Simulation Study of an ADSL Network Architecture: TCP/IP Performance Characterization and Improvements using ACK Regulation and Scheduling Mechanisms

Kaustubh S. Phanse

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. Luiz A. DaSilva, Committee Chair
Dr. Scott F. Midkiff, Committee Member
Dr. Ira Jacobs, Committee Member

Date: October 27, 2000.
Alexandria, Virginia

Keywords: ADSL, TCP/IP, PPP, ATM, ACK regulation, differentiated service, simulation

Simulation Study of an ADSL Network Architecture: TCP/IP Performance Characterization and Improvements using ACK Regulation and Scheduling Mechanisms

Kaustubh S. Phanse

(Abstract)

Asymmetric digital subscriber line (ADSL) is a broadband access technology capable of delivering large bandwidth over existing copper telephone line infrastructure. This research aims at characterizing and analyzing TCP/IP performance in presence of a new protocol stack (TCP/IP over PPP and ATM) being promoted for one of the ADSL network architectures.

Using extensive simulations, we verify the adverse effects of asymmetric links on the performance of TCP and additional throughput degradation caused by the overhead at the AAL5-ATM layers. This study involves unidirectional as well as bi-directional data transfer using different traffic mixes including bursty and non-bursty types of traffic. Bi-directional data transfer over asymmetric links results in *ACK compression* wherein TCP acknowledgements (ACKs) get bunched together behind larger data packets, further exacerbating the effect of asymmetry on TCP performance. By implementing the simulation model for PPP encapsulation over AAL5, we characterize its effect in terms of throughput degradation and excessive delay.

We quantify the improvement in the throughput obtained by delaying the TCP ACKs and by TCP/IP header compression. These techniques being effective for unidirectional traffic over asymmetric links, however, do not prove as effective when ATM enters the scenario or in presence of bi-directional data transfer. Further, we implemented a simulation model of the *Smart ACK Dropper (SAD)*, a technique to regulate the flow of TCP ACKs. Considerable improvement in performance especially in the presence of unidirectional data transfer is achieved using the SAD technique. Although the improvement is to a lesser extent in the presence of bi-directional data traffic, SAD helps the network in quickly recovering from the impact of ACK compression.

We also propose and implement certain customized queuing/scheduling and policing mechanisms to enable differentiated servicing of TCP ACKs and data packets, and mitigate the effect of ACK compression. While providing considerable TCP performance improvement in presence of SAD, custom queuing also allows fair sharing of bandwidth between TCP flows, unlike priority queuing, which starves the low priority flow. The committed access rate (CAR) policing scheme provides considerable performance improvement when used with SAD, and is especially useful when TCP ACKs compete with bursty data traffic over the slower upstream.

Acknowledgements

I would like to take this opportunity to thank my advisor Dr.Luiz DaSilva for his expert guidance throughout this research project as well as for being a very patient and supportive advisor. I not only found an excellent advisor and teacher in him, but also a very good friend.

I would also like to thank my committee members, Dr. Scott Midkiff and Dr. Ira Jacobs for their time and co-operation in reviewing this thesis.

I wish to express my gratitude to ECI Telecom Inc. for their financial support to this research project. My thanks to Dr. Kalyan Kidambi for his valuable guidance, and also for offering me the opportunity of working on-site at ECI Telecom Inc. as a research intern in the summer.

A special thanks to Nattavut Smavatkul for his guidance, especially in the early stages of my confrontation with the OPNET Modeler™. His unbelievable patience and ever willingness to help is remarkable.

Also, I would like to appreciate the Technical Support crew at OPNET Technologies Inc. for helping me with OPNET related problems along the course of this project.

For using the computing facilities for this research, I would like to thank the Virginia Tech Information Systems Center.

Also thanks to everybody at the Alexandria Research Institute for their help and co-operation.

Last and in no way the least, heartfelt thanks to so many others, friends and family members, whose names do not appear here!

CONTENTS

List of Figures

List of Tables

CHAPTER 1. INTRODUCTION

High speed Internet access in small office and home office environment (SOHO) has always had to bear with the *last mile problem*, i.e. the bandwidth bottleneck arising due to the use of traditional copper telephone wires in the local loop. In recent times, several broadband access technologies have been proposed. These include both wireline technologies such as the digital subscriber line (DSL) and hybrid fiber coax (HFC), and wireless technologies such as broadcast satellite and local multipoint distribution service (LMDS).

This thesis work examines one of the proposed network architectures for the asymmetric digital subscriber line (ADSL), and evaluates the various higher layer protocol performance issues. Specifically, we focus on characterization and evaluation of TCP/IP performance in the ADSL access network.

1.1 Overview and Motivation

Asynchronous digital subscriber line (ADSL) is a high-speed data transmission technology that promises large bandwidth increases over the existing copper cable infrastructure. ADSL in the access portion of the network, along with asynchronous transfer mode (ATM) in the backbone network provides a new platform for delivering broadband services to small offices/businesses and homes (SOHO). ADSL can support a wide variety of high bandwidth applications, such as high-speed Internet access, telecommuting, virtual private networking and streaming multimedia content [ADS98, Kwo99], many of which use the transmission control protocol (TCP) because of its ability to provide reliable communication. The robustness of TCP/IP in multiple networking environments is the primary reason for its large-scale deployment.

However, emerging access technologies such as wireless and ADSL pose new challenges to TCP performance. It is important that TCP performance considerations be taken into account in the design and development of such access technologies.

With Internet access being the primary market driver for ADSL, understanding the performance of TCP/IP and ATM over ADSL is crucial. Further, the ADSL access network standard calls for the use of point-to-point protocol (PPP) [Sim94] over ATM in the ADSL access network. The idea is to take advantage of various PPP functions such as authentication, network layer auto-address configuration, multiple network layer protocol support and encryption [Kwo99]. We study this proposed network architecture (TCP/IP over PPP and ATM over ADSL) as a part of this thesis.

1.2 Research Objectives

The main objective of this research project is to develop an understanding of the interoperation of the various protocols implemented in the ADSL network architecture being considered and hence gain insight into the related quality of service (QoS) issues. This thesis is based on a simulation study done to characterize and analyze the performance of the new protocol stack (TCP/IP over PPP over ATM) when implemented over asymmetric links. The simulation models that are developed can be used in the future for further study. The various research goals accomplished by means of a comprehensive simulation study are as follows:

- Analysis of the effect of asymmetric links (faster downstream and slower upstream) on TCP performance in presence of unidirectional and bi-directional data transfer;
- Validation of our baseline simulation models by comparing our results with some of those already published and with theoretical results;
- Analysis of the effect of overhead at the AAL-ATM layers in addition to the basic *asymmetry problem*;
- For scenarios with bi-directional traffic -- characterization of system performance in presence of bursty vs. non-bursty data traffic over the upstream;
- Implementation of LLC-based PPP encapsulation [GKL+98] simulation model in OPNET Modeler™;
- Characterization and analysis of TCP/IP performance when implemented over PPP and ATM over asymmetric links -- effect of additional overhead due to PPP encapsulation;
- Comparison of certain proposed solutions and their effectiveness in presence of the new protocol stack;
- Implementation of Smart ACK Dropper-ACK Regenerator (SAD-AR) [KSK99] techniques in OPNET Modeler™;
- Characterization of performance improvements of certain queuing/scheduling and policing mechanisms that provide differentiated service to TCP acknowledgements and data packets.

1.3 Organization of the Thesis

The remainder of the report is organized as follows. Chapter 2 introduces the xDSL family and provides a bird's eye view of a typical ADSL network architecture. In chapter 3, we present

an overview of the various performance issues related to the different network protocols being used over ADSL in the access network. In chapter 4, we introduce OPNET Modeler™, a network simulation software that was used for this simulation study and portray our simulation methodology, the different simulation scenarios considered and the various simulation models developed at every stage of this research. We present our simulation results and analyze these results in chapter 5 and finally present the main conclusions drawn from this study and suggest areas for future research in Chapter 6.

CHAPTER 2. BACKGROUND

2.1 xDSL Family

ADSL is one of a variety of digital subscriber line (DSL) systems being built upon the existing twisted-pair copper loop infrastructure. These different kinds of DSL technologies have popularly become known as xDSL indicating that it is a *growing* family of related standards and technologies, all set to provide high speed communication over long spans of twisted pair wire. Here 'x' stands for any one of the different versions of DSL systems shown in the Figure 2.1 below.

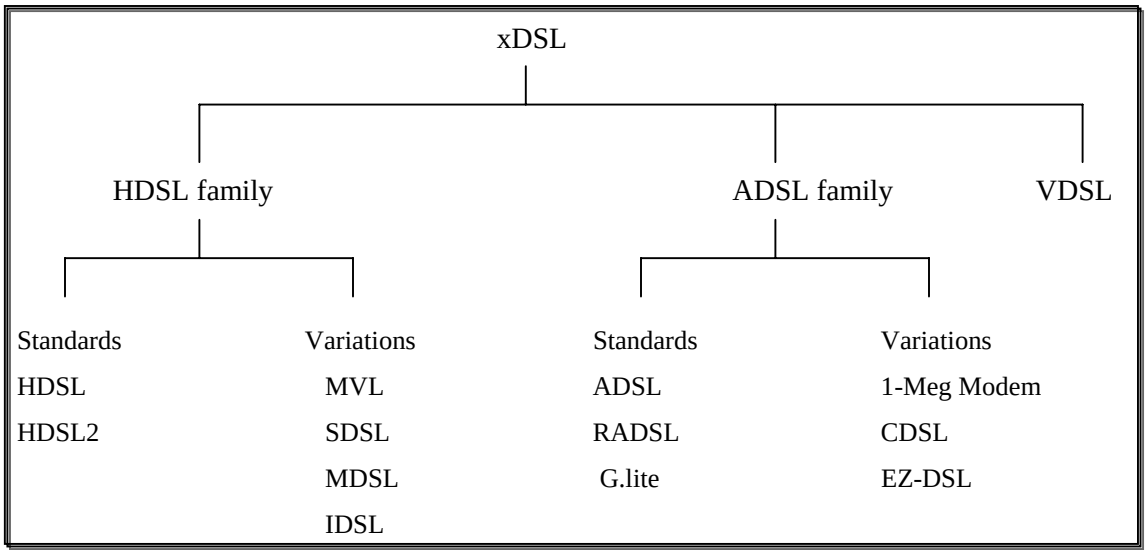

Figure 2.1 The xDSL Family [Gor99]

The following is a brief overview of some of the major categories of the xDSL family [Lan98]. To obtain a further insight into the various xDSL versions, the reader is encouraged to refer to [ADS], [Gor99] and [Lan98].

2.1.1 HDSL

High-speed digital subscriber line (HDSL) was the first version or variant[1] of DSL system introduced. Developed by Bellcore in the late 80's, it provides a full duplex DS1 over two twisted pairs up to 12,000 feet long. Before HDSL, DS1 services could be supplied only by either installing T-carrier repeaters in the loop or by using fiber optics and by perhaps installing a new

[1] Integrated Services Digital Network (ISDN) in itself was the original DSL system introduced.

fiber cable. However, both these approaches were costly and time consuming. HDSL was designed to provide these T1 connections quickly and inexpensively by using the existing loop pairs and only requiring some additional equipment at the central office and on customer premises.

HDSL is attractive mainly because of its pioneering high-speed service (transmission bandwidth of 800 kb/s over each telephone subscriber loop) in the local loop. However, it is not a good candidate for universal broadband services for a couple of reasons. Firstly, it requires two pairs to provide a full 1.5 Mbps of service and secondly, it cannot co-exist with voice telephone services on the same pair.

2.1.2 SDSL

Single-line DSL although being symmetric in nature (same transmission rates on the upstream and the downstream) like HDSL, stands distinct in that it operates over a single-twisted pair and allows transport of analog POTS service on the same line. The data rates generally range from 160 kb/s to 2.048 Mb/s [Lan98]. The problem with SDSL though is that it is NEXT (Near End Cross Talk) limited. We will look into further details of the various types of cross talks in Section 2.3 on *Physical Layer Issues*. This limits its operational span to about 10,000 feet or less at higher data rates. SDSL thus turns out to be a narrowly targeted service where a user is close to the central office, twisted pairs are scarce, and the upstream channel bandwidth is as important as downstream. However, the relative wide bi-directional bandwidth may appeal to many branch offices that have high upload as well as download requirements.

2.1.3 IDSL

IDSL is a rather *vendor-driven* entry into the xDSL clan. IDSL stands for ISDN (Integrated Services Digital Network) DSL, which is actually redundant since ISDN was actually the original DSL technology. IDSL in fact closely mirrors the ISDN Basic Rate Interface (BRI) qualification [Gor99]. It provides 128 kb/s of pure data transport and is not intended to support analog POTS on the same access line. The main difference between IDSL and ISDN is that rather than terminating in an ISDN switch, IDSL terminates on a router with most of its traffic being to and from the Internet. IDSL thus provides dedicated access rather than switched service. IDSL uses a single twisted pair and covers span of about 18,000 feet.

IDSL thus seems to meet two of the typical criteria for personal broadband services: large coverage span and single line service, but misses in its incompatibility with voice service and falls short of the bandwidths most of us will need in the future.

2.1.4 ADSL

As the name suggests, ADSL transmits asymmetric data streams, with much more bandwidth along the downstream (to the subscriber) as compared to the upstream (from the subscriber to the service provider). The reason for the asymmetric nature of transmission has less to do with transmission technology than with the cable plant itself. Twisted pair telephone lines are bundled together in large cables. A typical configuration has a cable carrying about fifty twisted pairs towards the subscriber, while the cables coming out of a central office (CO) may have hundreds or even thousands of pairs bundled together. An individual line from a CO to a subscriber is spliced together from many cable sections as they fan out from the central office. Twisted pair wiring is basically used to minimize the interference of signals from one cable to another caused by radiation or capacitive coupling. However, signals do couple to a certain extent, and couple more so as frequencies and the length of line increase. Also, it turns out that if one tries to send symmetric signals in many pairs within a cable, then the data rate and length of line one can attain is significantly limited. The reasoning behind this involves the concept of two types of crosstalks, namely the near-end cross talk (NEXT) and the far-end crosstalk (FEXT).

Furthermore, many of the target applications for digital subscriber services are asymmetric. Video on demand, home shopping, Internet access, remote LAN access, multimedia access, all feature high data rate demands downstream, to the subscriber, but relatively low data rates demands upstream. For example, MPEG movies with simulated VCR controls, require 1.5 or 3.0 Mb/s downstream, but can work just fine with no more than 64 kb/s (or even 16 kb/s) upstream. It is generally considered that a 10:1 ratio of downstream to upstream bandwidths does not compromise performance in most cases [ADS]. Table 2.1 shows some of the typical ADSL applications and corresponding bandwidth requirements. It is also noteworthy that ADSL allows the co-existence of these data applications along with the plain old telephone service (POTS) over the same pair and in effect not affecting the POTS. Thus, it provides the user with *always on* service i.e. one can talk over the phone or send a fax while surfing the Internet over the same line. In conclusion, ADSL can be looked upon as a relatively mature technology having been through several years of development and trials.

2.1.5 RADSL

Rate adaptive DSL can be looked upon as an intelligent version of ADSL. It can automatically assess the condition of the twisted pair and optimize the line rate for a given line quality. This is an important feature because the quality of spans varies widely depending on age, installation

practices, proximity to external electrical interferers, and a variety of other factors such as weather conditions, time of the day etc. RADSL modems automatically compensate for these conditions, allowing full bandwidth under optimum conditions but lowering bandwidths (rather than allowing error rates to rise) if line quality degrades. RADSL allows the service provider to provision service without having to measure a line and manually adjust or choose a modem to match. In addition, it also allows a provider to provision, via a management system, a fixed line rate to match a particular service and tariff class, rather than having to inventory a number of modems of specific data rates.

Table 2.1* Typical ADSL applications and corresponding bandwidth requirements [ADS]
The table provides the whole range of bandwidth that various applications generally require, although the whole range may not need ADSL deployment.

Business Applications	**Bandwidth Required (Mb/s)**
Telecommuting/SOHO	0.014 – 6.0
Internet	0.500 – 1.5
Desktop Video Conferencing	0.128 – 1.5
Distance Learning	0.500 – 6.0
Local Web Site Hosting	0.500 – 6.0
Telemedicine	0.500 – 6.0
Computer Telephony Integration	0.128 – 1.5
Consumer Applications	**Bandwidth Required**
Internet	0.500 - 1.5
Education	0.500 - 6.0
Video on Demand	3.0 - 6.0
Shopping	0.500 - 1.5
Interactive Video Games	0.128 - 6.0

2.1.6 G.LITE:

As far as the deployment of ADSL is concerned, a key roadblock is the necessity of a splitter (or diplex filter) to be installed at the customer premises to enable simultaneous voice and data service [Kwo99]. The splitter basically prevents interference between ADSL signals and POTS devices (e.g. phone, fax machines etc). Although ideally voice and data (ADSL signals) occupy distinct frequency bands, the higher frequency ADSL signals might affect the POTS devices in the home, because of the non-linearities in the POTS devices. Reciprocal interference may also occur wherein a POTS device causes interference to the ADSL modem operation. Installation of a splitter at the customer premises will require a *telco* technician to come to the customer premises. An alternative approach is to have a DSL modem with a built-in POTS splitter, but this will require a new wire to be installed at the customer premises to reinsert the voice signals back to the home POTS wiring.

This has motivated the development of a *splitterless* ADSL modem, which could be bought *off the shelf* and would not involve the inconvenience of installation of new equipment or wiring. The user could just plug in the modem like today's analog modems (or have it in-built in the PC) and get on line. Such pre-standard splitterless ADSL modem was known as universal ADSL (UADSL) [ADS]. This then became known as G.Lite specification (officially, ITU-T Recommendation G.992.2) [ITU] at the ITU-T in October 1998.

G.Lite is similar to RADSL since it is rate-adaptive. However, it does not have a splitter to prevent the interference between POTS devices in the home and the G.Lite modem. This in effect puts a limit on the data rate achievable using G.Lite, which now not only depends on the length of the local loop but also on the in-home wiring conditions and connected POTS devices. Generally, under *acceptable* home conditions and loop quality, G.Lite is expected to provide up to 1.5 Mb/s downstream and up to 512 kb/s upstream over 18,000-ft local loop span and lower data rates over larger spans. The G.Lite modem basically cuts back its power as soon as it detects a POTS device (e.g. a phone) going off hook, in order to prevent interference from itself to the POTS devices. This procedure is known as *fast-retrain* after which the G.Lite modem operates at lower bit rates (as limited by the lower power available). The G.Lite modem requirements need to be supported by the American National Standard Institute (ANSI) ADSL standard (T1.413) equipment at the central office (CO) for a wider and rapid deployment. The standardization is presently under way and it is expected that G.Lite will be built in many PCs (like analog modems) from year 2000.

2.1.7 VDSL

Very-high-rate DSL is an emerging technology (still undergoing standardization) that promises to deliver data rates as high as 52 Mb/s over the subscriber downstream over short spans of copper wire, and lesser rates over longer spans. Upstream rates are in the 1.5 Mb/s to 2.3 Mb/s range. 52 Mb/s can be supported over a 1000-ft pair, while up to 15 Mb/s can be provided for 3000-ft spans.

