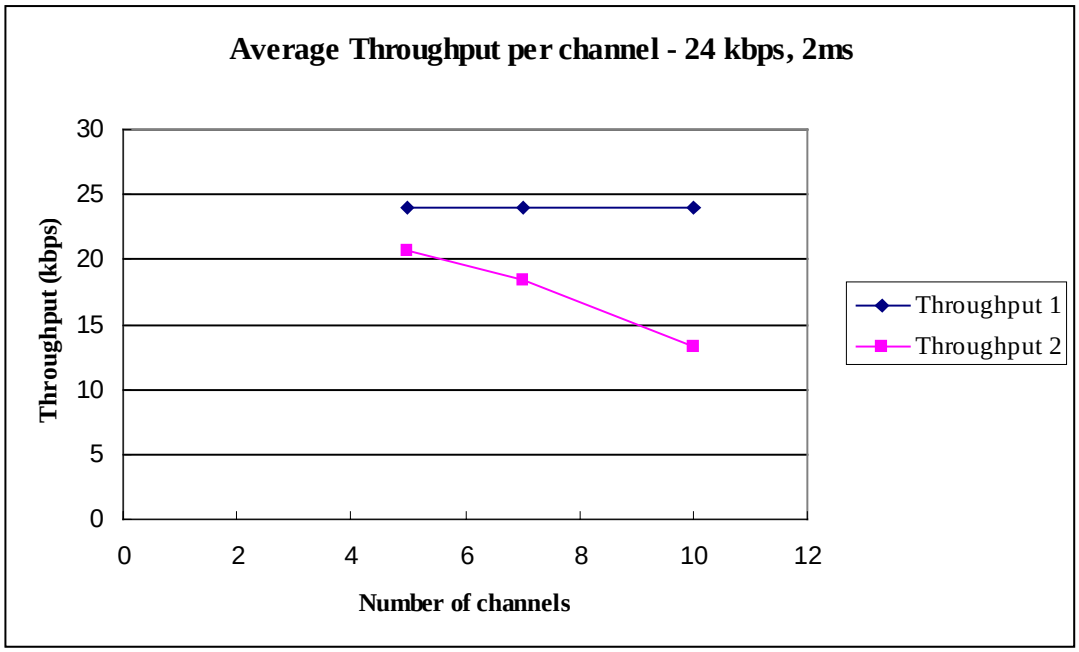


Figure 5.1 Average Throughput per channel – 24 kbps

In Figure 5.1, a comparison of the average throughput per channel at the receiver, is done for the AAL2 receiver modifications with that of the original receiver implementation, at a sending bit-rate of 24 kbps. It can be seen that as the number of users/channels increases, the average throughput per channel steadily decreases in the original receiver, whereas it remains constant at the maximum rate for the modified receiver. A similar result can be seen for a sending rate of 32 kbps in Figure 5.2.

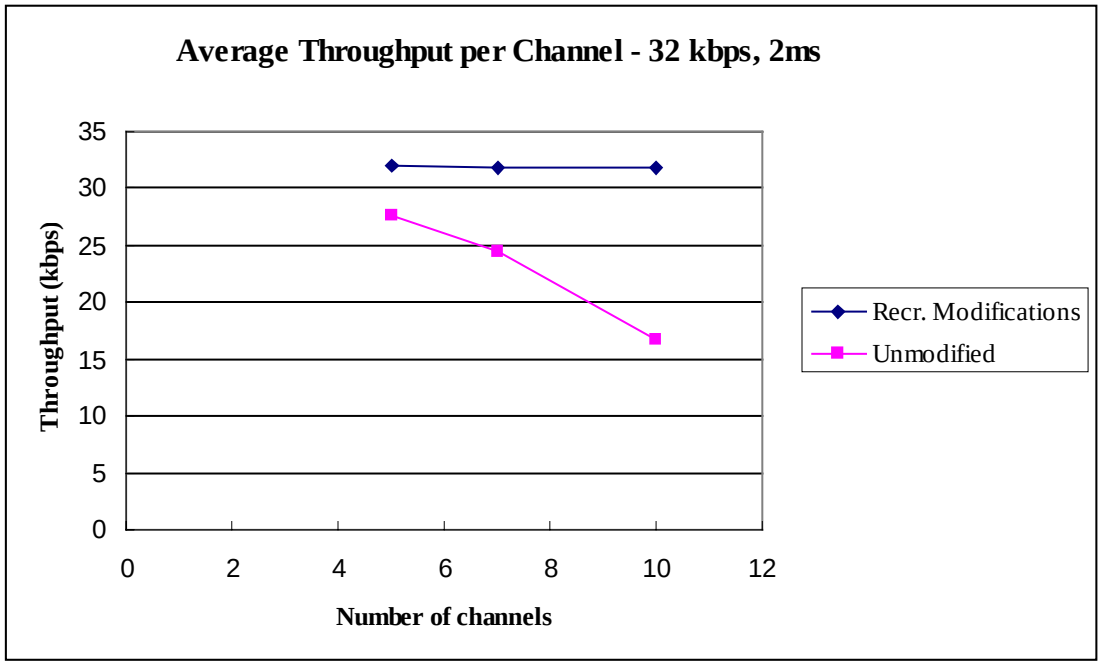


Figure 5.2 Average throughput per channel – 32 kbps

5.1.3 Bandwidth Utilization of AAL5 and AAL2

The following two plots give an indication of the difference between bandwidth required for AAL5 and AAL2. Intuitively, the presence of wasteful padding in AAL5 PDUs to fill a cell when the payload is small leads to more bandwidth being used for the same voice bit rate.

Figure 5.3 compares the bandwidth utilization between AAL2 and AAL5 for increasing number of voice flows. CBR traffic at 24 kbps with packet size of 24 bytes is used. The AAL2 timer_CU has a value of 2 ms. Initially, the bandwidth used by both the protocols are nearly the same. But as the number of channels increases, it can be seen that the bandwidth used for AAL2 is far lesser than that required for AAL5. With 5 flows simultaneously sending packets, the bandwidth required for AAL2 is roughly only half of the AAL5 value. This is due to the increased multiplexing effect for AAL2 with increased number of flows.