The high rates supported by VDSL over short spans are attractive for applications such as distribution of digital TV programming to the neighborhood, and for fiber to the curb (FTTC) applications. At this moment, although several different VDSL formats have been proposed, standardization is still in its early stages.

2.2 ADSL-Based Broadband Networks: System Architecture

Figure 2.2 shows a bird's eye view of a typical ADSL-based broadband network.

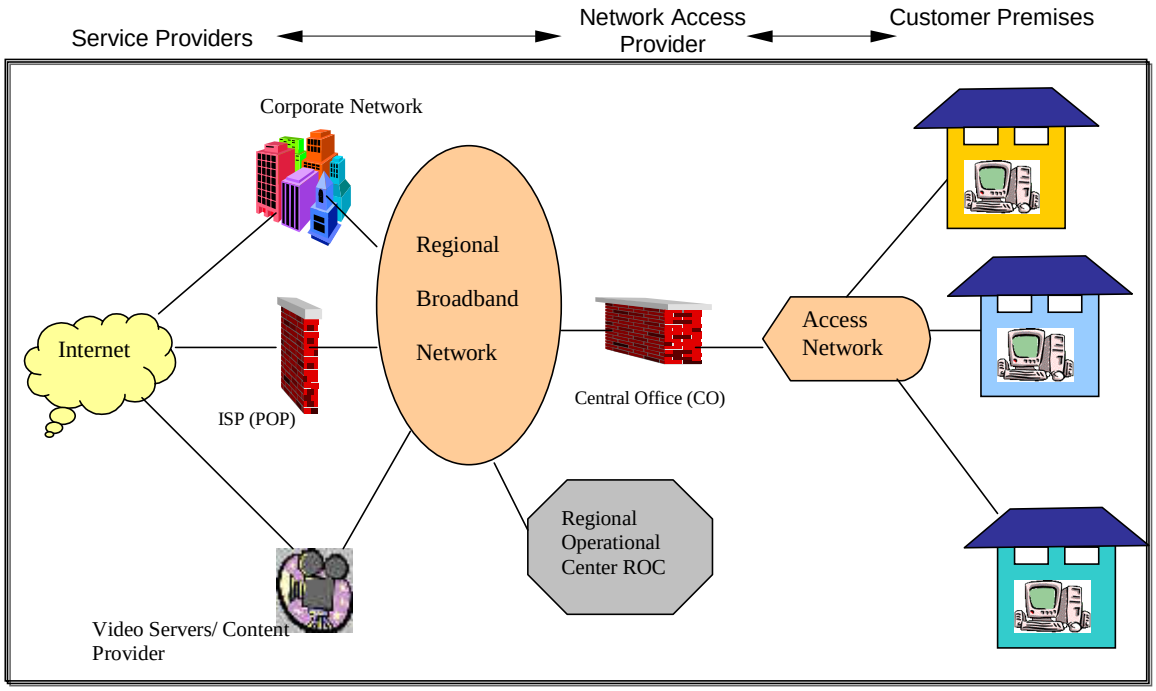

Figure 2.2 Broadband Connectivity (Access Configurations): Internet, Corporate Networks, Local Content, and Peer-to-Peer Communications

As seen in Figure 2.2, residential broadband requirements can be typically classified into four types of destinations to which the broadband network needs to provide connectivity. These

destinations include the Internet (through an ISP), the corporate network (for telecommuting), content providers, and peer-to-peer or person-to-person communication [Kwo99].

The end-to-end ADSL system architecture can be divided into the following sub-networks: the customer premises network, access network, the regional broadband network, and the service provider network. A more detailed architecture of the system is shown in Figure 2.3.

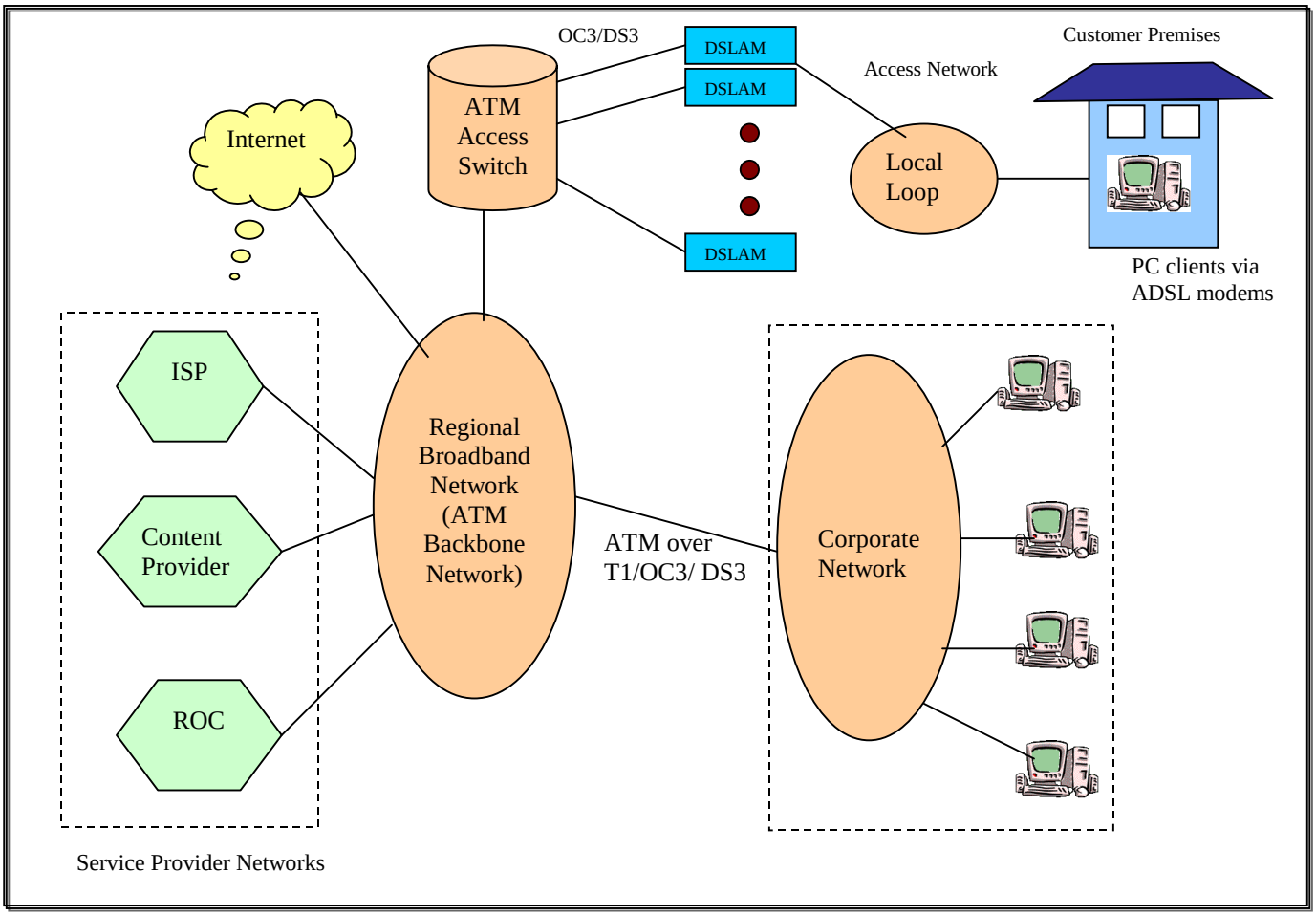

Figure 2.3: ADSL-based Broadband Network Architecture

2.2.1 Customer Premises Network

The customer premises network basically includes residences, small business offices and home offices (SOHO). The ADSL modem at the customer premises called the ADSL terminal unit at the residence (ATU-R) terminates the physical layer DSL signals. Typically a customer premise network may contain a single host connected to the ATU-R. However, SOHOs may contain one or more PCs or workstations (Figure 2.4). In such case the multiple PCs or other non-PC device may reside on a LAN and share a common gateway. The gateway to the external

network can be dedicated hardware such as a router or an external DSL modem, or a PC server acting as a router or a proxy server. A router or a PC generally has two network interface cards (NICs): one for connecting the ADSL modem (or serving as the ADSL modem if the PC has an internal ADSL modem card) and the other for the home LAN. In such a case, the ATU-R can even provide higher layer functions, such as adaptation from ADSL to data interfaces on the customer premises side (i.e. either a LAN interface or NIC in a PC).

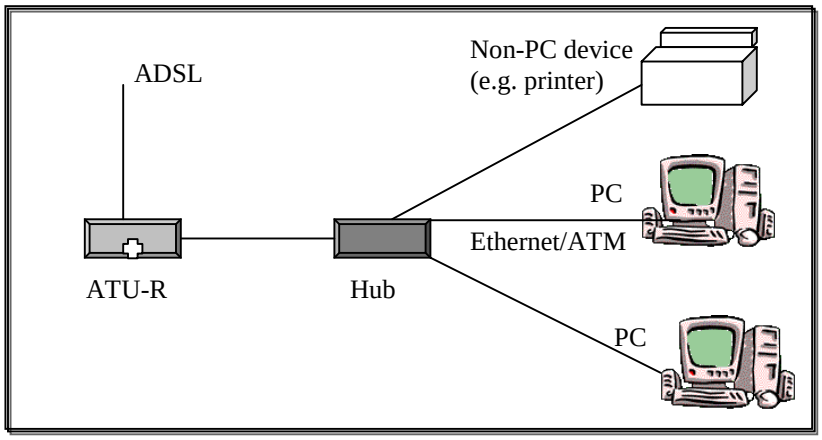

Figure 2.4 A typical SOHO connected to the external network through ATU-R

2.2.2 Access Network

The ADSL access network is comprised of the ADSL modems and the access multiplexer system at the central site (CO or a remote terminal) and the ADSL modems at the customer premises connected via the local loop. The ADSL modem at the central site is called as ATU-C, which terminates the physical layer DSL signals on the central site side of the access network. The access multiplexer system and the ATU-C are generally referred to as a single unit called the DSL access multiplexer (DSLAM). Since wide-area network (WAN) interfacing is quite expensive, implementing a multiplexing scheme that provides high subscriber concentration while guaranteeing individually negotiated quality of service (QoS) will be an important asset for network operators. In case of an ATM over ADSL scenario, the DSLAM effectively acts as an ATM multiplexer, which terminates the incoming ATM signaling protocol from each ADSL customer and generates a single ATM Forum standard user-network interface (UNI) signaling interface. The traffic from several such DSLAMs is then concentrated and switched onto the regional broadband network by an ATM access switch. The ATM access switch may or may not be co-located with the DSLAMs at the central site.

2.2.3 Regional Broadband Network

A regional broadband network, typically based on a synchronous optical network (SONET) infrastructure, interconnects several COs in a given geographical area. There is an increasing trend to deploy ATM over this SONET infrastructure for broadband connectivity among COs and service providers.

2.2.4 Service Provider Network

A service provider network typically includes the Internet service providers (ISP), corporate networks, content provider and regional operation center (ROC). ISPs mainly provide resources needed for efficient access to the Internet through the access network. Typical examples of these resources are bandwidth to the Internet backbone, domain name system (DNS) service, IP addresses, Web caching etc. The corporate networks are used by corporations to provide their employees (telecommuters) direct remote access (not through an ISP) to their intranet. Some other smaller corporations provide indirect access to their network through an ISP and hence do not have point-of-presence (POP) in the access network. A content provider network typically consists of a server farm for distributing content such as audio or video on demand, online shopping, distance learning etc. The network operator uses the regional operational center (ROC) to offer value-added services such as network management.

2.3 Market for ADSL and End-to-End Service Requirements

2.3.1 Business Opportunities for ADSL-Based Broadband Networks

ADSL technologies provide a new platform for high-speed connectivity and delivering broadband services to small business offices and homes (SOHO). It is this high-speed connectivity to homes, small businesses and remote offices that will create many new business opportunities for service providers. Such opportunities include:

- **High-Speed Internet Connectivity**

With the explosive growth of the Internet in recent years, it is one of the major market-drivers for ADSL. ADSL can deliver not only high-speed Internet access, but also an *always on-line* service that allows the user to surf the Internet while talking on the phone! In addition, the service provider can also offer content providers and consumers Web-hosting services and tools.

- **Branch Office Connectivity**

ADSL lends itself to branch office connectivity by effectively replacing leased lines.

Most business personal computer applications perform asymmetric communication, making ADSL an appropriate technology to connect remote offices to the enterprise.

- **Telecommuting Services**

Today, more and more corporations are embracing telecommuting, which provides a ripe opportunity for ADSL technology. With high-speed connectivity to employee's homes, the network operator can offer *virtual office* experience to telecommuters. In addition, the network operator can provide nationwide or even worldwide access to business networks either through Internet or wide-area broadband networks. This has led to an increased willingness among corporations to invest in ADSL network services.

- **Business-to-Business Services**

Businesses that have a common bond may want to share a Virtual Private Network (VPN) that is reliable and secure. The network operator can create a virtual private network using ADSL in conjunction with the existing backbone networks to interconnect businesses. Further businesses outside the operator's service area can be connected through the Internet using certain secured tunneling protocols.

- **Content Delivery Service**

Although a high bandwidth network connection looks attractive, it can be made more compelling by enhancing the quality and quantity of the content delivered over it. Content may take any forms such as shopping catalogs, travel services, real estate listings, yellow pages, music, video, games etc. Such a combination of high-speed networking and enriched content delivery will present an attractive offering to business and residential customers.

2.3.2 Functional Service Requirements

Now with these opportunities in mind, certain key functional requirements that must be addressed to enable a mass market for ADSL are:

- **Easy Migration from Existing Internet Service Provider (ISP) Access Infrastructure**

Presently, ISPs already have an infrastructure to support modem dial-up access. Hence any new broadband Internet access solution will have to take this existing infrastructure into account. Ideally, the broadband service model for accessing ISP service can reuse most of the networking, management and administration infrastructure, such as IP address and domain name administration, and hence will not require a paradigm shift for the ISP.

- **Simultaneous Connectivity: Internet and Corporate Network**

A telecommuter working from home may need to access the Internet while working, either for work related or non-work related reasons. Two ways to allow such simultaneous connectivity are

either to access the Internet through the corporate network's gateway or to support a separate Internet connection simultaneously with the corporate connection. In many cases (especially for accessing Internet for non-work related reasons), the second way may seem appropriate since it allows the telecommuter to access Internet directly without using the corporate network's resources. However, some corporations may not trust the simultaneous Internet connection for security purposes.

- **Multi-protocol Support**

Since all corporations need not necessarily run Internet Protocol (IP) exclusively, providing corporate connectivity requires interconnecting non-IP networks over the ADSL access network. Hence, such connectivity will involve protocol negotiation and address assignment.

- **Security**

In addition to Web-browsing, since ADSL will also be used for applications such as telecommuting and virtual private networking, security becomes crucial. Telecommuters and branch offices would like to communicate with the enterprise in a fashion that supports authentication, authorization and privacy.

- **Multicast Communication**

Live events are now commonly offered in audio and video on the Internet, and multicast is the preferred delivery mechanism. For example, during an election, most Internet sites carrying real-time results are jammed with a large number of users trying to log in. Instead if the information could be pushed out using multicast, then congestion could be avoided. Thus IP multicast delivery to homes and small businesses is another critical requirement for the ADSL access network.

- **Quality of Service (QoS) Support**

The network operator should be able to provide multiple service classes to satisfy the different needs of users (e.g. *ardent* network users as against *occasional* users). This is important since it has become very clear that many services including Internet access, cannot depend solely on a *one size fits all* paradigm. Some of the major broadband applications that are to be supported by ADSL include real-time applications such as multimedia streaming, video-conferencing etc. Such applications require some kind of a guarantee or quality of service (QoS) to ensure performance. QoS also implies that the network can prevent *aggressive* or *rogue* users from consuming network bandwidth and degrading performance of other users.

2.4 Physical Layer Issues

There are in effect a number of factors [Kwo99] which allow DSL technologies to offer broadband transmission over the existing twisted pair copper infrastructure, the key factor being that DSL technologies utilize the spectrum resources beyond the 3.3 kHz voice channel.

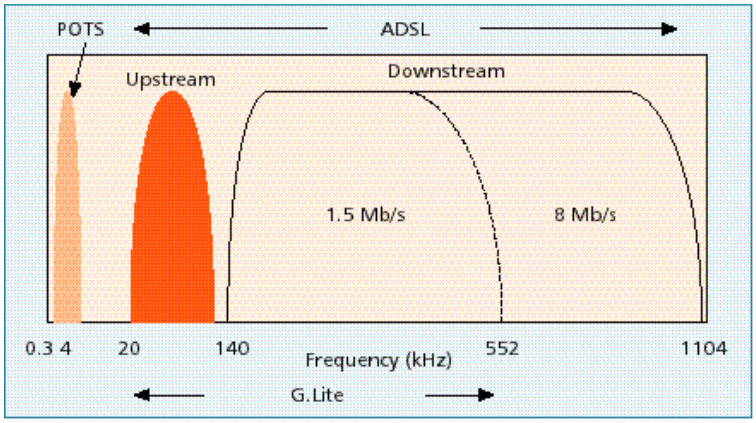

Figure 2.5 Frequency spectrum of POTS (analog modems) vs. ADSL and G.Lite [Kwo99]

As shown in Figure 2.5, an analog modem, which uses a conventional telephone connection, is limited to the voiceband spectrum (0-4 kHz) because the signal needs to fit within the public switched telephone network (PSTN) bandwidth, which is divided into 4kHz-based circuits. On the other hand, DSL modems can use 1 MHz or wider bandwidth since they need to operate only over the local loop (generally limited to a distance of 12,000-18,000 ft), which is dedicated to a particular home/office from the central office. Further, DSL modems take advantage of today's highly sophisticated digital signal processor and advanced modulation technologies to dynamically adapt the transmissions to the unique loop characteristics of an individual twisted pair to optimize bit rates. Two such popular modulation techniques used for ADSL are *carrierless amplitude and phase modulation (CAP)* and *discrete multitone (DMT)*. DMT is an ANSI (T1.413) standard, while CAP is a proprietary standard (patents owned by, among others, AT&T, Lucent Technologies and Globepsan) [Bai97].

CAP is similar to quadrature amplitude and phase modulation (QAM), in that the two orthogonal carriers are modulated and then combined. However, the main difference is that in CAP, the orthogonal signal modulation is done using two digital transversal bandpass filters with equal amplitude and $\pi/2$ phase difference. The signals are then combined and a digital-to-analog

converter (DAC) converts the combined digital signal back to analog before transmission. The carrier itself is suppressed before transmission, and hence the adjective *carrierless* [Bai97, ADS]. DMT is different from CAP, using many narrow-band carriers, all transmitting simultaneously in parallel. It uses frequency-division multiplexing (FDM) to achieve the simultaneous transmission of subcarriers. Multicarrier modulation requires orthogonality between all the subcarriers. In DMT, this is achieved using fast Fourier transform (FFT) which is one of the convenient ways to do this, although other sophisticated transforms have been developed recently. DMT's ANSI T1.413 standard specifies 256 subcarriers, each with a 4 kHz bandwidth. They can be independently modulated from zero to a maximum of 15 b/s/Hz. This allows up to 60 kb/s per tone. (Some implementations support 16 b/s/Hz, giving rates of 64 kb/s per tone.) At low frequencies, where copper wire attenuation is low and the signal to noise ratio (SNR) is high, it is common to use a very dense constellation (greater than 10 b/s/Hz is typical). In unfavorable line conditions, modulation can be relaxed to accommodate a lower SNR, usually 4 b/s/Hz or less, and still deliver the necessary noise immunity. There is also a third variant called discrete wavelet multitone (DWMT), in which the wavelet transform is used to achieve multichannel modulation instead of the fast Fourier transform (FFT). It is claimed that this will improve the system performance by improving the spectral containment of the channels.

While the DSL technologies are deployed and made operational, certain factors that may have an impairing or degrading effect on their performance are worth considering. The impairments can be broadly classified into two main categories: physical and electrical. Physical impairments arise from situations such as the use of mixed gauges, bridged taps resulting in unterminated branches that reflect signals and cause interference at the higher xDSL frequencies, improper splicing causing *micro-outages* of short duration, use of proprietary line extenders and digital loop carriers (DLC). Electrical impairments are generally termed as *interferers* or *disturbers*. In addition to the universal thermal noise that is encountered in most communication systems, other noise contributors such as radio frequency interference (RFI), impulse noise and crosstalk impair xDSL schemes. At xDSL frequencies a major concern is the radio frequency interference (RFI). This is caused mainly by the several amplitude modulation (AM) radio stations that broadcast in the same frequency range in which the xDSL systems operate. Impulse noise is generally caused from electrical appliances, lightning or, most relevantly, with a phone going off-hook or ringing. It is wideband in frequency and narrow in time. Crosstalk causes by far the largest contribution to capacity limiting noise for DSL systems [CKB+99]. Two different types of crosstalks encountered in multipair access cables are near-end crosstalk (NEXT) and far-end crosstalk (FEXT). NEXT is the interference that appears on another pair at the same end of the cable as the

source of the interference. On the other hand, FEXT is the interference that appears on another pair at the opposite or far end of the cable to the source of interference. NEXT affects any systems which transmit in both directions at once (e.g. echo-canceling systems), and it generally tends to dominate over FEXT wherever it occurs. NEXT can in principle be eliminated by not transmitting in both directions in the same band at the same time, separating the two directions in either two non-overlapping time intervals or two non-overlapping frequency bands (as in case of ADSL). This converts duplex transmission into independent simplex transmissions, avoiding NEXT at the cost of reduced bandwidth (over the upstream for ADSL). At high enough frequencies at which ADSL operates, the advantage of not transmitting NEXT outweighs the disadvantage of reduced bandwidth. Readers interested in further details about the modulation techniques (CAP/DMT) used for ADSL and other physical layer issues are encouraged to consult [ADS99, Bai97, CKB+99, Gor99].

2.5 Higher Layer Issues

The field of xDSL has recently proved to be a *hot* area for researchers. Although numerous research papers and articles have been published in this field, most concentrate on the physical layer and deployment issues. Only a few research papers address the issues of end-to-end broadband service over DSL in terms of protocol architectures, performance analysis, and quality of service. The development so far in this direction is still in early stages and thus has a lot of scope for further research and innovation. This research thesis mainly deals with such higher layer issues for one of the popular ADSL network architectures, focusing mainly on the ADSL access network part of it. It handles the problems of performance optimization and delivering service guarantees over such protocol architecture. A detailed discussion of higher-layer network protocols and related issues is thus reserved for the following chapters.

2.6 Chapter Summary

The chapter began with an overview of the different flavors of DSL technologies available (or being developed) and categorized under the xDSL family. This was followed by a discussion of the system architecture of a typical ADSL-based broadband network and its components: the customer premises network, the access network, the regional broadband network and the service provider network.

Then we discussed about the various business opportunities that ADSL could create for service providers and certain key functional service requirements that these service providers need to address in order to gain a mass market.

The section on *Physical Layer Issues* then provided with some insight into the technology which allows DSL schemes to offer broadband transmission over the existing twisted pair copper infrastructure. It also briefly discussed the various physical level impairments that these DSL schemes might have to deal with.

Finally in the section on *Higher Layer Issues*, we noted that there is a lot of scope for research and innovation in the area of end-to-end deployment of broadband services over ADSL and related higher network-layer issues.

The remaining portion of this thesis further investigates such issues and attempts to develop a better understanding towards performance characterization and optimization, in ADSL networks.

CHAPTER 3. Network Protocols and Related Performance Issues

Chapter 2 provided us with an overview of the xDSL family and typical configuration of ADSL-based networks. With that background we now aim to gain further understanding of the various *higher-layer* issues encountered in such networks. The first two sections of this chapter provide a brief outline of asynchronous transfer mode (ATM) and the Internet protocol suite (also popularly known as the TCP/IP protocol suite) that are being widely deployed over DSL technologies. The various issues that arise from the interoperation of these protocols among themselves and with ADSL are then treated in detail in the subsequent sections (3.3 through 3.5). We provide a brief overview on IP quality of service (QoS) in Section 3.6. Further, in Sections 3.7 and 3.8, we respectively take a look at two more protocols -- namely the point-to-point protocol (PPP) and the layer 2 tunneling protocol (L2TP) that are being promoted by current industry standards for ADSL networks. Finally we introduce one of the end-to-end ADSL network architectures that is being considered for ADSL deployment.

3.1 Asynchronous Transfer Mode (ATM)

ATM is a connection-oriented, packet switched (or, rather, *cell switched*) technology designed to offer flexible transmission support for a wide variety of services such as voice, video and data. ATM technology gained importance in the 1980s and early 1990s and, given its rapid growth, appears destined to play a major role in the public and private broadband networks of the future. ATM is a *connection-oriented* scheme because it allows hosts to communicate with their peers by establishing virtual circuits or connections in a fashion similar to circuit-switched networks. It is a cell-based switching technology because ATM segments all its information into fixed-size packets; this simplifies the switching and multiplexing functions. To distinguish these fixed-length packets from the more common variable-length packets normally used in most packet switching technologies, they are called *cells*.

3.1.1 ATM Resources

In this section, we will take a brief look at certain key resources or aspects, which make ATM one of the most flexible and popular high-speed networking technology.

- **ATM Cell**

The ATM PDU is a fixed length packet called *cell*. It is 53 bytes in length, with 5 bytes of ATM cell header and 48 bytes used by the AAL and user payload. The main advantage of having packets of a fixed, small size is that it enormously simplifies the design (both hardware and software) of ATM switches allowing building of fast, highly scalable switches. The faster switching of packets is important for real-time applications, wherein end-to-end delay is one of the most crucial qualities of service metrics. However, as compared to variable large size packets, having small fixed size cells also has the drawback of greater overhead, which may adversely affect parameters such as the bandwidth efficiency. We gain further insight into this trade-off in subsequent sections of this chapter.

- **ATM Traffic Parameters**

A host (ATM device) generally specifies the following traffic or contract parameters during its connection setup phase [Bla95, ATM99].

- **Peak cell rate (PCR)** is the permitted burst profile of traffic associated with each user-to-network interface (UNI) connection. In other words, it is the upper bound on the instantaneous transmission rate at which a host can transmit and is in effect the reciprocal of the minimum cell inter-arrival time.
- **Sustainable cell rate (SCR)** is a permitted upper bound on the average rate for each UNI connection (i.e. the average throughput).
- **Minimum cell rate (MCR)** is the minimum cell rate desired by the user for a given UNI connection.
- **Cell transfer delay (CTD)** is the total delay experienced by a cell from the time it enters the network to the time it leaves the network. This parameter includes propagation delays, queuing delays and service times at intermediate ATM devices.
- **Mean cell transfer delay (MCTD)** is the average of a specified number of cell transfer delays for a given number of connections.
- **Cell loss ratio (CLR)** is the ratio of the number of cells lost to the total number of cells sent by the source. Cells are generally lost in a network due to congestion or buffer overflows, or some other factors causing errors. If network congestion occurs, then the network first discards cells with a lower priority. CLR can be set separately for the low priority and high priority cells.
- **Cell delay variation (CDV)** describes the variability pattern or variance of cell arrival and propagation in a network.

- **Burst tolerance (BT)** is the maximum burst size that can be sent at the PCR. It is used to control the traffic entering the network, using the leaky bucket algorithm, which puts all the arriving cells in a buffer (bucket) and sends them at the SCR.
- **Maximum burst size (MBS)** is the maximum number of consecutive cells that can be sent out of the bucket at the PCR. The relationship between the MBS and BT is given by: BT = (MBS-1) / (1/SCR – 1/PCR)

- **Quality of Service (QoS) Provisioning**

Using the various traffic parameters mentioned above, a contract for desired QoS is negotiated between a source and destination. When all the ATM devices in the network between the source and destination are able to support the requested QoS, the connection gets established and all the traffic requirements for that particular type of service category are guaranteed for the duration of that connection. So far six such service categories have been defined. A summary of all these service categories and corresponding traffic parameters requirements is given below [Bla95, ATM99].

- **Constant bit rate (CBR):** This class is used for emulating circuit switching. The cell rate is constant with time. CBR supports applications, which require a fixed amount of bandwidth defined by the PCR. CBR is best suited for applications with stringent requirements on CTD and CDV such as telephone traffic and television.
- **Real-time variable bit rate (RT-VBR):** This service class allows variable rate of transmission while maintaining high demands on CTD and CDV. The rate variation is bound by the PCR, SCR and MBS, which define a worst-case scenario for which the QoS traffic contract will hold. Examples of RT-VBR applications are voice with speech activity detection (SAD) and interactive compressed video.
- **Non-real-time variable bit rate (NRT-VBR):** This class allows users to send traffic at a rate that varies with time depending on the availability of user information. Statistical multiplexing is provided to make optimum use of network resources. NRT-VBR is geared towards applications, which are bursty in nature and do not have constraints on delay and jitter (unlike RT-VBR). Banking transactions, airline reservations and electronic mail are some of the common examples of NRT-VBR.
- **Available bit rate (ABR):** This category of ATM services provides rate-based flow control and is aimed at data traffic such as web browsing, file transfer and electronic mail. Sources vary their transmission rates based on the network feedback, trying to dynamically vary the traffic flow in order to utilize all the available bandwidth in the network not used by other

service categories. This is done by the use of what is known as the resource manager (RM) cell. A source sends the RM cell to the desired destination and includes in it the rate at which it would like to send data cells. Switches along the path compare this requested cell rate with the available resources. If enough resources are available, the RM cell is then passed on unmodified; otherwise the requested rate is decreased before the RM cell is passed on. Once the cell reaches the destination, it is then turned around and sent back to the source, which thereby learns about the rate at which it can send data cells. RM cells are sent periodically to keep track of the available resources. Although the standard does not require the CTD and CLR to be guaranteed or minimized, it is desirable for switches to minimize delay and loss as much as possible. The users are allowed to declare an MCR, which is guaranteed to the connection by the network.

- **Unspecified bit rate (UBR):** This service category is better known as the *best effort* service, since it does not specify any bandwidth, has no guarantees for throughput, and places no constraints on delay and jitter. Since most LANs and IP networks also offer only best effort service, LAN emulation, IP over ATM and other non-critical applications use UBR. In general UBR is a good service for handling applications that have built-in retransmission schemes (TCP).
- **Guaranteed Frame Rate (GFR):** Guaranteed Frame Rate (GFR) is the newest of the service classes defined by the ATM Forum [ATM]. The GFR service is intended to support non-real time applications. The GFR service category requires that data cells from a source be organized in the form of frames that can be delineated at the ATM layer. The source is provided with an MCR guarantee under the assumption of given maximum frame size (MFS) and maximum burst size (MBS). This basically means that as long as the source sends conforming frames in a burst that does not exceed the MBS, the source should expect to see its frames delivered across the network with minimum losses. GFR also allows a source to send cells at a rate greater than the MCR and associated MBS, but the excess traffic will be delivered in the limits of the available resources. Furthermore, the service specifies that the excess traffic from several such sources should have a *fair share* of the available resources, although the definition of such *fairness* is implementation specific.

  In addition, GFR allows the use of marked or unmarked frames. A marked frame is a frame in which all the cells have the C bit or the cell loss priority bit set to 1 (indicating lower priority), while in an unmarked frame, all the cells have the C bit set to 0 (indicating higher priority). A cell with the C bit set to 1 is known as a *tagged* cell. This basically allows a network to determine which cells/frames could be dropped when it encounters congestion.

However, GFR does not provide the sources with any explicit feedback regarding the current level of congestion in the network.

All in all, the GFR service category guarantees the user a minimum service rate when the network is congested, while being able to send at a higher rate when additional resources are available.

3.2 The TCP/IP Protocol Suite

Before proceeding to a detailed investigation of the various performance issues related to the transmission control protocol (TCP) and the internet protocol (IP) in the following sections, we first take a glance at the profile of the TCP/IP protocol suite [Ste94]. The main protocols grouped under the TCP/IP suite are the transmission control protocol (TCP), the user datagram protocol (UDP) and the internet protocol (IP). The TCP/IP protocol layer has proven to be a scalable, *layer-2 independent* layer providing a powerful foundation for today's communication networks providing *ubiquitous* advanced services. As a result of its layer-2 independent nature, TCP/IP traffic can be transported across heterogeneous networks.

3.2.1 Transmission Control Protocol (TCP)

An important component of the popularity gained by TCP is the collection of algorithms used by it to perform flow control, congestion control and recovery. This in effect allows TCP to guarantee the reliable, in-order delivery of a stream of bytes. With stream data transfer**,** TCP delivers a stream of bytes identified by sequence numbers. This service benefits applications because they do not have to *chop* data into blocks before handing it off to TCP. Instead, TCP groups bytes into segments and passes them to the lower layer (generally IP) for delivery.

TCP is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. Further, TCP supports multiplexing/de-multiplexing mechanism that allows multiple application programs on a given host to simultaneously carry conversation with their peers over a single connection.

TCP also offers reliability by providing connection-oriented, end-to-end reliable packet delivery through the use of positive acknowledgements (*ACKs*), wherein the receiver, on receiving a packet, sends an ACK to the sender indicating the successful transmission of the packet. TCP also uses a timer, on expiration of which the sender retransmits the data packet. Such a *time-out* mechanism allows devices to detect lost packets and request retransmission. Now rather than sending a single packet and waiting for the ACK, TCP uses the *sliding window*

protocol, which makes efficient use of the bandwidth by enabling the sender to send multiple TCP segments before waiting for the ACK. Further, the receiver specifies the current window size in every packet, which is the number of data bytes that the sender is allowed to send before receiving an acknowledgment. Initial window sizes are indicated at connection setup, but might vary throughout the data transfer to provide congestion control and flow control [Ste94]. As shown in Figure 3.1, initially the sender window is initialized to a single segment size, and typically grows exponentially during slow start till it reaches the threshold 'x' (typically initialized to the receiver buffer window size). If a timeout occurs (the segment reaching late or not reaching at all), the window size is dropped down to a single segment size, while the threshold is dropped to x/2, since TCP looks upon the occurrence of a timeout as an indication of congestion. The window again grows exponentially reaching x/2 (the new threshold), beyond which it increases linearly during the congestion avoidance phase. Further details about TCP behavior can be found in [Ste94].

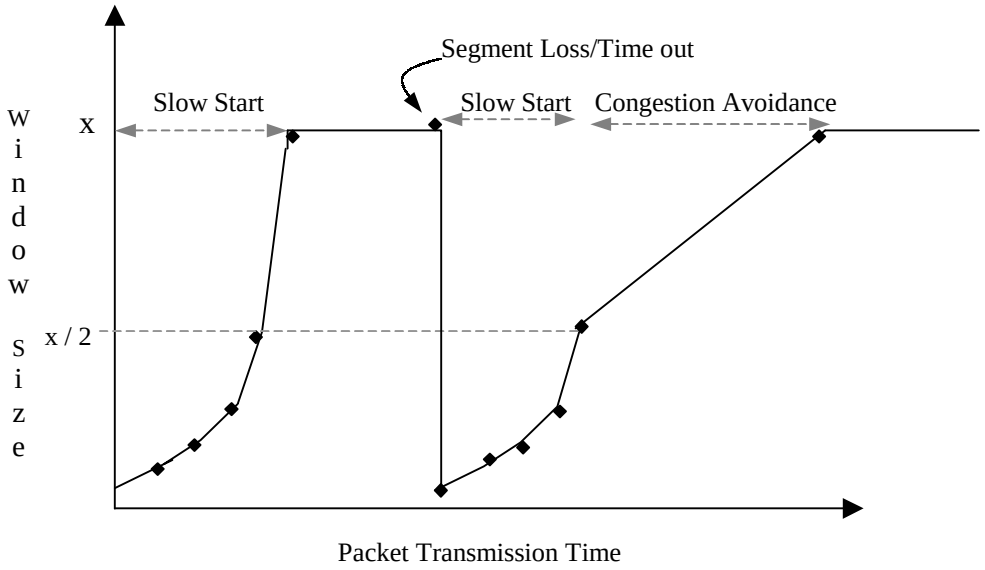

Figure 3.1 Slow start algorithm and congestion avoidance in TCP

3.2.2 User Datagram Protocol (UDP)

UDP is a much simpler transport protocol as compared to TCP. It mainly carries out the multiplexing/de-multiplexing function to enable multiple application programs on a given host to simultaneously carry conversation with their peers over a single connection. Although UDP does not provide flow control or reliable in-order delivery of data, it does ensure the correctness of the data being transmitted by use of a *checksum algorithm*. UDP computes the checksum over its

header, the message body (data), and another header called the *pseudo-header*, which involves the message length, source IP address and destination IP address fields. Thus, UDP is useful in situations where the reliability mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control or for real-time applications where retransmissions are not feasible.

3.2.3 Internet Protocol (IP)

The Internet Protocol (IP) is a network-layer (or layer-3) protocol that contains addressing information and some control information enabling packets to be routed. IP is the primary network-layer protocol in the Internet protocol suite. IP has two primary responsibilities -- providing connectionless, *best-effort* delivery of datagrams through an internetwork; and providing fragmentation and reassembly of datagrams to support data links with different maximum-transmission unit (MTU) sizes. By *best-effort* service, we basically mean that all the traffic is treated equally and a "best effort" is made to deliver it. However, by itself, IP does not provide any guarantee on when and how much of the traffic sent by a source device will actually reach the destination. However, the advent of new technologies and applications (e.g. real time audio/video) has made it necessary to devise schemes, which provide quality of service (QoS) in IP. We gain further insight into IP QoS in Section 3.6.

3.3 ATM over ADSL

With an overview of ATM and the TCP/IP suite of protocols in the earlier sections, we will now discuss the interoperation of these protocols when used over ADSL.

3.3.1 Why ATM?

ATM is well established as a transport technology that is well suited for integrating voice, video and data across local and wide area networks and to provide QoS support to these heterogeneous traffic types; while ADSL is positioned to deliver multi-megabit services over existing copper telephone lines into small offices and home offices (SOHO). The mutual benefits of integrating these two technologies makes the *ATM over ADSL* architecture a natural choice for providing the much-desired high-speed end-to-end connectivity to SOHO customers. ATM as the universal access interface will enable seamless integration of access and backbone networks, and will thus provide SOHO customers access to broadband Internet environments. *ATM over ADSL* will provide the remote-end users with connectivity to any endpoint on the ATM network

including other ATM backbone networks, corporate intranets and other utility servers such as a security server, Internet content caching server, or a video server. This will in effect enhance the Internet services in terms of performance, load sharing and redundancy.