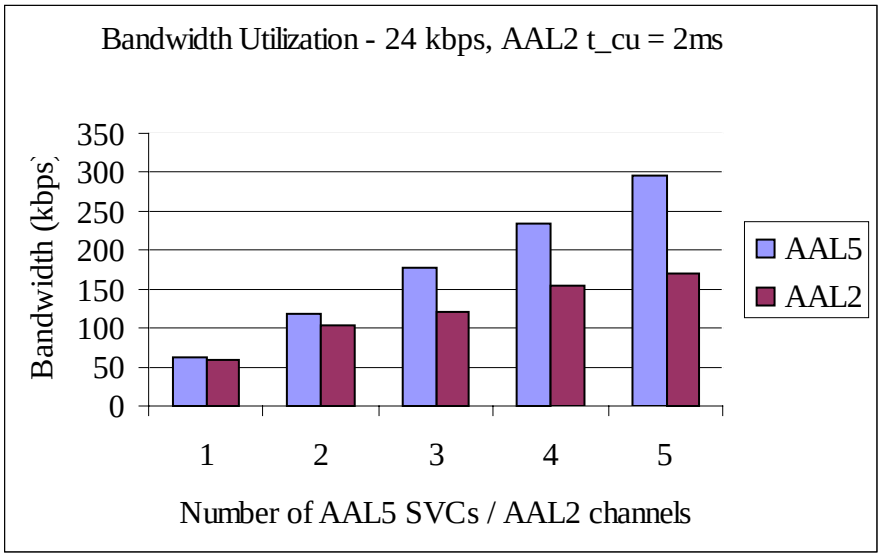


Figure 5.3 Bandwidth Utilization – AAL5 vs. AAL2 – 24 kbps

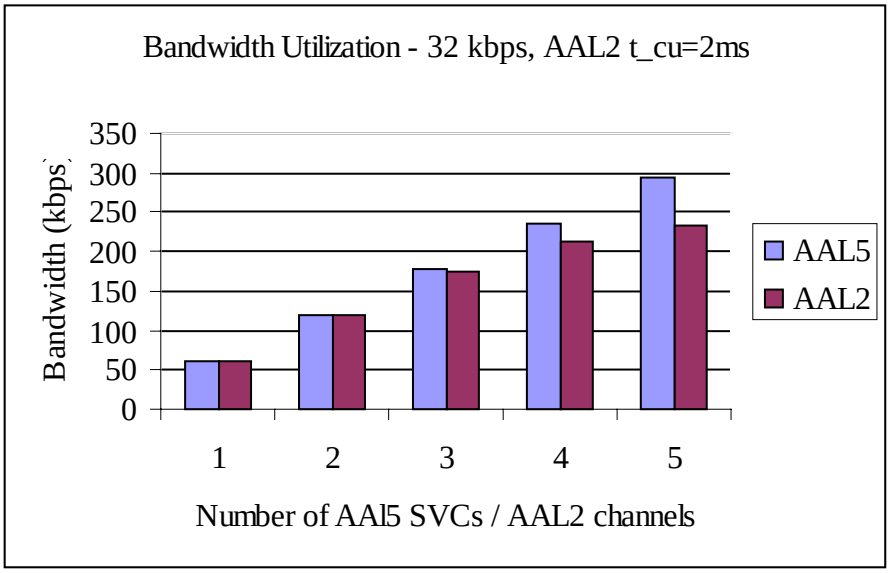


Figure 5.4 Bandwidth Utilization – AAL5 vs. AAL2 – 32 kbps

Figure 5.4 shows a similar plot for channel bit rates of 32 kbps. In this case, the bandwidth usage for both AAL5 and AAL2 is nearly same for up to 3 simultaneous flows after which AAL2 starts performing better than AAL5. Here, because of the packet size and the bit rate, the multiplexing effect is not significant in AAL2 until the number of flows increases to three. Initially, lot of padding is filled in cells.

It can also be seen from the two plots that the benefits of AAL2 are greater for lower bit rates. This is because packets get filled in cells more quickly and for increasing number of channels, multiplexing is better.

5.2 RTP over AAL2 framework

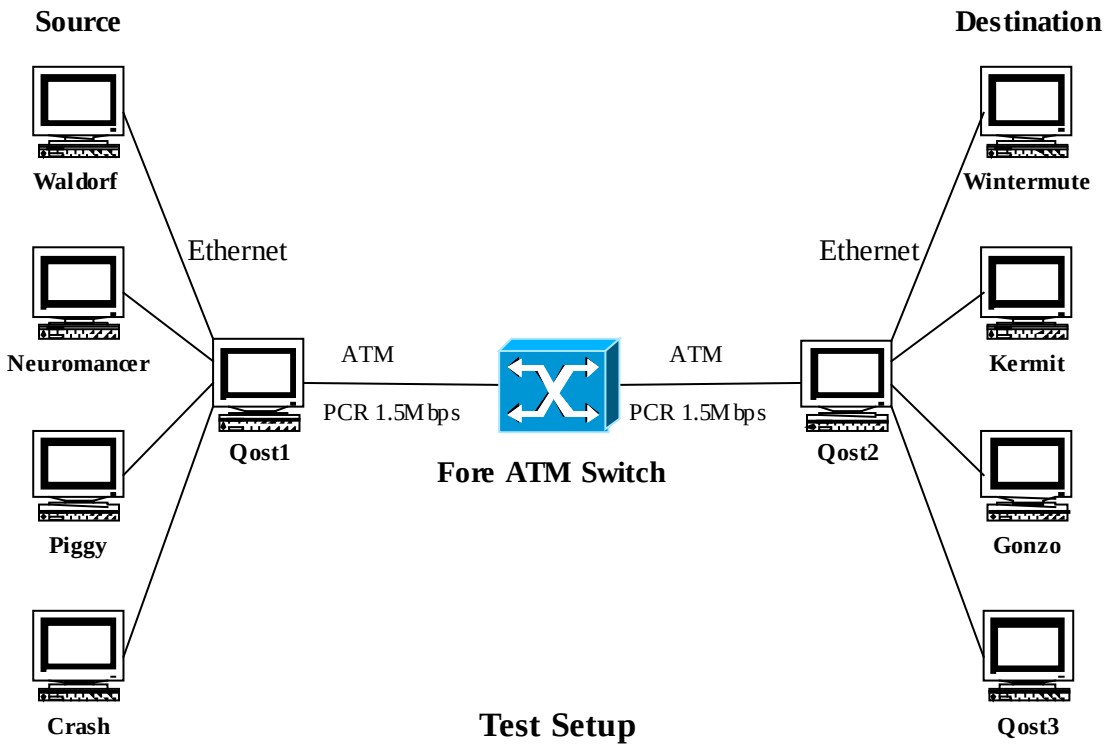


Figure 5.5 Test Environment

5.2.1 Test Environment

The setup for the evaluation is shown in Figure 5.5. Two 400 MHz Pentium II based Linux boxes running kernel version 2.2.13 and atm-0.59, serve as the two gateway nodes. They are interconnected through a ForeRunner ASX-200WG ATM switch running UNI-3.1 with OC-3 interfaces. The gateways have ENI-155 Mbps ATM cards. They also have the ATM signaling daemons and the ANP daemon running on them, along with the gateway module.