Further, use of ATM as the layer 2 protocol over ADSL access network provides certain distinct advantages such as [Kwo99, ALH+99]:

- **Protocol Transparency:** The network is independent of network layer protocol (IP, IPX, AppleTalk etc) being used.
- **Support of Multiple QoS Classes:** ATM provides the network operator the capability to guarantee different levels of services based on QoS classes mapped to user profiles or applications. ATM's fixed cell length makes it possible to guarantee bounded delays, makes channelized statistical multiplexing feasible, and enables finer implementation of scheduling policies of higher-level data units.
- **Fine-grained Bandwidth Scalability:** ATM bandwidth scalability matches the rate adaptivity of DSL technologies allowing optimum use of each copper loop.
- **Evolution to Other DSL Technologies:** ATM being independent of the physical layer, using it with DSL paves way for evolving access technologies such as Very-high-speed DSL (VDSL).
- **Simplified Network Management:** From the network management point of view, bringing ATM to the user's access network has another implicit advantage. Due to the connection-oriented nature of ATM, most billing, security, maintenance, and operation support infrastructure current switched service networks require can be supported, making it possible to charge on the basis of usage with an end-to-end managed QoS.

3.3.2 *ATM over ADSL* Access Network System

The key issue for an *ATM over ADSL* access network system is the incorporation of ATM interface to newly deployed user-access (UA) device for data-integrated services, while keeping the required *legacy* or *traditional* interface(s) (such as POTS, ISDN, *analog* TV etc.) in the device [ALH+99]. In other words, the UA device (an ADSL modem in most cases) must include an ATM interface to the residential gateway (RG). The RG unit, besides multiplexing traffic from interactive (real time virtual circuits) and non-interactive (non-real time virtual circuits) applications into a single data stream, performs the necessary control plane functions carried transparently to and from the service node by the integrated transport access network. Figure 3.2 shows such an *ATM over ADSL* transport system in the access network.

At the other end of the access network, the service access (SA) and transport access (TA) devices (which mainly comprise of the ATU-C and DSLAM) provide interfaces to the service provider ATM local area network (LAN) and network operator ATM transport wide area network (WAN) infrastructure, respectively. This is generally accomplished via an ATM access switch. This in effect allows ATM to become the common layer for all protocols across all the platforms providing uniform end-to-end QoS management.

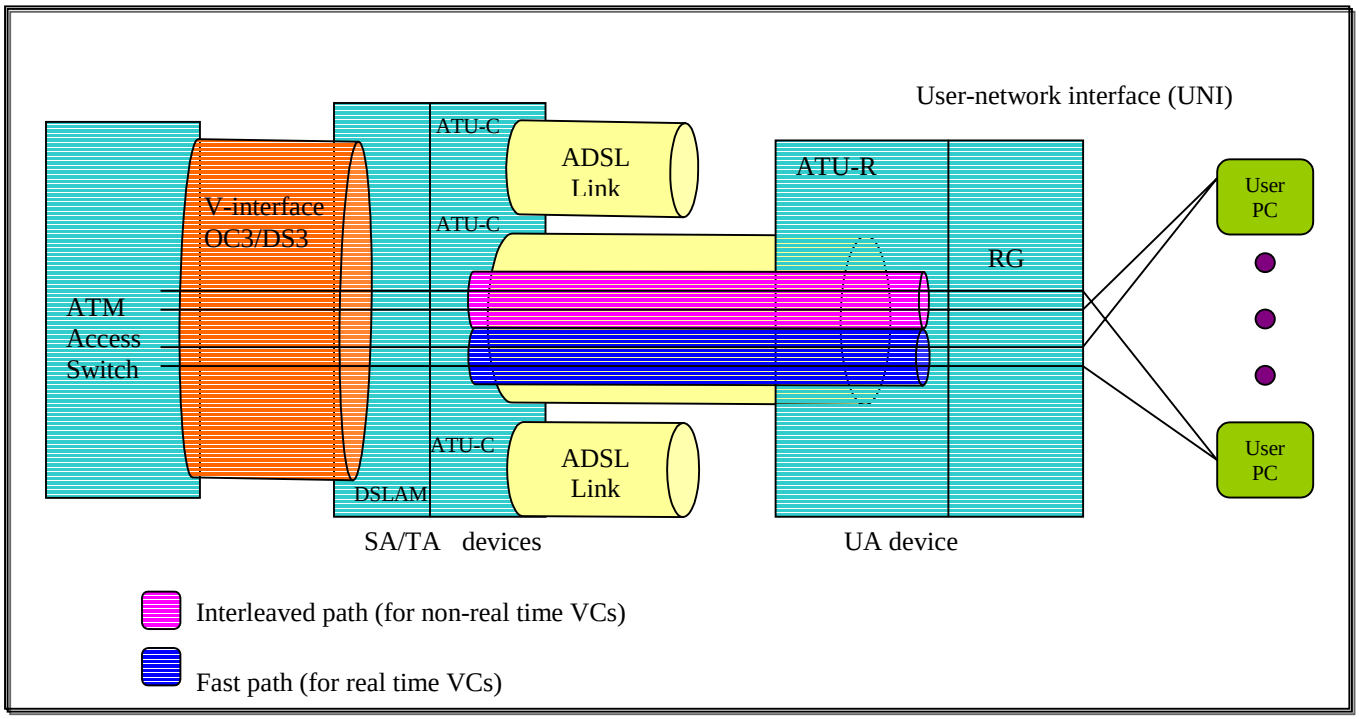

Figure 3.2 ATM over ADSL transport system

Typical ADSL transport of ATM data supports up to two frame based data transport *paths* with asynchronous multiplexing [BP97], namely the *fast path* and the *interleaved path*. The fast path is intended to provide low data transfer delay (up to 2 ms) as appropriate for real time interactive applications while the interleaved path may provide a greater latency (up to tens of milliseconds), but a very low bit error rate.

3.4 TCP and ADSL: The Asymmetry Problem

With applications such high-speed Internet, multimedia streaming, telecommuting etc being the primary market drivers for ADSL, the co-existence of TCP/IP and ATM over ADSL is of particular interest. This makes it crucial to evaluate the end-to-end performance of TCP/IP over

asymmetric networks i.e. systems in which the reverse link (user to network/service provider) is considerably slower than the forward link (network/service provider to user). This bandwidth asymmetry ratio can be as low as 10 as in some of the proposed cable modems and ADSL access services, to as high as 100 or even more where just a telephone line is used as the reverse link [LMS97].

Fundamentally, network asymmetry affects the performance of reliable transport protocols like TCP because these protocols rely on feedback in the form of cumulative acknowledgements from the receiver to ensure reliability. For a unidirectional flow of data, the TCP acknowledgements (*ACKs*) flow over the slow reverse link (upstream) regulating the flow of data packets or throughput over the forward link (downstream). The TCP throughput thus depends upon the feedback information that flows on the restricted upstream. The timely reception of this feedback information when disrupted will cause considerable throughput degradation along the faster downstream. Thus, the slower upstream may act as a *primary bottleneck* for the downstream throughput. For example, a low bandwidth acknowledgement path could significantly slow down the growth of the TCP sender window during slow start, regardless of the link speed in the direction of data transfer.

It must be noted that the asymmetry in raw bandwidth (bits per second) is not the asymmetry directly affecting TCP performance; rather, it is the *normalized asymmetry (k)* defined as the ratio of the raw bandwidths to the ratio of the packet sizes in both directions [BPK97]. Assuming that a receiver sends a TCP ACK for every segment it receives, the forward link speed is restricted to the forward link speed divided by k, when $k > 1$ [BPK97, KDM98]. For example, for a 10 Mb/s forward channel carrying 1000-byte data packets and a 100 kb/s reverse channel carrying 40-byte ACKs, $k = 100/25 = 4$. Thus the forward channel would be restricted to a maximum effective bandwidth of 2.5 Mb/s.

Bi-directional data transfer further exacerbates the *asymmetry problem* discussed above. Downloading a file or a web page (over the downstream) and sending e-mail (over the upstream) at the same time could be imagined to be a typical example of bi-directional data transfer. In such a two-way data transfer scenario, the data traffic along the reverse direction shares the slow upstream with the ACKs of the forward transfer. The smaller TCP acknowledgements get trapped (and bunched together) behind larger data packets resulting in what is known as *ACK compression* [KVR98]. This in effect delays the acknowledgements, hence causing the forward data transfer to stall for long periods of time. Further, this may also cause the round trip estimates to be highly variable, thus hampering recovery from packet losses.

3.5 TCP/IP over ATM

So far Internet (or TCP/IP) traffic has been provided with the *best effort* service. However, such a service may not meet the needs of timing-sensitive applications such as real-time interactive audio/video, which may require preferential treatment. With multiple service classes and a rich set of traffic management functions, ATM is expected to play an important role in providing guarantee or quality of service for such demanding applications. However, assuring quality of service (QoS) solely at the ATM layer does not guarantee end-to-end delivery of QoS. Apart from the fact that ATM networks are still not ubiquitous, the problem stems from the difficulty in understanding the interaction between the various ATM QoS delivery schemes and the higher layers such as TCP and IP. IP overlaid over ATM, does not provide packet differentiation and tends to multiplex traffic from several connections over a single ATM virtual circuit (VC). This may result in loss of certain QoS aspects provided by ATM over a VC.

The fact that TCP is not a link layer aware protocol, that is, it does not know to adapt its control mechanisms based on link layer characteristics further exacerbates the problem. For example, if TCP does not respond to the ATM available bit rate (ABR) resource management (RM) cell in time by decreasing its transmission rate, a connection may lose the QoS guarantees that may have been provided by ATM. Further, the congestion control mechanisms used by TCP also may have undesirable effect on the provision of predictable services of ATM. Research in dynamics of TCP over ATM [RF95] shows that TCP window size, receiver buffer size and TCP packet size (influenced by the large maximum transfer unit at the ATM interface) have a dramatic impact on the overall system throughput. In addition, the feedback mechanisms in TCP and ATM and their interaction pose a challenge of understanding the complex buffer management and scheduling schemes required.

3.6 IP QoS: IntServ versus DiffServ

Quality of service (QoS) can be described as a set of technologies that enables network administrators or service providers to manage the effects of congestion on application traffic by using network resources optimally, rather than continually adding capacity [Ber00]. From a user's perspective, it could be described as the degree of satisfaction he derives from the service he is receiving. Alternately, QoS could be defined in terms of specific performance metrics such as – end-to-end delay, delay variation or jitter, packet loss ratio, throughput etc.

3.6.1 Need for QoS

The explosive growth in the Internet over the last decade, along with simultaneous proliferation of new *bandwidth-hungry* applications has been putting compelling demands on today's networking world. Despite of the astounding rate at which network capacity is increasing, thanks to high transmission technologies such as optical fibers, xDSL, cable modems etc., today, network users still find themselves contending with congested networks. This has led to the development of what is today known as the *quality of service (QoS)* technologies.

3.6.2 Traffic Engineering Models for QoS Provisioning

Based on the various performance metrics mentioned above, the following two types of traffic engineering models have been proposed for QoS provisioning [QOS]:

- **Reservation-based:** In this model, resources for different types of traffic are explicitly identified and reserved. Network nodes classify incoming packets and use the reservations to provide differentiated services. Typically, a dynamic resource reservation set up protocol is used, in conjunction with admission control, to set up reservations. Further, the nodes use intelligent processing (e.g., RED, EPD) and queuing mechanisms (e.g., WFQ) to service packets.
- **Reservation-less:** In this model, no resources are explicitly reserved. Instead, traffic is differentiated into a set of classes, and network nodes provide priority-based treatment of these classes. It may still be necessary to control the amount of traffic in a given class, allowed entry into the network, to preserve the quality of service being provided to other packets of the same class.

Based on these two broad categories of traffic engineering models, the Internet Engineering Task Force [IET] has proposed two types of mechanisms for providing IP QoS – namely the Integrated Services Architecture (IntServ) [Int] and the Differentiated Services Framework (DiffServ) [Dif], which are briefly discussed in Sections 3.6.3 and 3.6.4 respectively.

3.6.3 Integrated Services Architecture (IntServ)

The "Integrated Services" model for the Internet was proposed in the early 1990s. This effort, which became known as IntServ, encompasses various types of service, including best-effort and real-time service, and allows for bandwidth reservations. It was concluded that using IntServ, routers must be capable of reserving resources in order to provide special QoS for specific user packet streams or flows.

To reserve resources, IETF defined the resource reservation protocol (RSVP), a network-control protocol that lets IP-based applications request QoS treatment for their data flows. When an application requests a specific QoS, RSVP is the signaling mechanism that delivers the request to each router or other layer-3 device (such as layer-3 switches) along the path of the requested service. With RSVP, an application can request a specific amount of bandwidth and a guaranteed service level.

Interestingly, RSVP is receiver-oriented, meaning that the receiving host is responsible for requesting resource reservations and transmitting these, back to the sender. This way, a receiver can request a QoS that is tailored to its particular requirements. By being receiver-oriented, RSVP avoids the problem of a sending host transmitting a higher-quality video stream than the receiver can handle. Figure 3.3 shows typical operation of RSVP.

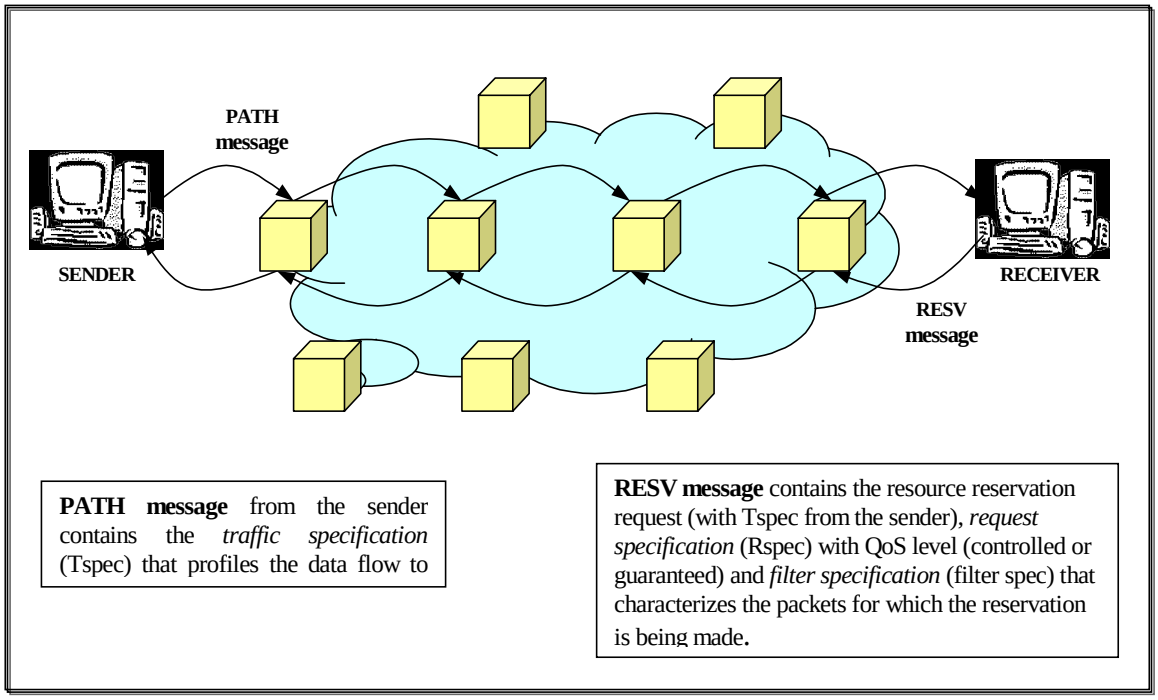

Figure 3.3 Typical operation of RSVP

However, one of the major drawbacks of RSVP is that it requires each router to reserve the resources, such as queuing capacity and memory, necessary to fulfill the request. RSVP's need to maintain state information for each such user means that it lacks scalability and hence is of limited use on the Internet. For example, it would be logistically impossible for routers to track the tens of thousands of traffic flows that RSVP would require over a wide area network.

Consequently, the IETF has issued guidelines on deploying RSVP and it has become generally accepted that RSVP is likely to play a role in enterprise networks, but not the core of the Internet.

3.6.4 Differentiated Services Architecture (DiffServ)

The IETF's DiffServ work is largely based on the *type-of-service (ToS) and precedence* field within the IPv4 header. The IP precedence field was first defined in IP to indicate a certain way by which a given IP datagram should be queued at routers or other network devices over an end-to-end connection. For example, high-priority traffic, as indicated by a specific value in the IP precedence field, should be put into a router's high-priority queue and forwarded before packets in a lower-priority queue.

DiffServ defines the first six bits of the ToS and precedence field as the D*iffServ codepoint (DSCP)* field as shown in Figure 3.4.

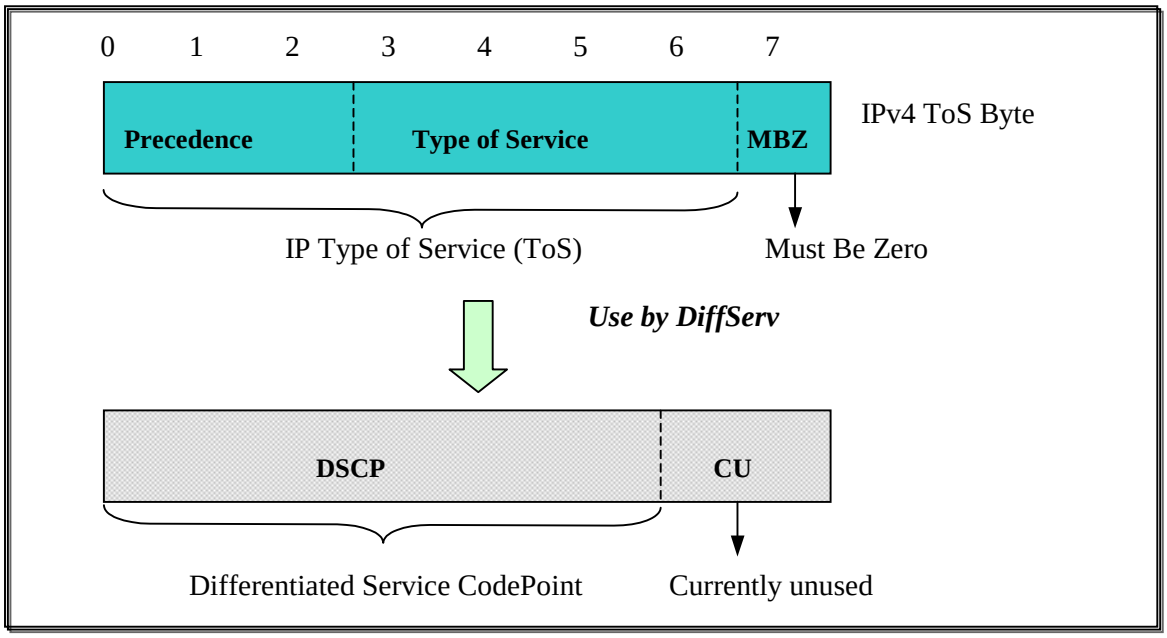

Figure 3.4 Use of IPv4 ToS and Precedence fields as DSCP in DiffServ

Hosts or routers sending traffic into a DiffServ network mark each of the transmitted packets with a certain DSCP value. Routers within the DiffServ network use the DSCP value to classify packets and apply specific queuing behavior known as *per-hop behavior (PHB)* based on the results of the classification. Traffic from many flows having similar QoS requirements is marked with the same DSCP, thus aggregating the various traffic flows to a common queue or scheduling behavior [Ber00].

Now, as compared to RSVP/IntServ, the distinguishing feature of DiffServ is its scalability. This stems from the main difference between the two mechanisms: RSVP/IntServ networks implement per-conversation traffic handling mechanism, while DiffServ uses aggregate traffic handling mechanism, which involves grouping several traffic flows together. Thus, DiffServ requires significantly less state and processing power in network nodes as compared to its counterpart. This is advantageous in large networks where routers or network nodes have to handle an enormously large number of conversations or traffic flows. However, the tradeoff of using aggregate traffic handling mechanism is that the QoS enjoyed by each conversation is dependent on the behavior of other conversations or traffic flows with which it is aggregated.

A reasonable compromise would be to perform resource reservation at the edge networks using RSVP and achieve service differentiation in the core network using DiffServ. This approach is currently being pursued.

3.7 Point-to-Point Protocol (PPP)

With the rapid growth of data networks, various organizations and vendors developed a number of network-layer or layer-3 protocols (e.g. IP, IPX, AppleTalk etc.). With these different protocols being developed, machines communicating with each other needed to know which network layer protocols were available to support a certain user application. Moreover, in some situations, it was desirable to negotiate options to be used during a session between two machines. These operations were occasionally performed using proprietary protocols. However, even worse, due to the lack of any common means, many times these negotiations were not performed. This generally led to wastage of resources such as bandwidth, IP address space etc. Further, until the advent of point-to-point protocol (PPP) [Sim94], the industry did not have a standard means to define a point-to-point encapsulation protocol [Bla99]. The development of PPP solved these two problems.

PPP provides methods for negotiating a wide variety of operations and options, as well as a standard means to encapsulate the layer- 3 protocols. Today, PPP is used to encapsulate multi-protocol datagrams and to transmit them over a serial communication link.

3.7.1 PPP Operation

The PPP operation is summarized in the state diagram shown in Figure 3.5. It comprises of three main component-protocol phases [Sim94]:

- **Link Control Protocol (LCP) phase:** In the initial or starting state, the LCP automaton detects whether the physical link is ready to use. In event of an *UP* event (indicating that the physical layer is ready for transmission), the LCP at a host establishes connection with its peer through exchange of *Configure* packets. Further, various negotiations such as the maximum packet length, use of compression, authentication protocol to be used etc., take place between the peers during the link establishment phase.

  LCP is also used to close a link at any given time. The termination of a link may occur due to loss of carrier, authentication failure, link quality failure, expiration of idle-period timer or administrative closing of the link. When PPP is closing a link, it informs the network-layer protocols so that they may take appropriate action.

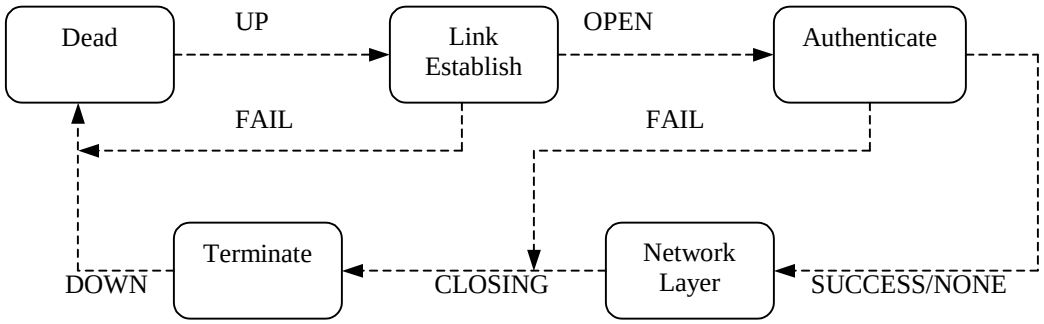

Figure 3.5 PPP operation state diagram [Sim94]

- **Authentication Protocol phase:** Although authentication is not mandatory, it may be desirable to require a peer to authenticate itself before allowing any exchange of network-layer protocol packets. If an implementation at a host desires authentication, then it is indicated to the peer machine during the link establishment phase. However, it is important that the implementation not allow the link quality determination packets (during the LCP phase) to delay the authentication indefinitely. Hence, authentication may be allowed to occur simultaneously along with the link quality determination. The implementation should also ensure that no network-layer packets are allowed to flow before authentication is complete. The various authentication protocols commonly used are [Car98] Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP) and its variant Microsoft-CHAP (MS-CHAP).
- **Network-Layer Control Protocol (NCP) phase:** After PPP has completed the earlier phases (link establishment and authentication) each network-layer protocol is to be configured

individually by appropriate network control protocols (NCPs). Each NCP may be opened or closed at any given time.

3.7.2 PPP over ATM

The PPP over ATM architecture has been accepted by the industry as a standard for remote access using DSL technologies. The basic idea behind this architecture is that, once an ATM layer connection is established between the customer premises and the service provider network, the session setup and termination at the network level can be done using PPP [Kwo99]. Further, we can also take advantage of the various PPP functionalities such as authentication, layer-3 auto-configuration (such as IP address assignment, domain name auto-configuration etc.), multiple layer-3 protocol support, encryption, and compression, which can be now delivered over ATM.

For using PPP over the ATM adaptation layer 5 (AAL5), two types of encapsulation techniques have been proposed [GKL+98], namely the logic link control (LLC) encapsulated PPP and virtual circuit (VC) multiplexed PPP. The basic idea is to use AAL 5 as a framing technique, unlike most other implementations of PPP (without ATM), which use HDLC-like framing [Sim(1)94].

- **Logic Link Control (LLC) encapsulated PPP:** The LLC encapsulated PPP [GKL+98] frame format is as shown in Figure 3.6. The LLC header consists of 1 byte each for Destination Service Access Point (DSAP), Source Service Access Point (SSAP) and Frame Type (FT) fields. This is followed by 1 byte of Network Layer Protocol Identifier (NLPID) field representing PPP. For transmitting IP datagrams, the PPP Protocol Identifier (PI) field occupies 1 or 2 bytes (e.g. 2 bytes for IP as the network layer). The IP datagrams are placed in the PPP Information Field (IF). In addition, padding may be done up to the negotiated maximum receive unit (MRU) for a particular PPP session if required. The encapsulation thus adds 6 bytes of overhead for each IP datagram before transmission.

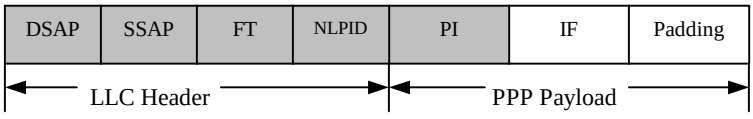

Figure 3.6 LLC encapsulated PPP frame format

- **VC-multiplexed PPP:** The VC-multiplexed PPP frame format that constitutes the common part convergence sub-layer (CPCS) packet data unit (PDU) payload corresponds to the PPP payload part of the PPP frame format shown in Figure 3.6.

The choice of either of these two methods is driven by system requirements. Although VC-multiplexing results in a slightly lesser overhead than LLC-encapsulated PPP, it sets up a separate VC for each protocol. On the other hand, LLC-encapsulated PPP can support multiple protocols on a single VC (albeit with slightly more overhead) and hence has gained more popularity. It is typically used for encapsulating IP datagrams over AAL5 in the ADSL network architecture considered for this research.

3.8 Layer-2 Tunneling Protocol (L2TP)

The basic PPP functionalities defined in Section 3.7 allow encapsulating and transporting multi-protocol packets across layer-2 point-to-point links. Typically, a user sets up a connection with a network access server (NAS) using some access technique such as the narrowband modem dial-up or ADSL, and then runs the PPP session over it. In such a case, both the layer-two and PPP termination point reside on the same device. However, due to the varied utilities of PPP resulting in universal deployment of PPP in the Internet, there is a need to extend the PPP functionalities over multiple links and multiple networks. This may also require the layer-two and PPP end-points to be on different devices (e.g. one on a local concentrator and another at the NAS respectively). This is done using what are commonly known as “tunneling” protocols.

Layer-2 tunneling protocol (L2TP) [TVR+99] is one such tunneling protocol, which elevates the visibility of PPP procedures across multiple links and networks. L2TP encompasses the attributes of two proprietary tunneling protocols namely: the point-to-point tunneling protocol (PPTP) developed by Microsoft and layer-two forwarding protocol (L2FP) developed by Cisco [Bla99]. For a DSL access network, L2TP allows the user to have a layer-two connection to an access concentrator, typically a digital subscriber line access multiplexer (DSLAM), and the concentrator then tunnels the PPP frames to the NAS over a shared infrastructure such as frame relay circuit, ATM backbone network or the Internet.

A typical L2TP configuration for DSL networks is as shown in the Figure 3.7. It consists of the following main components [TVR+99]:

- **L2TP Access Concentrator (LAC):** The LAC acts as one side of an L2TP tunnel and is peer to the other tunnel endpoint, which is the L2TP network server (LNS) discussed below. The LAC forwards the packets received from the remote (client) system to the LNS using L2TP

tunneling. The connection between the LAC and the remote system is either a local connection (for a client residing at the LAC) or a PPP link (for a remote host). The LAC de-tunnels the packets sent by the LNS before forwarding them to the remote system.

- **L2TP Network Server (LNS):** The LNS acts as an endpoint for the L2TP tunnel on one side and is peer to the LAC. The LNS is the logical termination point of a PPP session that is being tunneled from the LAC.
- **L2TP Tunnel:** An L2TP tunnel exists between the LAC and LNS. It consists of a control connection and zero or more ongoing L2TP sessions. The tunnel carries encapsulated PPP frames and control messages between the LAC and the LNS. Each L2TP tunnel is assigned an identifier (ID), which has local significance only at each tunnel endpoint. In other words, every tunnel may be given different tunnel Ids by each end of the tunnel. Tunnel Ids are selected and exchanged between the LAC and LNS as assigned tunnel ID attribute value pairs (AVPs) during creation of a tunnel. A single L2TP tunnel can be used to carry multiple PPP sessions. Such L2TP sessions, which exist in a tunnel are also assigned IDs and have local significance as that of the tunnel IDs. Further, the tunnel ID and session ID in each message is that of the intended recipient rather than the sender.

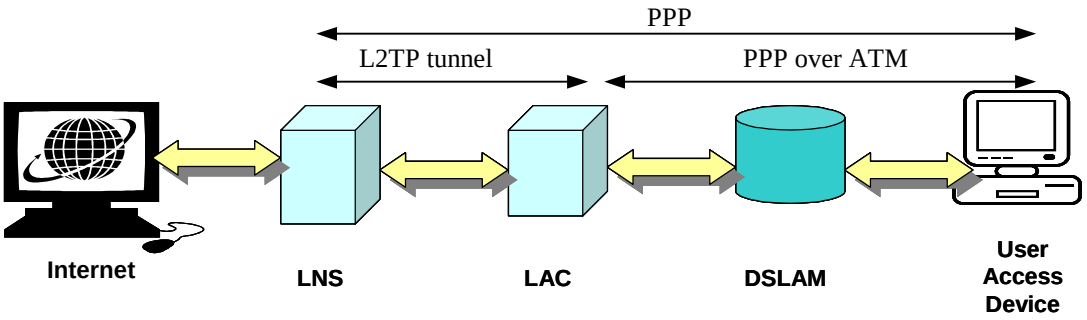

Figure 3.7 Typical L2TP system configuration

To establish communication between the two endpoints of a tunnel, L2TP defines two types of messages or packet formats – the control messages and the data messages. As the names suggest, the control packets are used to setup, maintain and tear the L2TP tunnels, while the data messages are used for encapsulating PPP packets into an L2TP tunnel and correctly identifying them for transport between two appropriate nodes.

3.8.1 L2TP Access Aggregation Model for ADSL Networks

L2TP access aggregation model shown in Figure 3.8 is one of the models being promoted for deployment in ADSL networks [ADS98, ALH+99]. This model depicts the structure of the

underlying protocol stacks for the network configuration shown earlier in Figure 3.7 and is a pictorial representation of our discussion of the various protocols and their functionalities (when used together in an ADSL network). We will be considering this protocol stack model as the basis for our simulation study.

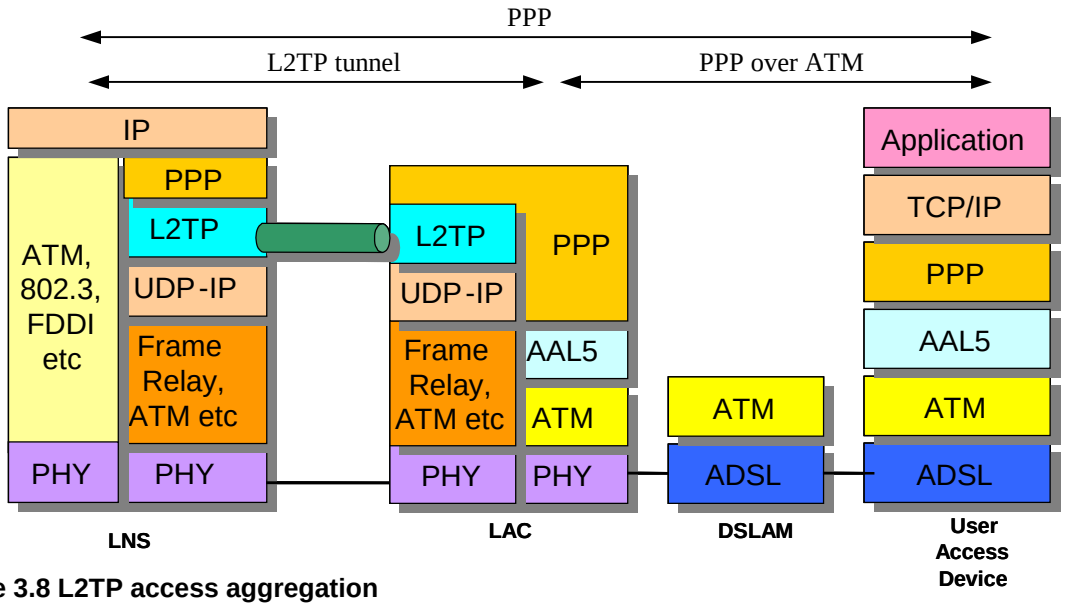

Figure 3.8 L2TP access aggregation

3.10 Chapter Summary

This chapter provided us with an overview of the different network protocols typically found in an end-to-end ADSL network architecture. The first two sections provided insight into the attributes of ATM and the TCP/IP protocol suite. Sections 3.3, 3.4 and 3.5 detailed the various performance issues that arise as a result of the interoperation of these protocols when used over ADSL. Further, in Section 3.6, we briefly enumerated the two proposed IP QoS architectures: *IntServ* and *DiffServ*. Lastly, the chapter introduced the point-to-point protocol (PPP) and layer-2 tunneling protocol (L2TP), which facilitate end-to-end seamless connectivity to DSL customers.

CHAPTER 4. Simulation Models and Methodology

This chapter describes the different simulation models and scenarios used as a part of this research. We first take a brief look at the OPNET Modeler™ network simulation tool in Section 4.1. In Section 4.2 we describe the end-to-end model used to simulate a typical ADSL network configuration. The subsequent sections, 4.3 through 4.7, provide a detailed discussion of the various simulation scenarios and the different process models developed and implemented as a part of these scenarios. Finally, in Section 4.8, we take a look at certain system parameters (application traffic models, key protocol attribute settings and performance metrics) considered for the various simulations.

4.1 OPNET Modeler™

OPNET Modeler™, a product of the OPNET Technologies Inc. [OPN] was used for system modeling and for performing detailed simulations. The network simulation environment in OPNET is organized in a hierarchical manner consisting of three structural levels: the *network model* level, the *node model* level and the *process model* level. At a given level in the hierarchy, a model is configured or defined in terms of the models at the next lower layer. This means that a network model will consist of a group of node models arranged and connected in a certain way. Likewise, a node model consists of one or more process models connected in a certain manner. The node models typically represent the protocol stack underlying the network devices. The process models in turn are finite-state machines (FSMs) built using C/C++ and Proto-C (which uses OPNET-defined simulation kernels or functions). Thus, each model can be imagined to be an object, each with a set of attributes that can be modified. Hence, OPNET can be described as being object-oriented in its approach to modeling components within simulations.

Further, the OPNET Modeler™ includes an extensive library of built-in models and functions, many of which have been utilized in building the simulation models for this study. OPNET also allows the user to collect data or results by selecting appropriate statistics before each simulation run. With its graphical user interface (GUI), OPNET facilitates the presentation of the models developed or results gathered by the user. All these attributes make OPNET a very powerful and flexible network simulation tool, resulting in its extensive use both in academic as well as industrial environment all over the world.

4.1.1 Word of Caution!

We would like to spread a *word of caution* to current and potential OPNET users, based on certain *problems* that we encountered during this research exercise.

We initially developed our process models in OPNET Modeler version 6.0. Keeping in mind the long-term goals of this research project (which will continue beyond this thesis), version 7.0 was adopted in the last quarter of the project, to take advantage of the newly introduced IP quality of service (QoS) models.

OPNET Modeler version 7.0 allows a user to use his custom models[2] from version 6.0 on its GUI by using the /models/*compat_std* (compatible standard) directory instead of the default *std* directory. The user only needs to reconfigure the node models with his custom process models (and hence reconfigure the network models). However, this is possible only if the incompatibility is up to the network or node model level. Due to enhancements in certain newer (version 7.0) models, the incompatibility between the two OPNET Modeler versions goes all the way down to the process model level in case of certain protocol model suites! For example considerable changes have been made to the process models in both the ATM and IP model suites in OPNET version 7.0 as compared to version 6.0.

Many of the custom process models developed as a part of this research resulted from modifying some of the existing process models mainly from OPNET's ATM Model Suite (AMS) and IP model suite. Thus, *usefulness* of the various process models developed (by modifying the already existing OPNET models) tended to be *version-bound* and in effect *time-bound*. The cost paid was in terms of refurbishing of the models developed in OPNET Modeler version 6.0 to make them compatible and useful in version 7.0, considerably delaying the completion of this research.

4.2 End-to-end ADSL Network Simulation Model

The end-to-end network set-up used in most of the simulation scenarios is shown in Figure 4.1. The *Client* and the customer premises equipment (*CPE*), representing a small office/home office (SOHO) environment, along with the DSL access multiplexer (*DSLAM*) form the ADSL access portion of the network. ADSL (between the *CPE* and the *DSLAM*) is modeled using a single simplex link in each direction with different raw bandwidths. Among the ADSL technologies, G.Lite [ADS, Kwo99] is one of the widely deployed ADSL standards. Typical

[2] By custom models we mean those developed by modifying the already existing OPNET models, rather than totally new models implemented by a user.

values of the upstream and downstream bandwidths for G.Lite are 512 kb/s and 1.5 Mb/s respectively, over an 18,000 ft. access link. However, due to the rate-adaptive characteristic of G.Lite, or for operation beyond 18,000 ft., lower upstream speeds are common. Thus, we used a 96 kb/s upstream channel and T1 (1.544 Mb/s) downstream channel for most of the simulations. For illustrative purposes, we also ran some simulations using different raw bandwidths of 64 kb/s, 48 kb/s, 28.8 kb/s and 16 kb/s for the upstream. However, it must be noted that, as far as the basic *asymmetry problem* is concerned our focus is on the normalized asymmetry (k) defined in Section 3.4, rather than the raw bandwidth asymmetry. The links along the backbone network (*DSLAM* ←→ *Gateway* ←→ *Internet Gateway* ←→ *Servers*) are high-speed (OC3/DS3) links. The Internet cloud is in general a representative of the server side of the network.