The ANPD sets up CBR VCs with a PCR of 1.5 Mbps. There are four sources and four destination hosts, each of which is set up on a 400 MHz or higher speed Linux box. The hosts are connected to the test network via 100 Mbps Ethernet links.

5.2.2 Test Application

The Robust Audio Tool or RAT, is the network audio application that is used to generate audio traffic between pairs of end-hosts. RAT has support for a number of codecs. However, RAT doesn't support the standard low-bit rate codecs due to licensing problems. The only low-bit rate codec in RAT whose payloads fall within the CPS packet limits of AAL2 is the 5.8 kb/s,10 pole LPC codec. The amount of audio data, in milliseconds, sent in each packet can be varied. For the LPC codec, audio samples of 20 ms and 40 ms duration generate packets of sizes 26 and 40 bytes respectively, including the RTP header. Since the values fall within the AAL2 CPS packet limit, these two packet sizes are used in the evaluation.

RAT is used to send audio files from the transmitter to the receiver. Two different speech files, each of approximately 5 minutes duration were used between each transmitter-receiver pair. These two files have different speech and silence characteristics and hence generate varying output streams. Silence suppression is enabled in the application.

5.2.3 Bandwidth Utilization

Efficient bandwidth utilization is the main focus in this work. Hence it is one of the most important metrics to evaluate the performance of the framework.

The greater efficiency of AAL2 in transporting small size packets, as compared to AAL5, has been demonstrated [13] and in Section 5.1.3. Since AAL5 is the other adaptation layer being considered for voice transport over ATM using RTP header compression, ideally a comparison between the two AALs should have been done, in the presence of header compression. However, in the Linux-ATM stack, AAL5 SAR is done in hardware

by the ATM network interface card. The AAL5 trailer is not accessible in the kernel to convey any header compression information between the two ends. Therefore, the RTP header compression scheme could not be tested with AAL5. However, from the comparative analysis between AAL2 and AAL5 given in Section 4.1.6.1 and from the above results, it can be seen that AAL2 does have an inherent advantage over AAL5 due to its multiplexing capabilities. Moreover, the analysis also shows that RTP header compression is totally ineffective for AAL5.

Still, in order to get an idea of the bandwidth efficiency provided by the framework when actual voice traffic is sent over it, its performance is compared with that of the normal IP over ATM method (CLIP).

As shown in the figure above, there are four transmitter-receiver pairs. Tests were conducted with one, two and all four pairs simultaneously active. Even though, a larger number of flows can be sent through the framework (up to 15 channels could be opened simultaneously on a single host [13] without causing degradation in performance), the present setup uses only up to four flows. This is due to incompatibilities between RAT and the Linux boxes and sound cards available here. Only a limited number of capable end hosts could be set up successfully. Moreover, multiple instances of RAT cannot be started on the same machine without causing degradation in its performance.

The bandwidth is measured at the ATM switch using SNMP. The MIB *ChannelCells* is measured once every 100 ms, and the average of 100 such readings is plotted as a sample point. This is repeated over the entire duration of the test.

5.2.3.1 One voice flow

A single voice flow was sent for approximately 5 minutes. The LPC codec was used with 20 ms sampling interval. A total of roughly 11,000 packets were transmitted.

The bandwidth curve is not constant but keeps continuously varying depending upon the voice characteristics at the transmitter and hence the packet generation rates.

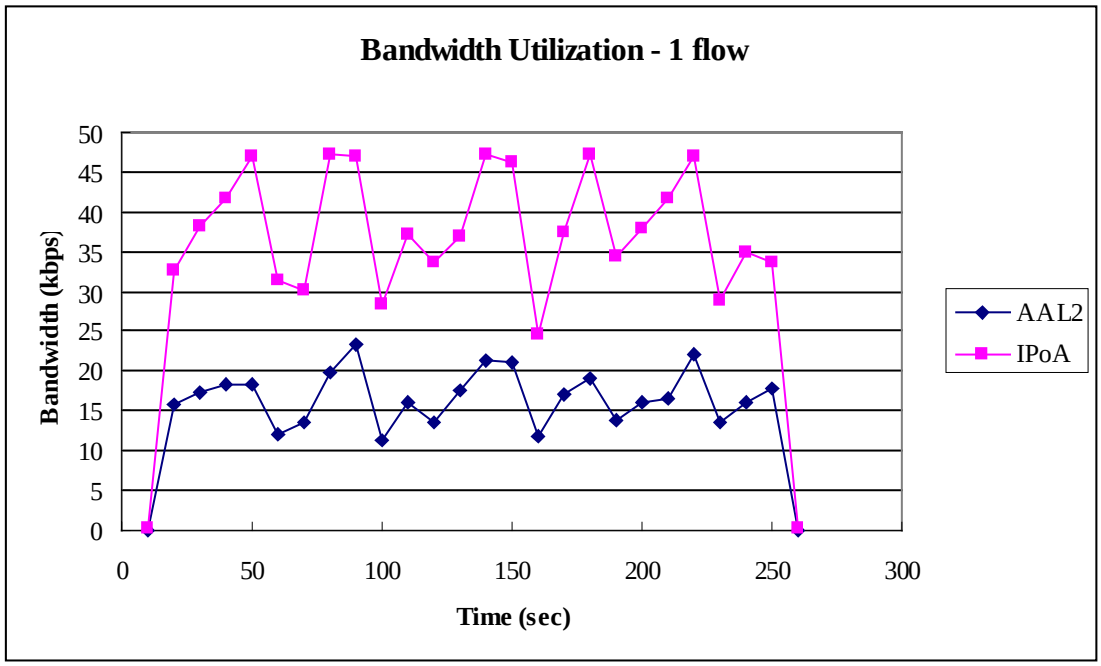


Figure 5.6 Comparison of Bandwidth Required – AAL2 vs. IPoA – 1 flow

It can be seen from the plot of Figure 5.6 that the bandwidth curves for AAL2 and IPoA follow the same varying pattern closely, but the bandwidth used for the IPoA method is more than twice that for the AAL2 framework with header compression. This is mainly due to the overhead of the IP and UDP headers, which are present for every packet transmitted. The absence of these headers in the AAL2 case, along with the compressed RTP header, causes most of the gain. It should be noted that with only one flow, the cell does not often get filled within the timer_CU limit and a large number of cells are padded.

5.2.3.2 Two voice flows

Two tests were run, the first with both flows at 20 ms sampling intervals, and the second with one flow each at 20 ms and 40 ms sample periods. The results of the first test, plotted in Figure 5.7, show the same trend as in the 1-flow case. It can also be seen that the bandwidth used up for the AAL2 framework is greater than for the 1-flow case, which

is expected, but it is less than double of it. This indicates that there is multiplexing taking place between packets of the two flows, leading to a slight reduction in bandwidth consumption.

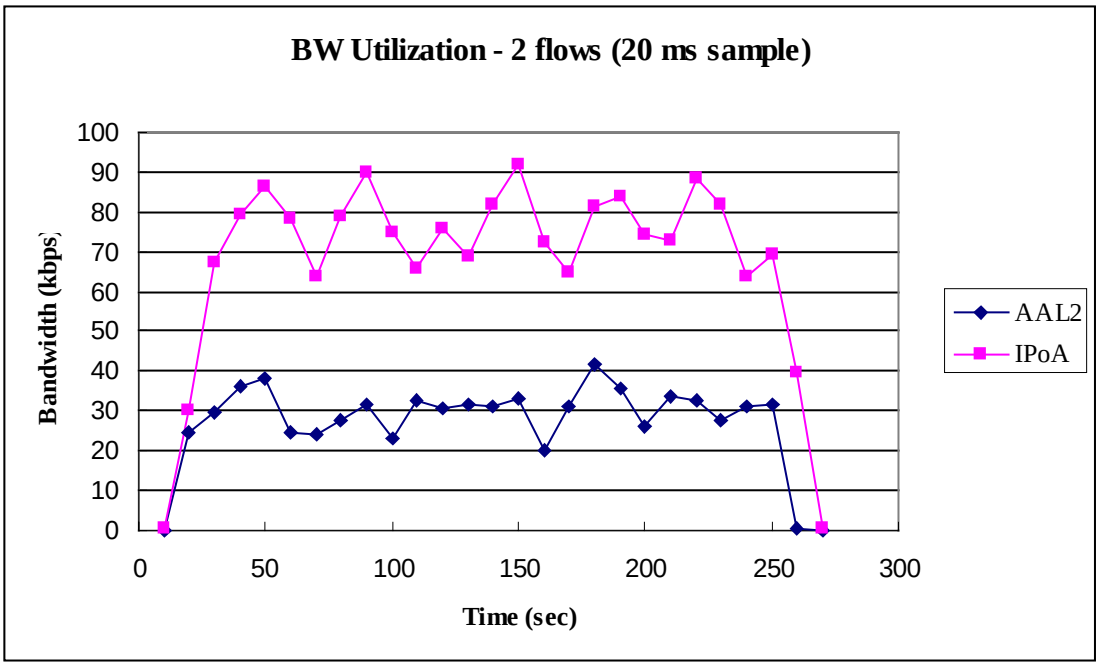


Figure 5.7 Comparison of Bandwidth Required – AAL2 vs. IPoA – 2 flows

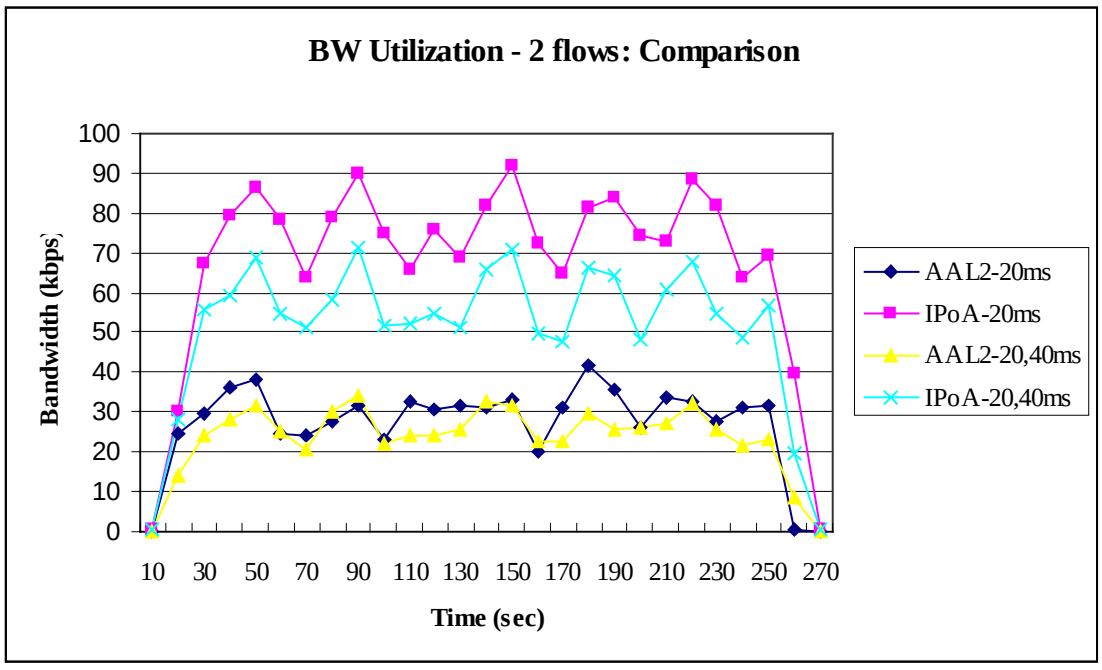


Figure 5.8 Comparison: AAL2 vs. IPoA – 2 flows of different sampling intervals

Figure 5.8 has plots of both the 2-flow cases in the same graph. The interesting point to be noted about the AAL2 plots is that the bandwidth used is less in the case where the flows have different sampling intervals. The number of packets generated for the 40 ms frames is considerably less (around 5000) due to the increased sample period. Because of their larger size, the packets are fitted into cells more efficiently and as a consequence, the bandwidth utilization is better.

Similarly, from the IPoA plots, it is seen that the bandwidth consumption for the two different flows, though greater than for AAL2, is lesser compared to the two 20 ms flows. Again, this is mainly because of the reduction in the number of packets for the 40 ms flow, which in this case means lesser protocol header overhead and correspondingly smaller cell rates.

5.2.3.3 Four voice flows

Two flows of 20 ms frame size and two flows of 40 ms frame size were sent simultaneously. A combination of two different voice files was used, one file with two host pairs and the other file with the remaining two.