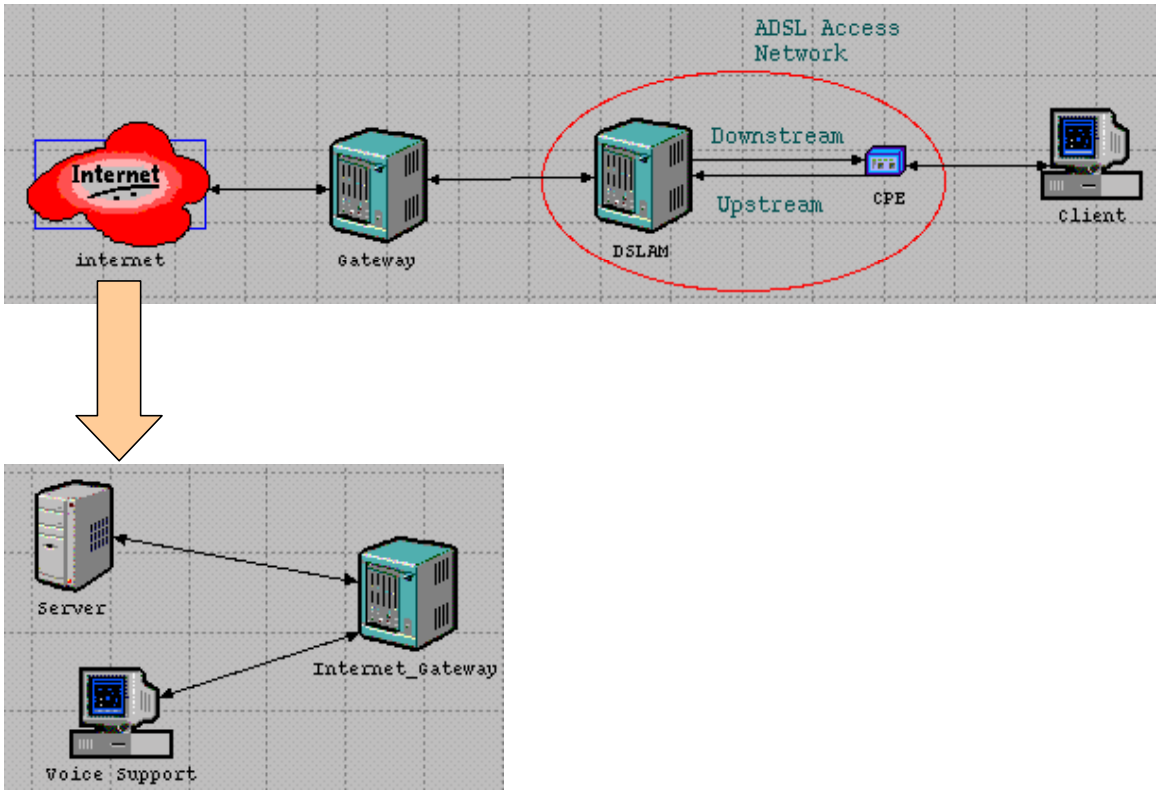

Figure 4.1 Simulation set-up for end-to-end ADSL network configuration

The various network devices seen in Figure 4.1 were configured using some of the already existing OPNET models in case of certain scenarios, and by using custom models that were developed and implemented as a part of this study for certain other scenarios. The details are provided in subsequent sections, which describe the various simulation scenarios considered.

4.3 TCP/IP over ADSL

The *TCP/IP over ADSL* scenario is used mainly to study the effects of bandwidth asymmetry on the performance of TCP. This helps in benchmarking a baseline scenario before we characterize TCP/IP performance in presence of other network protocols such as PPP, ATM etc. Also we validate our baseline simulation model and emphasize the significance of normalized asymmetry (*k*). As a part of this scenario, both unidirectional (data transfer along the downstream) and bi-directional (data transfer along both the downstream and the upstream) traffic has been considered.

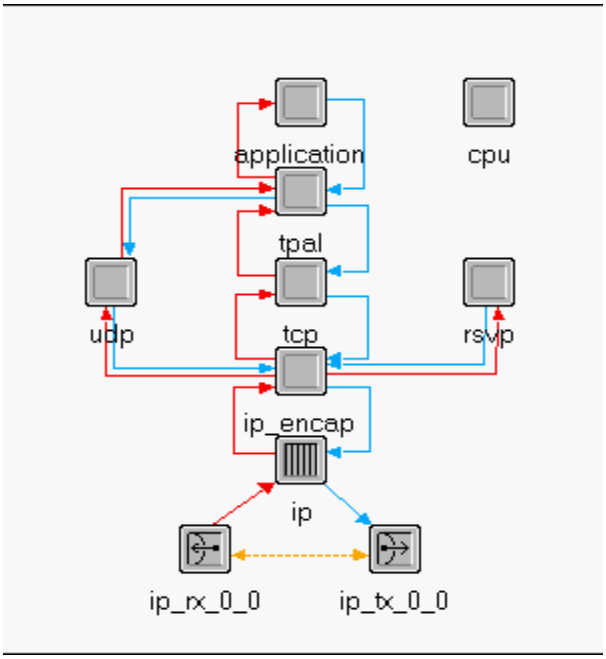

Figure 4.2 Protocol stack structure (node model) for a TCP/IP server or client

The node model shown in Figure 4.2 represents the protocol stack for a server or client available from the *ppp* suite of OPNET models. Please note that the *ppp* model suite in OPNET actually represents serial line IP (SLIP) and not the point-to-point protocol (PPP) that has been implemented as a part of this research project. These models were modified to obtain *ppp_server_ete_delay_adv* and *ppp_wkstn_ete_delay_adv* node models, which respectively represent the *Client* and the *Server* in this scenario. These models are distinct from the already existing OPNET models since these provide a probe to obtain end-to-end packet delay (and also specifically TCP ACK delay, which is one of the important metrics considered), a statistic not readily available in the existing OPNET models. The probe was implemented using certain OPNET defined simulation kernels, which allow the user to capture the creation time of a packet

as well as the current simulation time. The already existing process *ip_rte_v4* (which represents the *ip* layer in the stack structure shown in Figure 4.2) was modified to implement the end-to-end per packet delay statistic.

4.4 TCP/IP over ATM over ADSL

As mentioned in Section 3.2.1, using ATM as the data link layer provides some distinct advantages. However it is noteworthy that the TCP/IP over ATM approach has some drawbacks mainly in terms of *overhead*. The segmentation of IP datagrams at AAL-ATM layers results in additional overhead and increases the load over the slower upstream. The objective of this scenario is to investigate whether this additional overhead causes further performance degradation as compared to the basic *asymmetry problem*. The node model shown in Figure 4.3 represents a server/client model available in the OPNET ATM model suite (AMS). These models were used to derive custom node models *atm_server_ete_delay_adv* and *atm_wkstn_ete_delay_adv* respectively for the *Client* and the *Server* to obtain end-to-end packet delay (and TCP ACK delay) along with other existing statistics.

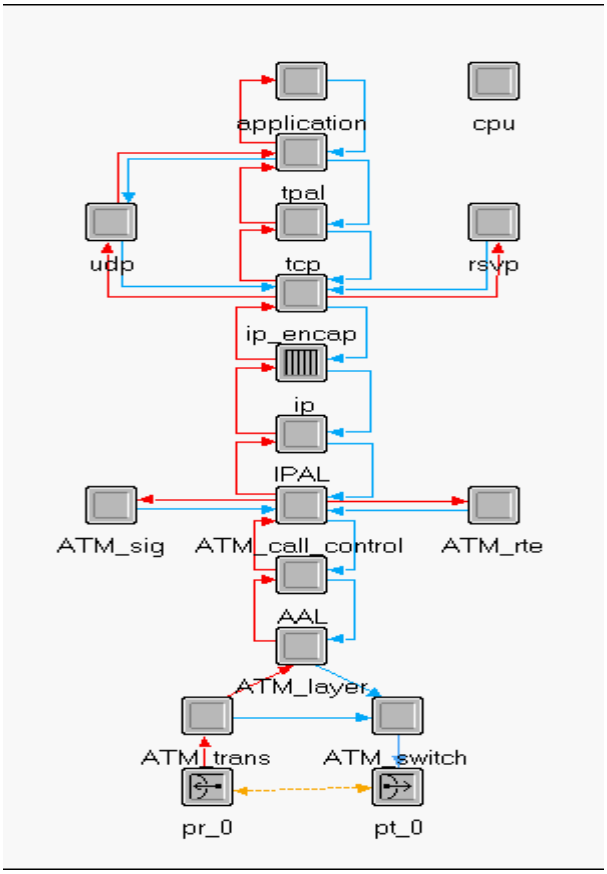

Figure 4.3 Protocol stack structure (node model) for an ATM server or client

4.5 TCP/IP over PPP over ATM over ADSL

From the viewpoint of quantifying system performance metrics such as throughput and delay, we focused mainly on the effect of the PPP encapsulation of IP datagrams before they get segmented at the ATM adaptation layer (AAL5). The encapsulation of the IP datagrams is done in the logic link control (LLC) encapsulated PPP [GKL+98] frame format as was shown in Figure 3.6.

The LLC header consists of 1 byte each for Destination Service Access Point (DSAP), Source Service Access Point (SSAP) and Frame Type (FT) fields. This is followed by 1 byte of Network Layer Protocol Identifier (NLPID) field representing PPP. For transmitting IP datagrams, the PPP Protocol Identifier (PI) field occupies 2 bytes (for IP as the network layer). The IP datagrams are placed in the PPP Information Field (IF). The encapsulation thus adds 6 bytes of overhead for each IP datagram before transmission. Using the *TCP/IP over PPP over ATM over ADSL* scenario, we investigate whether this additional overhead has any adverse effect on TCP performance.

The PPP encapsulation model over AAL 5 was implemented in OPNET by modifying the existing AAL layer process model. The LLC type PPP frame format mentioned above was implemented using the *Packet Editor* in OPNET Modeler. The packet editor basically allows the user to define custom packet formats including the individual fields and their sizes (in number of bits) within the packet. Each of these fields can be defined as different data types such as *integer* (if only a dummy field occupying certain number of bits is to be defined), *structure* (if certain fields are to be accessed and processed during simulation execution), *packet* (if one packet is to be encapsulated into another packet format) etc. Since we are mainly interested in encapsulation of IP datagrams in PPP frame format, we define a packet format *ppp_pk_fmt*, where in all the fields except the IF field (shown above) are defined as data type *integer* with appropriate lengths (in bits). The IF field is defined as data type *packet*, wherein the IP datagram will be encapsulated.

The AAL process model in OPNET is dynamic in nature and consists of two processes, which form a parent-child process hierarchy as shown in Figure 4.4. The parent process *ams_aal_disp_v3* invokes the *ams_aal5_conn_v3* process. The *ams_aal_disp_v3* process mainly deals with creating and invoking the appropriate underlying child process to handle the new connections, incoming signals and data. Since we specifically use AAL5, the child process model of concern is the *ams_aal5_conn_v3* process. It specifically deals with the functionalities of the AAL5 such as segmentation of incoming network layer packets into AAL5 CS-PDUs and

sending them to the ATM layer, buffering and re-assembly of incoming ATM cells into network layer packets and sending them to the network layer etc.

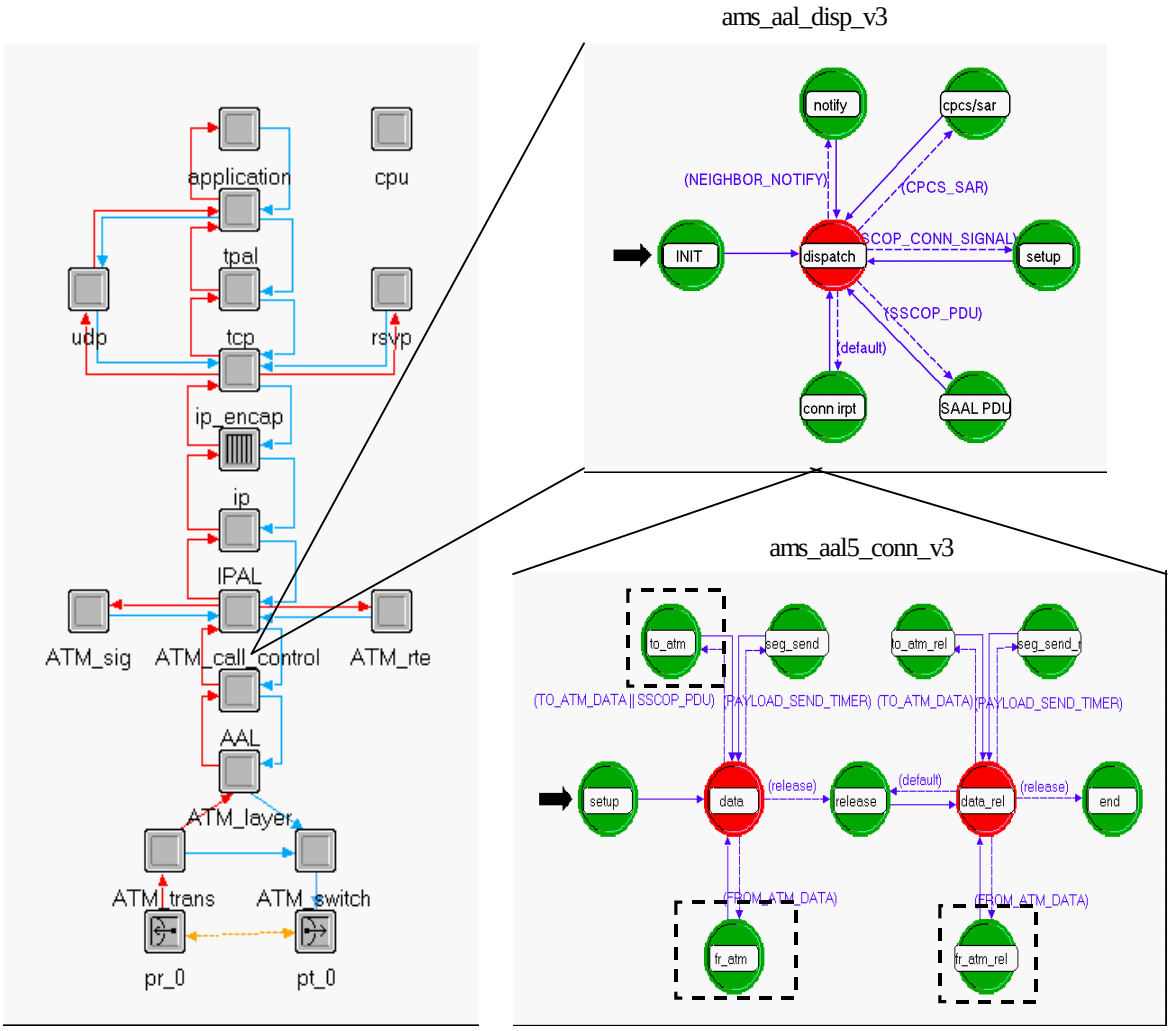

Figure 4.4 Parent-child processes hierarchy in AAL simulation model implementation

Both these processes were modified to implement custom process models *ams_ksp_aal_disp_v3* and *ams_aal5_ksp_conn_v3* respectively. The LLC type PPP encapsulation of IP datagrams (before sending them to the ATM layer) has been implemented in the *to_atm* state of the *ams_aal5_ksp_conn_v3* process, while the reassembly of ATM cells into AAL 5 CS-PDU and then the decapsulation of IP datagram from the PPP frame is implemented in the *fr_atm* state of the *ams_aal5_ksp_conn_v3* process model. Both these states are marked with dotted lines in Figure 4.4. This modified AAL5 process (*ams_aal5_ksp_conn*_v3) is then defined in the appropriate function (invoking the child process) within the *function block (FB)* (where all

the functions used in a process model are defined) of the *ams_ksp_aal_disp_v3* process model. The detailed code is provided in Appendix A.

4.6 Proposed Solutions

In our discussion about the effects of asymmetry on TCP performance in the earlier chapter (Section 3.4), we saw that the slower upstream is the primary bottleneck to the TCP throughput over the downstream. Any congestion over the slower upstream will disrupt the flow of TCP acknowledgements (*ACKs*), in turn degrading the downstream TCP throughput. We also discussed the upper bound on the downstream TCP throughput (for unidirectional data traffic) determined by the *normalized asymmetry (k)*, which implied that for $k > 1$ if there is more than one ACK for every k data packets, then the slower upstream will get saturated before the faster downstream, possibly limiting throughput over the downstream.

Several researchers [BPK97, Jac90, KVR98, ZT99] have proposed schemes which help reduce the TCP ACK traffic over the upstream in effect improving TCP performance over asymmetric links, albeit for unidirectional data transfer in most cases. Most of these schemes have been tested for TCP/IP implemented directly over asymmetric links. However, we implement these schemes (and in some cases extend the techniques) and compare their effectiveness in presence of the new protocol stack (TCP/IP over PPP over ATM) typically found in an ADSL network. Further, we also account for the performance of certain queuing/scheduling schemes in presence of bi-directional traffic. In this section we discuss the OPNET models used or developed in order to implement these varied schemes.

4.6.1 Delayed TCP ACK Technique

The basic idea underlying this technique is to delay the TCP ACKs being sent in response to the received TCP segments. This in effect reduces the number of TCP ACKs transmitted over the upstream, alleviating congestion. OPNET allows user control over the amount of time TCP waits after a segment is received, before it sends out the next TCP ACK. This is achieved by appropriate setting of the attribute *Maximum TCP ACK delay* at the *network model* level. The improvement in TCP performance on using this technique has been analyzed in [ZT99]. However, we further investigate the limitations involved in using such a scheme to alleviate the *asymmetry problem*.

4.6.2 TCP/IP Header Compression

This scheme is based on the CSLIP technique proposed in [Jac90]. Typically at the IP level, TCP ACKs are transmitted as 40 byte packets (20 bytes each for TCP and IP headers). This scheme involves compressing the TCP and IP headers in effect reducing the size of the TCP ACKs and hence may aid in reducing the load over the slower upstream. OPNET allows users to set the TCP/IP header compression ratio at the *network model* level.

4.6.3 Smart ACK Dropper and ACK Regenerator (SAD-AR)

The *SAD-AR* scheme [KSK99] is an extension of the ACK filtering technique proposed in [BPK97]. However, certain distinct features of the *SAD-AR* scheme allow it to address the shortcomings or problems encountered by the ACK filtering technique.

The ACK filtering scheme [BPK97] is a gateway-based technique, which basically involves decreasing the number of TCP ACKs being sent over the slower upstream by taking advantage of the fact that TCP ACKs are cumulative. When a new acknowledgement arrives from the receiver, the sender scans its queue for a previous ACK from the same flow. If it finds ACKs belonging to the same flow, it clears the queue of some fraction (possibly all) of older acknowledgements, and enqueues the latest acknowledgement at the tail of the queue. The goal is partially to free some space in the queue for other data packets and ACKs, while decreasing the ACK load over the constrained upstream. There is no per-flow state, or multiple queues, but there is overhead in linearly searching the queue for ACKs. This in turn limits the performance of this scheme for queues with large number of packets. Further, although dropping of ACKs reduces traffic over the upstream, it results in a bursty source, since the TCP source now receives one ACK for every (n+1) data packets and hence sends n+1 packets for every ACK it receives (n being the number of ACKs dropped). This may cause unwarranted packet losses due to buffer overflows in the forward direction. Also, for TCP Reno version, if duplicate acknowledgements get dropped during ACK filtering, then it will cause the TCP fast retransmit-recovery scheme to breakdown. To take care of the burstiness and the fast retransmission problem, Balakrishnan et al., [BPK97] propose a sender-adaptation scheme. However, this leads to making changes in the end systems in the network, which again limits successful (or popular) deployment of such a scheme! Further the authors report that in some cases ACK filtering alone provides either insufficient compensation for asymmetry or even degrades performance, necessitating the use of other techniques like ACK congestion control (ACC, which involves TCP modification at the end systems) and/or ACKs-first (priority) scheduling for compensation. This is mainly because of the delay in sending out ACK information, which might interact with TCP timeouts. We now briefly describe the *SAD* and

AR algorithms and see how they meet the shortcomings of the ACK filtering and other associated techniques.

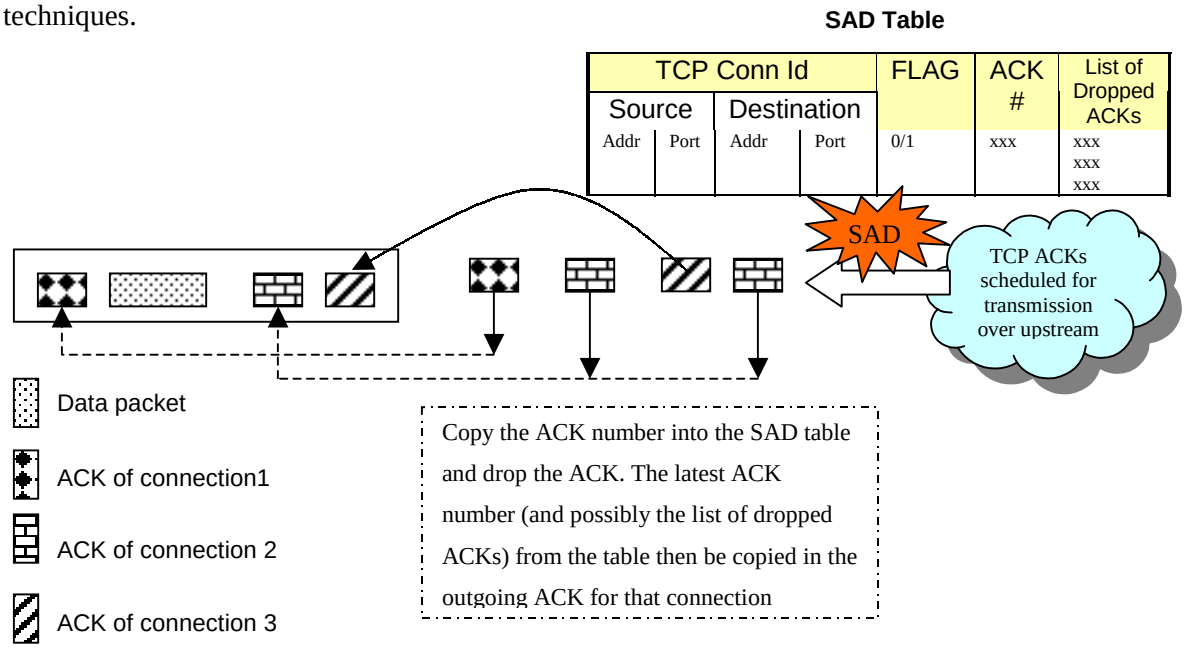

Figure 4.5 Smart ACK Dropper (SAD)

The *Smart ACK Dropper* and the *ACK Regenerator* algorithms would be typically implemented at the *CPE* and the *DSLAM* (shown in Figure 4.1) respectively in the ADSL access network. No changes in the end systems are required. The basic functioning of the *SAD* algorithm is as follows. A FIFO queue is assumed for the TCP ACKs to be sent over the upstream. At any given time only a single TCP ACK per flow is maintained in the queue. This is achieved by maintaining a TCP connection table or a hash table used to store information pertaining to each flow or TCP connection. This per-flow information includes the following: TCP connection identifier (*IP source and destination addresses; TCP source and destination ports*), a single bit *FLAG* to indicate whether a TCP ACK belonging to a particular flow is already present in the queue, *TCP ACK number* field to store the ACK number of the TCP ACK currently in the queue and a *List* of TCP ACK numbers of the ACKs that are dropped.

Any new connection set up by TCP is registered in the hash table. When a TCP ACK arrives and the *FLAG* bit is zero (no ACK belonging to that flow is present in the queue), we enqueue the ACK, set the *FLAG* bit and copy the ACK number into the table. If the *FLAG* bit is already set upon arrival of a non-duplicate ACK, then we update the per-flow ACK number information to this latest value, and drop the ACK. When an ACK is dequeued for transmission, the latest value

of ACK number from the hash table is copied to the header and the *FLAG* bit in the hash table is reset to zero.

Now as far as the *ACK Regenerator (AR)* is concerned, we suggest three approaches. One option is to copy the list of ACK numbers of the dropped ACKs (maintained in the SAD table) into the IP header of the outgoing TCP ACK (possibly using a header extension!) as shown in Figure 4.5, or the list may be copied in small packets used for in-band signaling between the CPE and the DSLAM. This information is later on used by the *ACK Regenerator* scheme at the other end of the access network to regenerate the ACKs dropped by the SAD scheme. Alternately, we do not maintain the list of dropped ACKs in the SAD table. Instead, we can have the DSLAM regenerate and interpolate appropriate ACKs based on the ACK numbers of two consecutive ACKs arriving at the DSLAM. These approaches are further discussed in the Section 6.3 *Suggestions for Ongoing and Future Work.*

We will now see how the *SAD-AR* schemes are able to make up for the shortcomings of the ACK filtering technique discussed earlier. To combat the source burstiness, the *AR* is implemented at the other end of the ADSL link (typically a *DSLAM* as shown in Figure 4.1). The basic idea is to obtain the information about dropped acknowledgements embedded in the received ACK, and hence regenerate the dropped ACKs. As a result of this the source remains unaware of the ACK dropping and *normal* TCP behavior is observed at the end systems.

Secondly, the duplicate acknowledgements in TCP fast retransmission can be handled in two ways. We can have the *SAD* scheme drop all the ACKs (including duplicate ACKs) of a flow if an ACK belonging to that flow is already present in the queue. In such a case, we maintain a *dup-ACK count* to keep a record of the duplicate ACKs that were dropped. This information can then be used at the *ACK Regenerator* to ensure the regeneration of the duplicate acknowledgements along with other ACKs. The other option is to have *SAD* drop only the non-duplicate ACKs of a flow, if an ACK belonging to that flow is already present in the queue. The choice between these two options is implementation specific. In the former case, we ensure that at most a single ACK from each flow is present in the queue at any given time. Thus the maximum buffer size (for unidirectional data transfer) is always equal to the number of ongoing TCP connections. This also results in the most optimal case as far as the load over the upstream is concerned. The cost paid is in terms of greater processing for the *SAD-AR* schemes, unlike the latter case, wherein we do not process duplicate acknowledgements at all. However, in such a case, the maximum buffer size tends to be variable and we do not ensure optimal upstream load, especially if a lot of duplicate acknowledgements are generated.

Finally it must be noted that *SAD* implements O(1) algorithm to process the outgoing ACKs at the *CPE*. This is due to the fact that it searches the hash table and not the FIFO queue, unlike the ACK filtering technique, wherein the FIFO queue is linearly scanned for every incoming/outgoing ACK, resulting in O(n) processing. The trade-off is the need to maintain a per-flow state using a hash table.

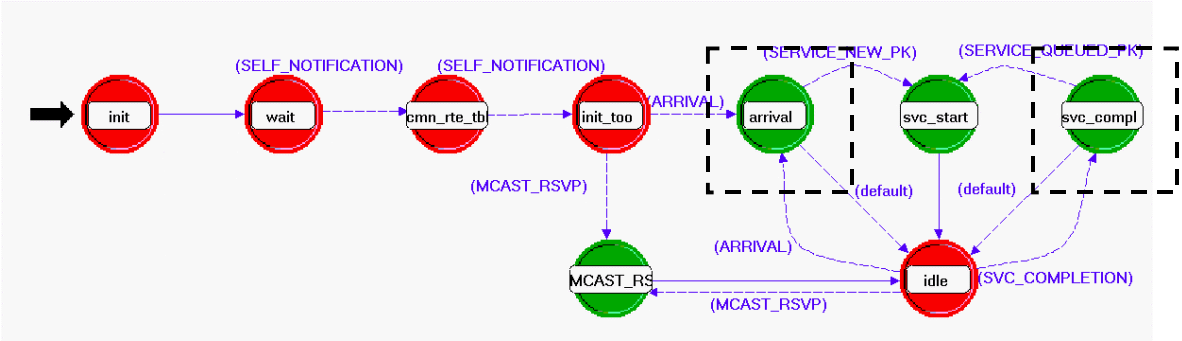

Figure 4.6 Implementation of SAD-AR schemes in the ip_rte_v4 process model

Both the SAD and AR schemes were separately implemented by modifying the *ip_rte_v4* process resulting in *ip_ksp_SAD_rte_v4* and *ip_ksp_AR_rte_v4* process models, respectively. The SAD algorithm described earlier was implemented at the CPE by addition of extensive code in the *arrival* and *svc_compl* states (marked by dotted lines in Figure 4.6) of the original *ip_rte_v4* process model. The arrival state involves maintaining of the SAD connection table and appropriate processing (enqueuing or dropping) of ACKs when they arrive at the IP layer. The *svc_compl* state involves the copying of the latest ACK number from the SAD connection table into the outgoing ACK before it is dequeued. Also, certain elements in the connection table (such as the *FLAG* indicating presence of an ACK in the queue, number of ACKs dropped etc) are initialized in this state. The SAD connection table was defined as a data type *list* (defined in Proto-C, similar to the concept of linked list inC/C++). The pointers to the list (alongwith several other variables were defined in the *state variable (SV*) block while certain other temporary variables were defined in the *temporary variable (TV)* block in the process editor of OPNET Modeler. Each row of the SAD connection table was defined as a data type *structure* (containing variables for the IP source/destination address, TCP source/destination port etc.) in the *header block (HB)* of the process editor. The code in the various blocks of these process model has been provided in Appendix A.

4.6.4 Differentiated Service and Scheduling Mechanisms

In addition to the various schemes mentioned above, we will now briefly discuss certain queuing/scheduling mechanisms that may be used to classify various packets into traffic classes and to provide differentiated servicing of these classes. The purpose of this section is to introduce the various options plausible for differentiated servicing at the IP layer and to provide a launching pad for further research and investigation into quality of service (QoS) for the ADSL network architecture considered. It must be noted that as far as the ADSL access network is concerned, the *asymmetry problem* discussed so far, especially in presence of bi-directional data traffic, which is one of the major challenges for a differentiating service mechanism that may be implemented. If both the data flows (downstream and upstream) are equally critical then it is necessary for such a mechanism to allocate or share resources in order to not starve either of the data flows. The problem is a bit simplified if improvement in one of the flows (e.g. downstream) is desirable at the cost of the other flow (e.g. upstream). In addition, another challenge is to be able to aptly share resources among several flows over the same link (downstream or upstream).

An overview of the various scheduling techniques considered is as follows. We study some of the IP QoS models available in OPNET such as custom queuing (CQ), priority queuing (PQ) and committed access rate (CAR). Both CQ and PQ facilitate traffic classification, prioritization and differential servicing, while CAR is a policing scheme, which allows bandwidth management. CQ can be looked upon as a round-robin weighted fair queuing (RR-WFQ) mechanism. It guarantees a certain minimum bandwidth to the various traffic classes, in proportion to their priorities. Hence, it does not starve the low priority traffic. This is unlike the traffic prioritization in PQ where in the low priority traffic does not get a chance for transmission till the queue for the higher priority is empty.

CAR is a policing scheme, which consists of certain rules based on attributes such as class of service (COS), data rate, normal burst size and excess burst size. These rules define the traffic handling mechanisms when traffic is within the defined policies versus when it violates those policies.

4.7 DSLAM Oversubscription

The focus of the scenarios considered so far has been on the ADSL access network and the different issues related to TCP/IP over PPP over ATM protocol stack when used over asymmetric links. The current scenario can be looked upon as a first step in extending the analysis focus beyond the access network. We now focus on the interface between the access network and the

backbone network, which is typically a *DSLAM*. We employ simulations to characterize the combined effects of asymmetry and of multiple clients competing for bandwidth (available between the *DSLAM* and the *Gateway*) on TCP performance. The scenario of interest is portrayed in Figure 4.7.

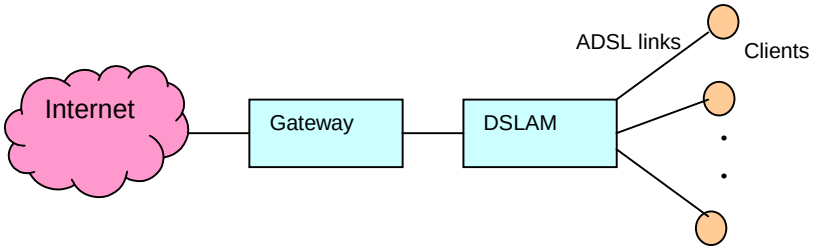

Figure 4.7. Multiple clients competing for bandwidth

In circuit-switched networks, oversubscription is looked upon as major revenue-generating mechanism. When optimally designed, it can prevent waste of bandwidth capacity while allowing maximum number of subscribers on a minimum amount of network infrastructure. The main assumption when implementing oversubscription is that not all users want to transmit at the same time, and hence one can take advantage of statistical multiplexing.

However, in our simulations, we consider a worst-case scenario where in all the clients or subscribers transmit at the same time. This is mainly because, having statistical multiplexing in the simulations does not allow us to gain an insight into TCP performance issues, which is the main focus of this simulation experiment. Further, for practically simulating a realistic oversubscription simulation scenario, we use a scaled-down bandwidth of 1.544 Mb/s for the link between the *DSLAM* and the *Gateway*. Further, the use of downstream speed of 1.544 Mb/s for all the links in the ADSL access network, allows us to directly relate the degree of oversubscription to the number of clients used (e.g. 8 clients correspond to an oversubscription ratio of 8:1). The simulation set-up (referring to Figure 4.1) involves TCP/IP over PPP over ATM implementation up to the *Internet Gateway* and TCP/IP over the *Internet* similar to that used in our earlier simulations.

4.8 System Parameters

4.8.1 Application Traffic Models

The choice of a reasonable traffic model is important for any performance evaluation study. In ADSL systems the downstream provides a greater bandwidth for the large inflow of data traffic. However, flow of bi-directional data traffic is also common (e.g. a user downloading a large file

and at the same time sending e-mails). In our simulations, in addition to unidirectional data transfer we also consider bi-directional data transfer, where in TCP data traffic flows over the downstream and the corresponding TCP acknowledgements share the slower upstream with data traffic in the reverse direction. Using unidirectional data traffic allows us to analyze the *basic asymmetry* problem (effect of normalized asymmetry on TCP performance). Further, bi-directional traffic allows us to study the pronounced effect of asymmetry in such cases. OPNET™ allows modeling of different types of application traffic such as file transfers, e-mail traffic, video streaming, voice etc. File Transfer Protocol (FTP) was used to model long file transfers and generate TCP data traffic along the downstream for both unidirectional and bi-directional traffic scenarios. The main reason for using FTP is that the impact of a slow acknowledgement channel on TCP behavior is best understood by considering such bulk data sources. This also allows us to successfully emulate the real-life traffic scenario for ADSL networks involving large downstream traffic load in applications such as video-on-demand, streaming multimedia etc. For certain baseline scenarios where in we validate our simulation models, typically a single TCP flow is considered.

In case of bi-directional traffic scenario, we consider two classes of traffic to be transmitted over the upstream. These are bursty traffic and non-bursty traffic. Bursty traffic involves data being transmitted in bursts with idle periods of arbitrary durations in between data transmissions. This was modeled using the e-mail traffic in OPNET. Non-bursty traffic involves data being sent at a fixed rate with no bursts. This traffic was modeled using digitized voice. UDP was used as the transport protocol for both e-mail and voice traffic, ensuring that there is no piggybacking of TCP ACKs on data packets. This allows us to clearly observe the effect of *ACK compression* [KVR98], wherein the ACKs of a TCP connection arrive at the source bunched together. Another way to disable TCP piggybacking is by setting the *Maximum TCP ACK delay* attribute in OPNET to zero. This is useful when we deal with TCP data traffic in both directions.

4.8.2 Protocol Attributes

The model of TCP used in the simulations is based on the TCP-Reno version. It supports Jacobson's congestion control mechanism, exponential back-off, enhanced round trip time (RTT) estimation based on both mean and variance of the RTT, Karn's algorithm and the fast retransmit/recovery mechanisms. The default TCP receive buffer size of 64 KB is used. The TCP maximum segment size (MSS) of 1500 bytes (i.e. IP MTU of 1540 bytes) was used for most simulation experiments. However, in certain baseline simulations for both *TCP/IP over ADSL* and

TCP/IP over ATM over ADSL scenarios, different segment sizes were used to emphasize the effect of packet size over asymmetric links.

The ATM modules used are based on OPNET ATM Model Suite (AMS). AAL5 is the ATM adaptation layer used and all the simulation results obtained at the time are for UBR traffic.

4.8.3 Performance Metrics

- **Throughput:** The downstream throughput was one of the important metrics used to characterize the performance of TCP over asymmetric links in all the simulation scenarios. Further, throughput over the T1 link between the DSLAM and the Gateway was also observed specifically for the *DSLAM Oversubscription* scenario.
- **Delay:** Two types of delays were considered: queuing delay at the upstream and end-to-end delay experienced by each TCP ACK in the reverse direction.
- **FTP download response time:** This metric allowed us to characterize the effect of asymmetry on the time taken by a TCP flow to complete a single file transfer.
- **TCP congestion window behavior:** The size of the TCP congestion window was another critical metric in our simulation study, especially while analyzing the performance of the Smart ACK Dropper (SAD) algorithm.
- **TCP segment or *ACK* numbers:** The plot of the TCP segment or *ACK* numbers as a function of time was used to further characterize the system behavior in presence of the Smart ACK Dropper-ACK Regenerator (SAD-AR) scheme.
- **Fairness Index:** Fairness index [Jai91] is defined as follows

$$\text{F. I.} = \frac{\left(\sum_i f_i\right)^2}{n\sum_i f_i^2}$$

where f_i is the throughput for flow *i* and *n* is the total number of competing flows

It was used to study the effect of competing traffic on the fairness of various traffic flows in the *DSLAM Oversubscription* scenario.