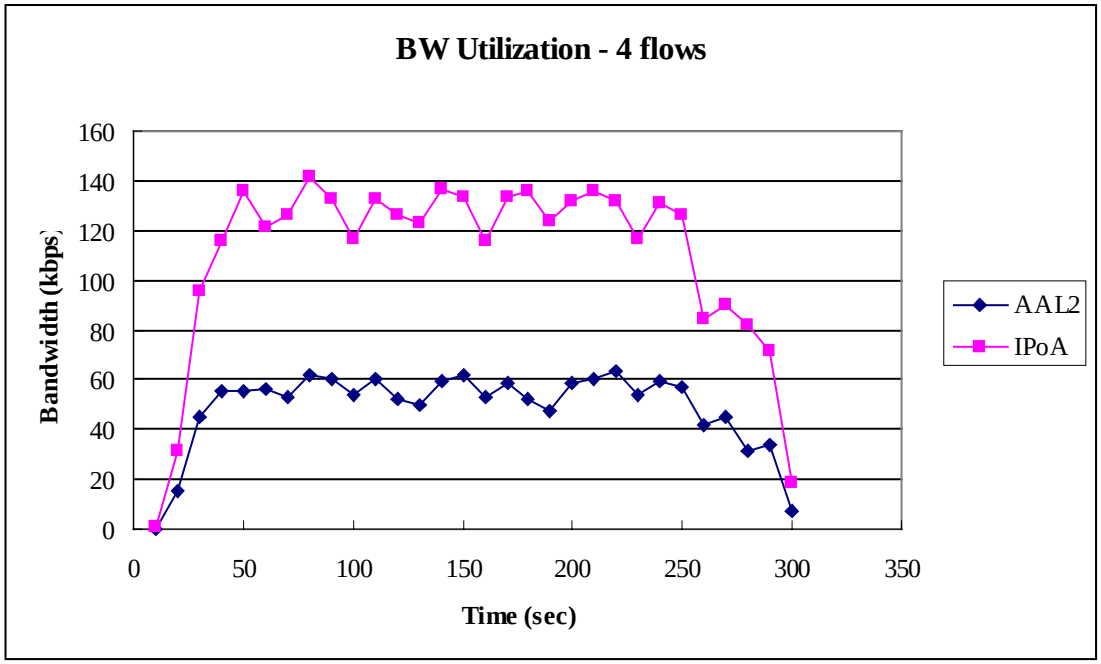


Figure 5.9 AAL2 vs. IPoA – 4 flows of different sampling intervals

The results follow the expected pattern where the AAL2 framework performs better than the IP over ATM method. This trend can be expected to continue for larger number of voice flows where the multiplexing at the AAL2 level will be greater. It should also be noted that RTP header compression would have a big effect on the bandwidth for increasing number of flows. For a small number of flows, compression might not benefit too much because there may not be enough packets arriving to efficiently fill cells and the space created by compression will get filled up with useless padding.

5.2.4 Determination of jitter

From the tests conducted in the previous section, it is seen that bandwidth is utilized much more efficiently using the RTP over AAL2 framework. But the packets have to undergo extra processing in the two gateways for compression and decompression and header removal and re-assembly. Another source of delay could be the AAL2 transmitter and receiver state machines. Packets have to wait for the cell to get filled up or the timer_CU to expire before they can be transmitted. Similarly at the receiver, the individual AAL2 CPS packets must be de-multiplexed from arriving cells.

Most audio applications have a playout buffer at the receiver where arriving packets are collected to account for delayed or re-ordered packets. The packets in the buffer are converted back into audio samples and played out at the correct rate. Even if packets are delayed, if they are within a certain limit, the receiver can take care of it. For voice packets, even more important than delay is the variation in the delay or the jitter.

The following tests attempt to detect if the implemented framework introduces any jitter and to measure the variation in the delay. The transmitter generates RTP packets at intervals equal to the sampling period with some additional packetization delay. However, there will be variations in this interval caused by process scheduling problems, workstation load effects or due to the characteristics of the audio sample files. In any

case, these packets, if they do not suffer delay jitter effects in the network, should arrive at the receiver at the same intervals.

Hence, the inter-packet transmission times are measured at the source, and the corresponding inter-packet arrival times at the receiver are also measured. *tcpdump* output trace is used to measure these timestamps at both sides. At the transmitter, the timestamps indicate the instants when packets are written to the UDP socket after formation. At the receiver, the timestamps represent the instants just before packets are read by the application.

The difference between two consecutive packet events (transmissions/arrivals) is calculated over the entire duration of the flows, which again are around 5 minutes each, with flows of both 20 ms and 40 ms sample periods. The LPC codec is used again.

The individual jitter values are then calculated as the difference between the receiver intervals and the corresponding transmitter, for all packets. The average and standard deviation of these jitter values for the entire data range is then calculated. Since the packets can arrive at the receiver both earlier as well as later than the intervals at the transmitter, the jitter values can be both positive and negative. This process is carried out for one and four voice flows. The 4-flow case is compared with a 4-flow IPoA case. The values are given in Table 5.1.

Number of flows	**Average (μsec)**	**Standard Deviation (μsec)**
1 flow	0.1582	4695.65
4 flows	-0.099	3375.28
4 flows (IPoA)	-0.0056	169.804

Table 5.1 Jitter between transmitter and receiver

It can be seen that for all cases, the average value of the jitter is nearly zero, which is the desired result. The framework does introduce some variation in delay, as can be

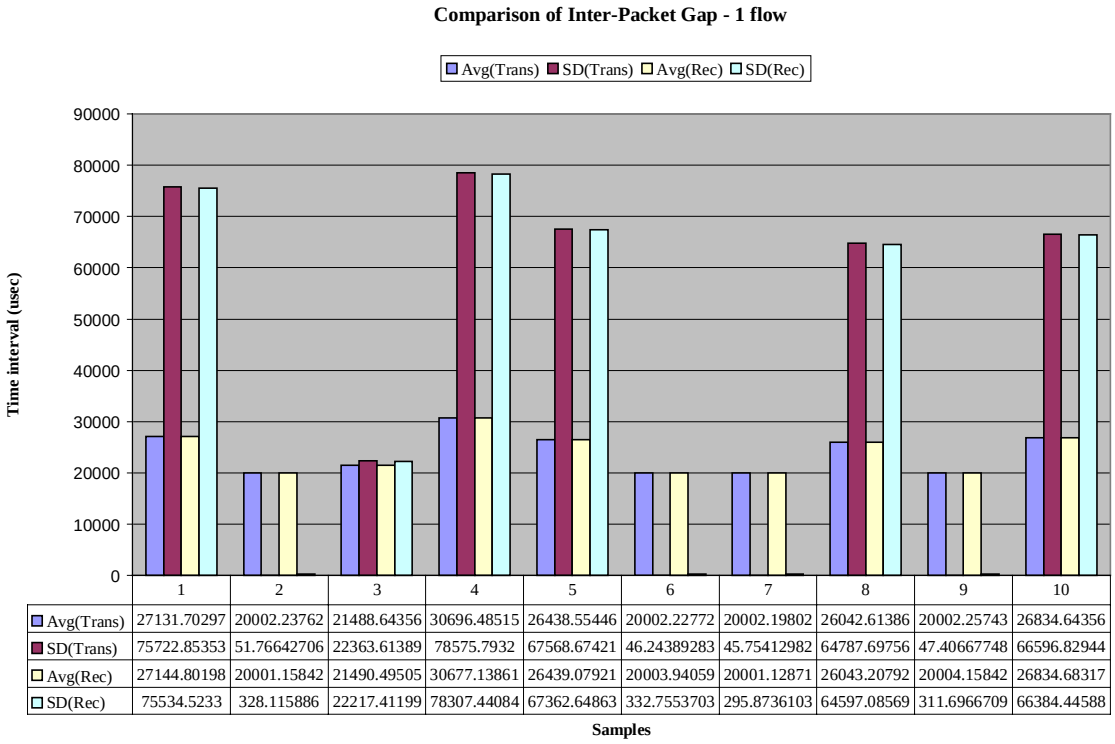
concluded from the standard deviation values. But the values are still very low (~3-5 ms) and are well within acceptable limits. The IPoA approach, which doesn't introduce any delay, has a much lower standard deviation as compared to the AAL2 case.

It was observed during the tests that there are a large number of timer expiries occurring throughout the test duration. This is mainly due to the low bit rates and the small number of flows being used. The packet generation at the transmitter is also not regular because of the voice file characteristics and also silence suppression. Packets at random have to wait until the timer expires to be sent. This causes the variation observed in the interval differences between the two ends. For larger number of flows, this variation is expected to decrease due to reduced effect of the timer and steady arrival of packets.

5.2.5 Comparison of Inter-packet times – individual samples

The previous section shows the jitter values between the two ends for the entire duration of the flows. They serve to show that the network maintains the timing characteristics of the packets throughout the flow duration. However, there could be subtle variations occurring at different times during the tests. Hence, ten sample sets of 100 consecutive packets each are selected randomly from the entire range of readings and the individual values are compared between the source and the receiver. The standard deviation and the average values within these sample sets are plotted, in microseconds (usec).

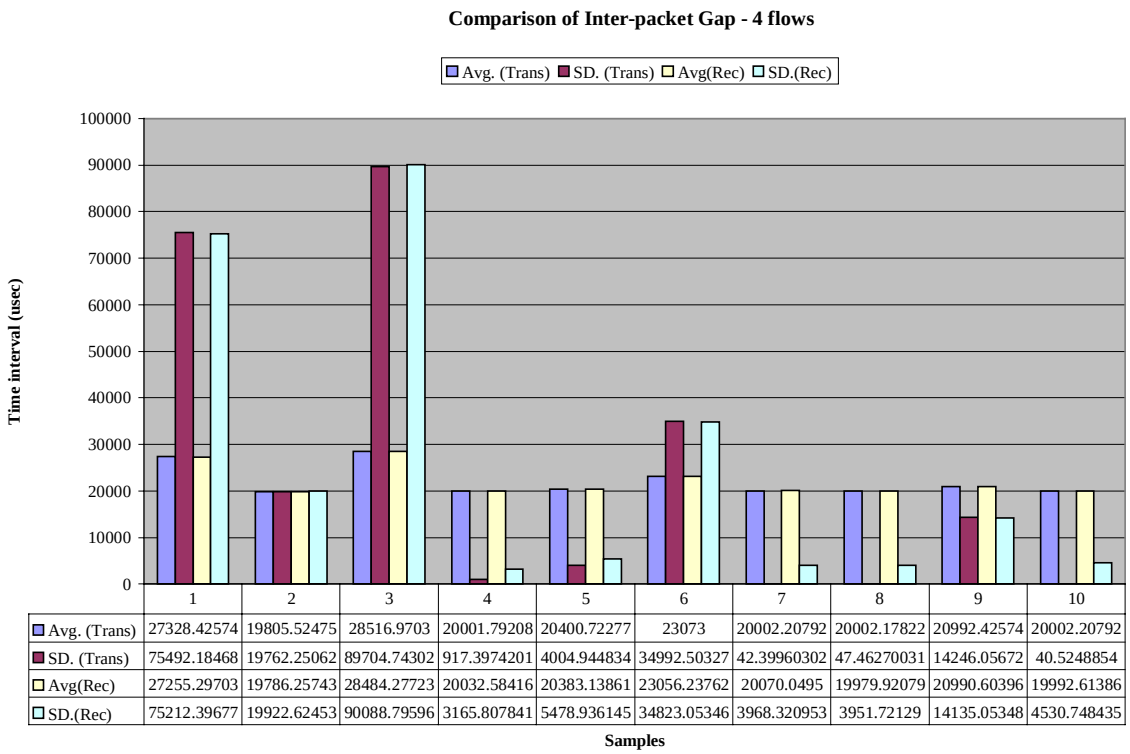
5.2.5.1 One voice flow



	1	2	3	4	5	6	7	8	9	10
Avg(Trans)	27131.70297	20002.23762	21488.64356	30696.48515	26438.55446	20002.22772	20002.19802	26042.61386	20002.25743	26834.64356
SD(Trans)	75722.85353	51.76642706	22363.61389	78575.7932	67568.67421	46.24389283	45.75412982	64787.69756	47.40667748	66596.82944
Avg(Rec)	27144.80198	20001.15842	21490.49505	30677.13861	26439.07921	20003.94059	20001.12871	26043.20792	20004.15842	26834.68317
SD(Rec)	75534.5233	328.115886	22217.41199	78307.44084	67362.64863	332.7553703	295.8736103	64597.08569	311.6966709	66384.44588

Figure 5.10 Inter-packet transmission/arrival times – Individual samples – 1 flow

For the 1-flow case plotted in Figure 5.10, it can again be said that even for the individual packets, the framework maintains the timing constraints. The transmitter itself has a lot of variations while sending packets but as can be seen, the standard deviations for the various sample sets are almost the same at both ends.

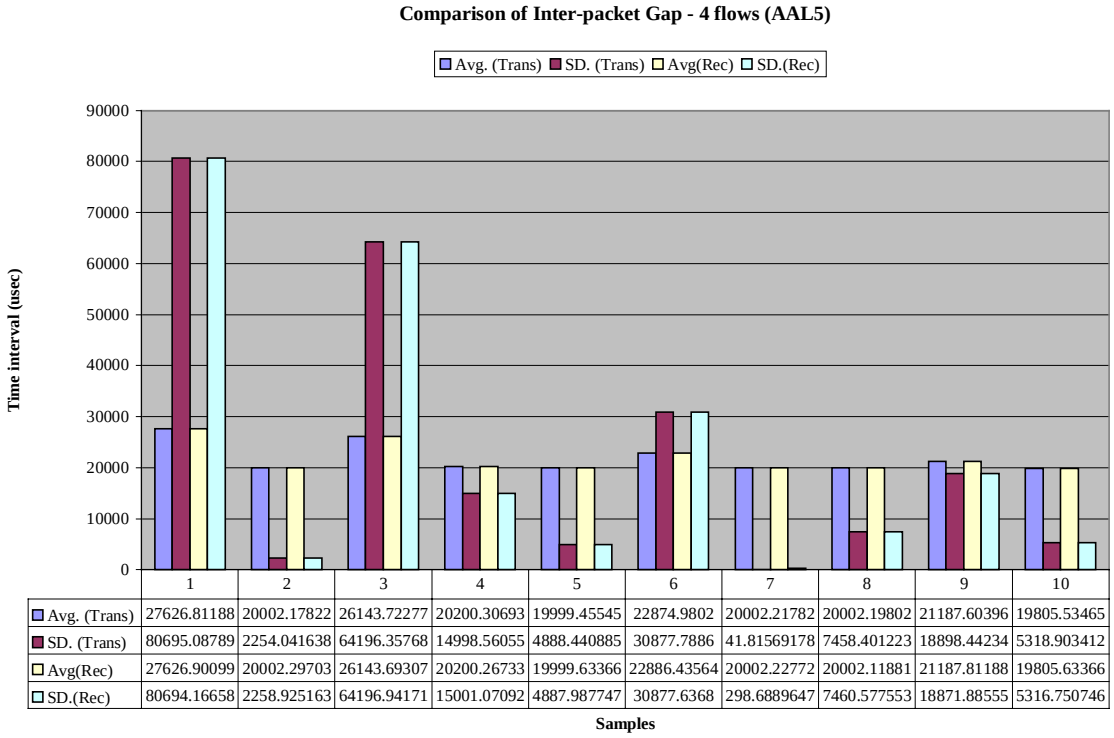


	1	2	3	4	5	6	7	8	9	10
Avg. (Trans)	27328.42574	19805.52475	28516.9703	20001.79208	20400.72277	23073	20002.20792	20002.17822	20992.42574	20002.20792
SD. (Trans)	75492.18468	19762.25062	89704.74302	917.3974201	4004.944834	34992.50327	42.39960302	47.46270031	14246.05672	40.5248854
Avg(Rec)	27255.29703	19786.25743	28484.27723	20032.58416	20383.13861	23056.23762	20070.0495	19979.92079	20990.60396	19992.61386
SD.(Rec)	75212.39677	19922.62453	90088.79596	3165.807841	5478.936145	34823.05346	3968.320953	3951.72129	14135.05348	4530.748435

Figure 5.11 Inter-packet transmission/arrival times – Individual samples – 4 flows

5.2.5.2 Four voice flows

Figure 5.11 gives a similar comparison for 4 flows. Again, in most sample cases, the averages and standard deviations are nearly same for both ends. In some cases, there is a very small deviation value at the transmitter but the corresponding value at the receiver is slightly higher, around 3-4 ms. This is similar to the observations in the previous section. This could be explained as the probable effect of the AAL2 timer, which is set at 2ms. One way where this effect could appear is if a packet was partly filled into the last part of a cell and the remaining bytes of the packet had to wait for the entire 2ms period until the timer went off, to be sent. This value could decrease for a larger number of simultaneous flows or for higher bit rate flows, where the number of packets flowing through the network would be much more. Still, a deviation of 3-4 ms is not a high value and is acceptable for audio applications to handle. For example, RAT has an internal minimum jitter limit of 80-100 ms within which it can provide good voice quality.



	1	2	3	4	5	6	7	8	9	10
Avg. (Trans)	27626.81188	20002.17822	26143.72277	20200.30693	19999.45545	22874.9802	20002.21782	20002.19802	21187.60396	19805.53465
SD. (Trans)	80695.08789	2254.041638	64196.35768	14998.56055	4888.440885	30877.7886	41.81569178	7458.401223	18898.44234	5318.903412
Avg(Rec)	27626.90099	20002.29703	26143.69307	20200.26733	19999.63366	22886.43564	20002.22772	20002.11881	21187.81188	19805.63366
SD.(Rec)	80694.16658	2258.925163	64196.94171	15001.07092	4887.987747	30877.6368	298.6889647	7460.577553	18871.88555	5316.750746

Figure 5.12 Inter-packet transmission/arrival times for AAL5 – 4 flows

Figure 5.12 plots the average and standard deviation for 10 sample sets of inter-packet events when sent over AAL5 (CLIP). As can be seen, there is almost no difference in the values at the transmitter and receiver. The standard deviation values are also almost identical for all the ten sets, unlike the AAL2 case, where for some samples, a small difference was observed. This is mainly because there are no delay sources in the network for AAL5 apart from scheduling variations and queuing, if any.

Chapter 6

Summary

A framework for transporting RTP voice flows over an ATM network using the ATM Adaptation Layer 2 was designed and a preliminary implementation of the same was done in Linux. Real-time modifications were made to the AAL2 support in the Linux kernel to achieve realistic timing capabilities and performance levels.

Some of the issues involved in transporting RTP encapsulated VoIP packets over ATM were discussed and the current framework was compared to other similar efforts. A simple signaling module was implemented to be compatible with the existing AAL2 signaling architecture, and was used to establish AAL2 channels over ATM SVCs for the voice flows. Some end-to-end interoperability issues were outlined.

A kernel based mapping of RTP flows to AAL2 channels was developed and packet forwarding was done directly between the IP layer and AAL2, thus avoiding a lot of application-level processing and protocol overhead. Transfer of IP and UDP protocol headers over the ATM network was avoided, reducing bandwidth requirements. A RTP header compression technique was implemented to obtain further bandwidth reduction.

Voice traffic was sent over the framework to validate the functioning of the implementation and study the performance based on some simple metrics. Some of the advantages of the framework in carrying low bit rate RTP voice traffic were seen.

6.1 Conclusions

It can be seen from the obtained results that the framework does improve the efficiency with which voice streams can be transported over ATM. A network audio application, RAT was used to generate RTP based voice flows of reasonable length. Low bit rate coding was used with different sampling intervals to generate packets of different sizes.

The real-time modifications create a marked improvement in performance of the AAL2 support. The fine-grained timer values make it practically feasible to use the implementation.

The bandwidth utilization is significantly better using the RTP over AAL2 framework than for the normal IP over ATM method. This is due to reduction of header overheads and the multiplexing benefits of AAL2.

The implemented RTP header compression can be used for AAL5 too. However, it can be seen from the comparative evaluation tables that compression is not effective for AAL5 with most of the default codec parameters. Whereas for AAL2, the reduced packet sizes complement the bandwidth efficiency obtained due to multiplexing.

For smaller number of flows, the advantages of compression may not be too apparent because of the smaller number of packets and reduced multiplexing. But as the number of flows increases, the cells get filled more efficiently, there is less padding and the bandwidth gets utilized better. This is in accordance with the properties of AAL2.

It is also seen that the implemented framework does not make the overall operation complex, and the timing characteristics of the audio packets are faithfully maintained. In most of the cases seen, the framework does not introduce any additional jitter. Occasionally a small variation is observed, mainly due to the effects of the timer in the AAL2 transmitter. However, it is a very insignificant value as compared to the network jitter limits of audio applications. This has been confirmed by subjective tests to determine the audio playout quality.

6.2 Future work

This framework is restricted in its applicability due to the limited range of AAL2 CPS packet sizes. It cannot be used with the high bit rate codecs in their default modes as the

resulting RTP payload will exceed the maximum CPS limits. However, it is possible to use such codecs by using non-default values of packetization intervals such that the packet sizes decrease. But doing so may decrease the protocol efficiency. Hence, studies need to be made to determine the usage of these codecs.

Even for the low bit rate codecs, it is possible to increase efficiency by choosing appropriate sampling intervals. But using different values may compromise the delay performance for those codecs. Careful consideration of packet sizes is required to maintain delay performance while increasing efficiency.

The effect of packet losses has not been considered here. It is possible for packet losses to occur while traversing low speed and congested links in the Internet. Also, losses may occur on the link containing the framework. In the case of AAL2, a loss of one cell may affect more than one packet due to multiplexing. The effect of such losses on the compression states and possible voice quality degradation should be investigated.

The framework has been tested with a small number of flows due to limitations of the test environment. But it will definitely support a larger number of simultaneous sessions and the performance should correspondingly improve. It should be tested further to determine practical limits.

RTP itself supports multiplexing of packets from different sessions into one RTP payload to make efficient use of bandwidth and reduce protocol overhead. The current framework cannot effectively deal with such scenarios. It should however be interesting to see how RTP multiplexing can be mapped on to the AAL2 based framework and if there are any advantages to doing so.

Bibliography

[1] IETF RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*, January 1996, ftp://ftp.isi.edu/in-notes/rfc1889.txt

[2] IETF RFC 1890, *RTP Profile for Audio and Video Conferences with Minimal Control*, January 1996, ftp://ftp.isi.edu/in-notes/rfc1890.txt

[3] ITU-T Recommendation H.323, *Packet based multimedia communications systems*

[4] ATM Forum Technical Committee, AF-SAA-0124.000, *Gateway for H.323 Media Transport Over ATM*, July 1999

[5] de Prycker M., *Asynchronous Transfer Mode - Solution for Broadband ISDN*, Prentice Hall International, 1995, pp. 105-159

[6] The ATM Forum Technical Committee, *User-Network Interface (UNI) Specification Version 3.1*, Sept 1994

[7] Chen, T. M. and Stephen S. L., *ATM Switching Systems*, Artech House, Incorporated, 1995, Chapters 5-10, pp. 81-233

[8] Almesberger, W., *Linux ATM internal signaling protocol, Draft Version 0.2*, Laboratorie de Re'seaux de Communication, Nov. 5, 1996

[9] ITU-T Recommendation I.363.2, *B-ISDN ATM Adaptation Layer Type 2 Specification*, Feb 1997

[10] Moondhra, V., *Implementation and Performance Analysis of ATM Adaptation Layer Type 2*, Master's thesis, University of Kansas, Lawrence, Kansas, January 1998

[11] IETF RFC 2508, *Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*, ftp://ftp.isi.edu/in-notes/rfc2508.txt, February 1999, Casner, Jacobson

[12] Balaji S., Menon R., *UTIME - Micro-Second Resolution Timers for Linux*. http://hegel.ittc.ukans.edu/projects/utime.

[13] Dhananjaya Rao, Dr. Joseph B. Evans, *Real-Time Modifications to AAL2 in Linux and Performance Evaluation*, Technical Report, Information and Telecommunications Technology Center, University of Kansas, July 2000

[14] Aarti Iyengar, Dhananjaya Rao, Dr. Joseph B. Evans, *Implementation of ATM Adaptation Layer 2*, Technical Report ITTC-FY2000-TR-15662-01, Information and Telecommunications Technology Center, University of Kansas, July 1999

[15] ATM Forum Technical Committee, AF-VTOA-0113.000, *ATM Trunking using AAL2 for Narrowband Services*, February 1999

[16] Raghu Vatte, D.W. Petr, Performance Comparison between AAL1, AAL2 and AAL5, Technical Report ITTC-FY98-TR-13110-03, Information and Telecommunications Technology Center, University of Kansas, March 1998

[17] IETF Internet Draft, *Multiplexing Scheme for RTP Flows between Access Routers*, October 1999, El-Khatib, Luo, Bochmann, Feng

[18] Sheth, S., *Implementation of a Congestion Control Scheme for Active Narrowband ATM Networks*, Master's thesis, University of Kansas, Lawrence, Kansas, January 1999

[19] Beck M., Bohme H. et al, *Linux Kernel Internals*, 2nd Edition, Addison-Wesley, 1998

[20] Comer, D. E., *Internetworking with TCP/IP Volume I*, 3rd Edition, Prentice Hall International, 1995

[21] IETF RFC 1577, *Classical IP and ARP over ATM*, January, 1994

[22] ATM Forum Technical Committee, AF-LANE-0112.000, *LAN Emulation over ATM Version 2*, February 1999