- **Client-to-server FTP traffic ratio:** Defined as the ratio of FTP data transfer rate from the transport layer to the application layer at the client to the respective rate from application layer to transport layer at the server, it was used in the *DSLAM Oversubscription* scenario.

4.9 Chapter Summary

This chapter dealt with the simulation methodology adopted as a part of our research exercise. We first took a brief look at the OPNET Modeler™, a network simulation package used for this

simulation study. In the following sections we discussed the end-to-end network simulation set-up and the different simulation scenarios considered. We also discussed the various simulation models developed as a part of these scenarios. In the following chapter, we will now analyze our simulation results obtained using these simulation models.

CHAPTER 5. Simulation Results

This chapter presents the results obtained from the various simulation experiments performed as a part of this research project. The chronological order of sections in this chapter allows us to get a flavor of the various performance issues encountered at every stage of this research exercise, till we finally address the ones related to the entire protocol stack (TCP/IP over PPP over ATM over ADSL). Sections 5.1, 5.2 and 5.3 present the results for the simulation scenarios *TCP/IP over ADSL*, *TCP/IP over ATM over ADSL* and *TCP/IP over PPP over ATM over ADSL* respectively. Sections 5.4 and 5.5 respectively present the simulation results for the *Smart ACK Dropper (SAD)* technique and certain custom queuing/scheduling schemes implemented. In all these sections (5.1 to 5.5), we deal with results for a single-client or single-customer premise type of environment. We extend the simulation environment to a multi-client scenario in Section 5.6, wherein certain preliminary results are presented for multiple clients competing for available bandwidth.

5.1 TCP/IP over ADSL

5.1.1 Effect of Normalized Asymmetry (k): Unidirectional Data Transfer (Single Flow)

As mentioned earlier (Section 3.4), the asymmetry in raw bandwidth (bits per second) is not the asymmetry directly affecting TCP performance; rather, it is the *normalized asymmetry (k)* defined as the ratio of the raw bandwidths to the ratio of the packet sizes in both directions [BPK97, LMS 97]. Assuming that a TCP acknowledgement is generated for each data segment received, the theoretical bound on the downstream throughput is its raw bandwidth divided by *k* [BPK97, KDM98].

The purpose of this section is to validate our baseline simulation models for TCP/IP data transfer over asymmetric links. We set the *maximum TCP ACK delay* attribute to zero in accordance with the above assumption that an ACK is sent for each TCP segment received. We plot the downstream throughput for a single TCP flow (10 MB file transfer) for different values of normalized asymmetry (*k*). This is done by varying the raw bandwidth asymmetry for a certain data packet size (TCP acknowledgements are 40 bytes for all the scenarios) and alternatively by varying the data packet size for a given raw bandwidth asymmetry. The results for both these cases are shown in Figures 5.1 and 5.2 respectively.

- **Variation in raw bandwidth asymmetry**

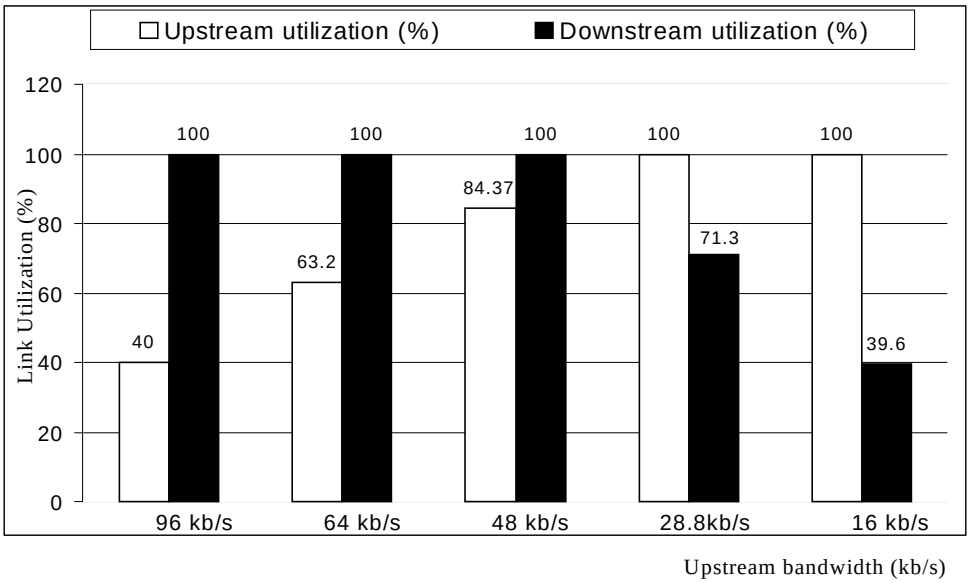

Figure 5.1 Effect of normalized asymmetry (using variation in raw bandwidth asymmetry) on TCP performance (Single TCP flow over 1.544 Mb/s downstream)

We maintain the downstream raw bandwidth at 1.544 Mb/s and consider varied upstream raw bandwidths of 96 kb/s, 64 kb/s, 48 kb/s, 28.8 kb/s and 16 kb/s. For each of these cases the normalized asymmetry factor *k* is calculated with data packet size (IP maximum transfer unit or MTU) of 1540 bytes and TCP ACK size of 40 bytes. For example, with 16 kb/s upstream, $k = \left(1.544\times10^{6}/16\times10^{3}\right)/\left(1540/40\right) = 2.5$, and hence the theoretical upper bound on the downstream utilization is 100/2.5 = 40%. Similarly, for upstream bandwidth of 28.8 kb/s, $k = 1.4$ and the theoretical bound on downstream utilization is 71.4%. For all other cases, k being less than 1, the asymmetry does not affect the downstream link utilization and 100% link utilization is possible. Simulation results for upstream and downstream utilization for the TCP flow have been plotted in Figure 5.1.

Table 5.1 Validation of simulation model: effect of normalized asymmetry by varying the raw bandwidth asymmetry (Downstream: 1.544 Mb/s)

Upstream bandwidth (kb/s)	Normalized asymmetry factor (k)	Downstream utilization (%)		FTP download response time (seconds)
		Simulation	Theoretical	
96	0.4	100	100	63.3
64	0.6	100	100	63.5
48	0.8	100	100	64
28.8	1.4	71.3	71.4	84.8
16	2.5	39.6	40	144.5

We compare our simulation results with the theoretical results and also indicate the effect of asymmetry on the FTP download response time (the time taken by TCP flow to complete the file transfer) in Table 5.1. The simulation results for the downstream link utilization match with the theoretical bounds, thus providing some validation for our simulation model.

- **Variation in packet size**

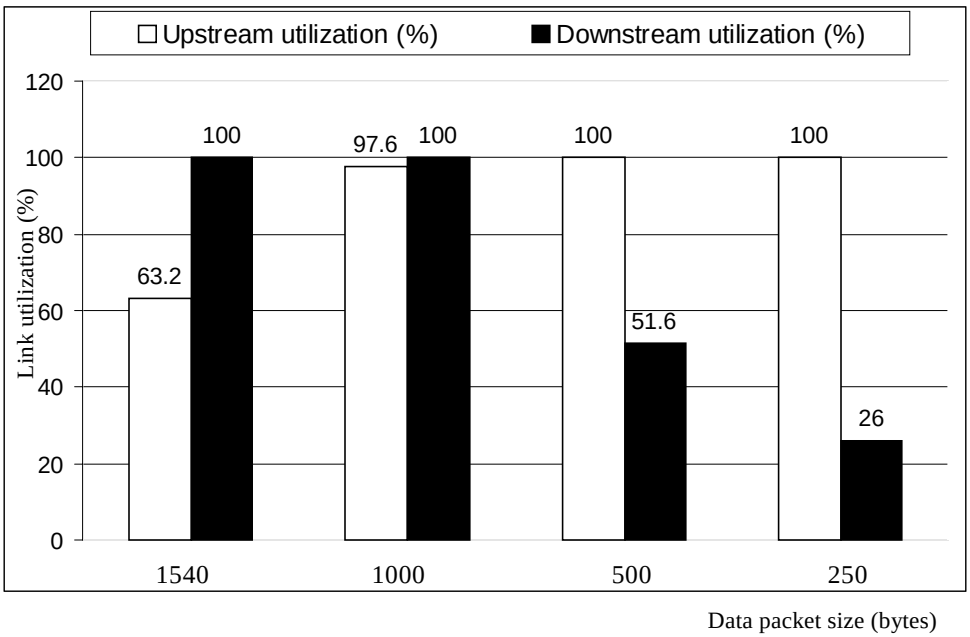

Figure 5.2 Effect of normalized asymmetry (variation in data packet size) on TCP performance (Single TCP flow with 1.544 Mb/s downstream and 64 kb/s upstream)

We now consider a scenario with fixed raw bandwidth asymmetry: 1.544 Mb/s downstream and 64 kb/s upstream and analyze the performance for different data packet sizes (in effect stressing the significance of k). The data packet sizes (IP MTU) used are 1540 bytes, 1000 bytes, 500 bytes and 250 bytes, while the TCP ACK size is 40 bytes for all cases. Calculating k using the definition of normalized asymmetry, it is found that for the four cases mentioned above the value of k is 0.6, 0.965, 1.93 and 3.86 respectively. As shown in Figure 5.2, in spite of having a fixed raw bandwidth asymmetry, we observe downstream throughput degradation for values of $k>1$ and no degradation for $k<1$.

We again compare our simulation results with the theoretical upper bound on the downstream throughput or utilization. Table 5.2 shows a close conformance between the simulation and theoretical results.

Table 5.2 Validation of simulation model: effect of normalized asymmetry by varying the data packet sizes (Downstream: 1.544 Mb/s, Upstream: 64 kb/s)

Data packet size (bytes)	Normalized asymmetry factor (k)	Downstream utilization (%)		FTP download response time (seconds)
		Simulation	Theoretical	
1540	0.6	100	100	63.5
1000	0.965	100	100	64.3
500	1.93	51.6	51.8	118.9
250	3.86	26	25.9	247.8

5.1.2 Unidirectional Data Transfer (Multiple Flows)

We now consider effect of asymmetry in presence of multiple unidirectional TCP flows for two scenarios: 96 kb/s upstream and 16kb/s upstream. To successfully emulate a real-life traffic scenario for ADSL networks involving large downstream traffic load, bulk data transfer with multiple TCP flows has been considered.

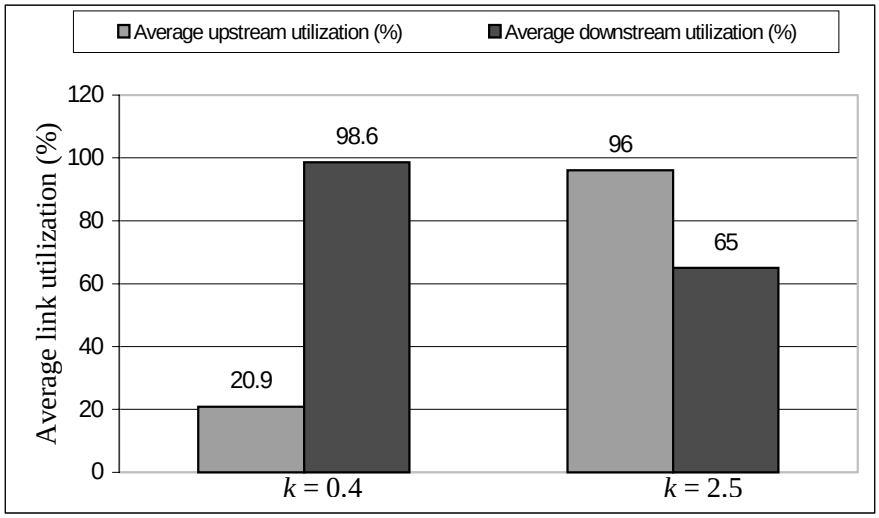

Figure 5.3 Effect of asymmetry on TCP performance in presence of unidirectional data traffic involving multiple TCP flows (downstream: 1.544 Mb/s)

Simulation results are shown in Figure 5.3. For the same amount of traffic generated, we observe downstream throughput degradation in case of 16 kb/s upstream (k = 2.5) and not for 96 kb/s upstream (k = 0.4), as expected. Note that these results have been plotted by using a typical value of 50 ms for the *maximum TCP ACK delay* attribute.

From all the above results, it is seen that in asymmetric networks, low upstream bandwidth can act as a significant bottleneck to the TCP throughput over the downstream channel. The low bandwidth causes congestion over the reverse link, which in turn results in downstream throughput degradation.

We now compare the effectiveness of two proposed solutions [BPK97, ZT99] in presence of high degree of asymmetry (1.544 Mb/s downstream and 16 kb/s upstream).

- **Delayed TCP ACK Technique**

One possible solution to overcome the asymmetry problem is to delay the acknowledgement of received packets, in effect reducing the load over the upstream.

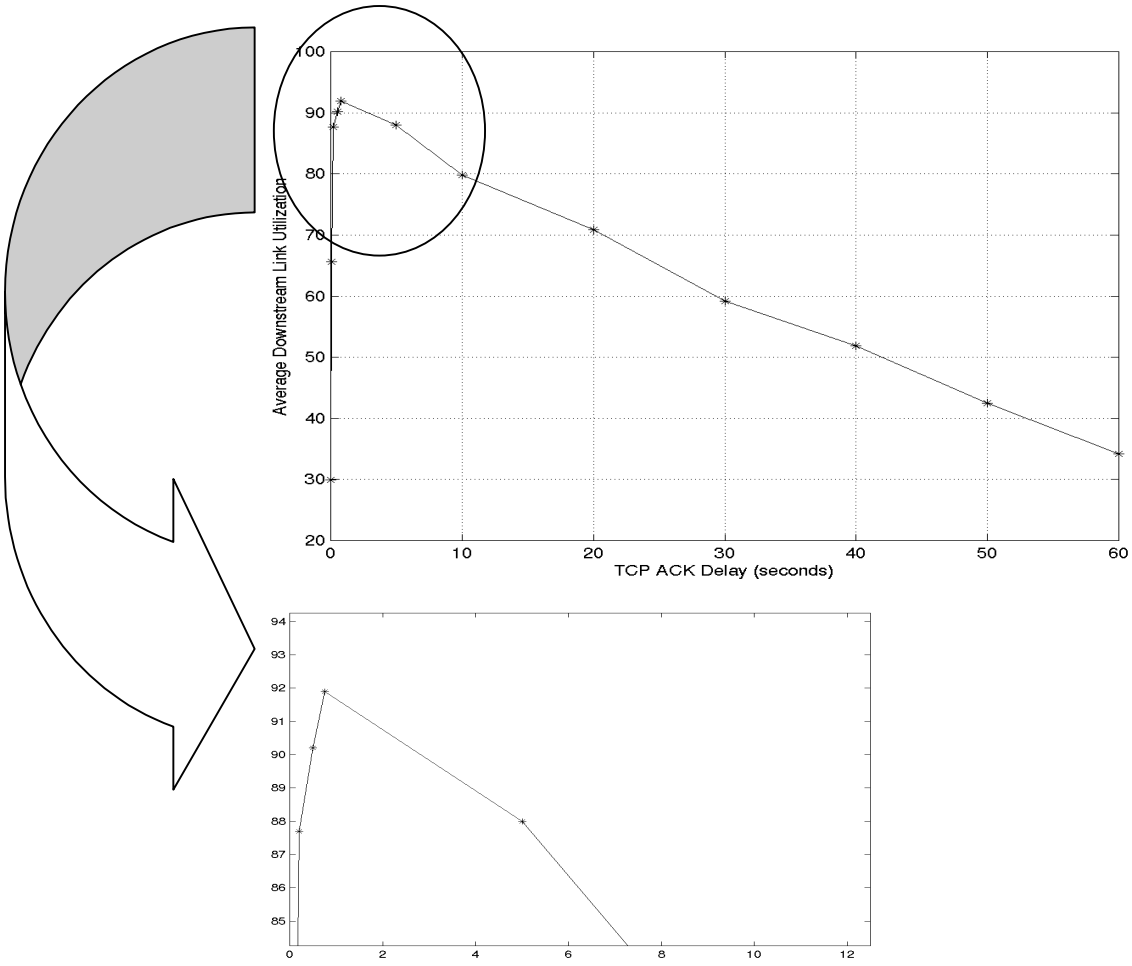

Figure 5.4 Break-even point for the delayed TCP ACK technique: Average Downstream Utilization vs. Maximum TCP ACK delay (Downstream: 1.544 Mb/s, Upstream: 16 kb/s)

To substantiate this claim, simulations were run for zero TCP ACK delay (i.e. every received packet is acknowledged immediately as used in 5.1.1) and for maximum TCP ACK delays of 50 ms and 200 ms (which are among the typical values found in practice). It was found that the scheme with a maximum TCP ACK delay of 200 ms gives better performance than the one with 50 ms followed by the one without any delay. However, it is important to note that the delayed

TCP ACK technique will be limited in improving the performance (downstream throughput) up to a *break-even point*, beyond which further delaying of acknowledgements would actually hamper the downstream throughput, as shown in Figure 5.4.

- **TCP/IP Header Compression**

We now analyze another technique to alleviate the congestion over the slower upstream. Based on the SLIP header compression (CSLIP) technique proposed in [Jac90], we implemented TCP/IP header compression. The basic idea behind this technique is to reduce the size of ACKs, which in turn reduces the normalized asymmetry factor *k*, alleviating the asymmetry problem. The simulation results comparing the delayed TCP ACK and header compression techniques are summarized in Table 5.3. The header compression technique was found to perform even better than the delayed TCP ACK technique with maximum TCP ACK delay of 200 ms.

Table 5.3 Comparison of delayed TCP ACK and TCP/IP header compression techniques (Downstream: 1.544 Mb/s, Upstream: 16 kb/s)

Simulation scenario	Maximum TCP ACK delay	Average downstream utilization
16 Kbps_nodelay	0 ms	30%
16 Kbps	50 ms	65%
16 Kbps_delayed	200 ms	80%
16 Kbps_compress (with header compression)	50ms	98%

5.1.3 Bi-directional Data Transfer (Multiple Flows)

We now consider the case where data transfer occurs along both the downstream (1.544 Mb/s) and upstream (96 kb/s) channels. A typical example is a user sending out data (e-mail) over the upstream and at the same time receiving data traffic (file transfer) along the downstream. In this case, the asymmetry problem is more pronounced, since the TCP ACKs corresponding to the forward data flow now share the constrained upstream with much larger data packets, in effect increasing the degree of asymmetry.

In order to study this phenomenon and to further investigate the effect of the nature of upstream data traffic (bursty versus non-bursty) on the performance of the forward data transfer, we used different traffic mixes in our simulations. Two scenarios were considered: the downstream traffic comprising of FTP in both cases and the upstream data traffic comprising of bursty (e-mail) traffic in one case and non-bursty (digitized voice) in the other. UDP was used as the transport protocol for both e-mail and voice traffic, ensuring that there is no piggybacking of

TCP ACKs on data packets. This allows us to clearly observe the effect of *ACK compression* [KVR98], wherein the ACKs of a TCP connection arrive at the source bunched together.

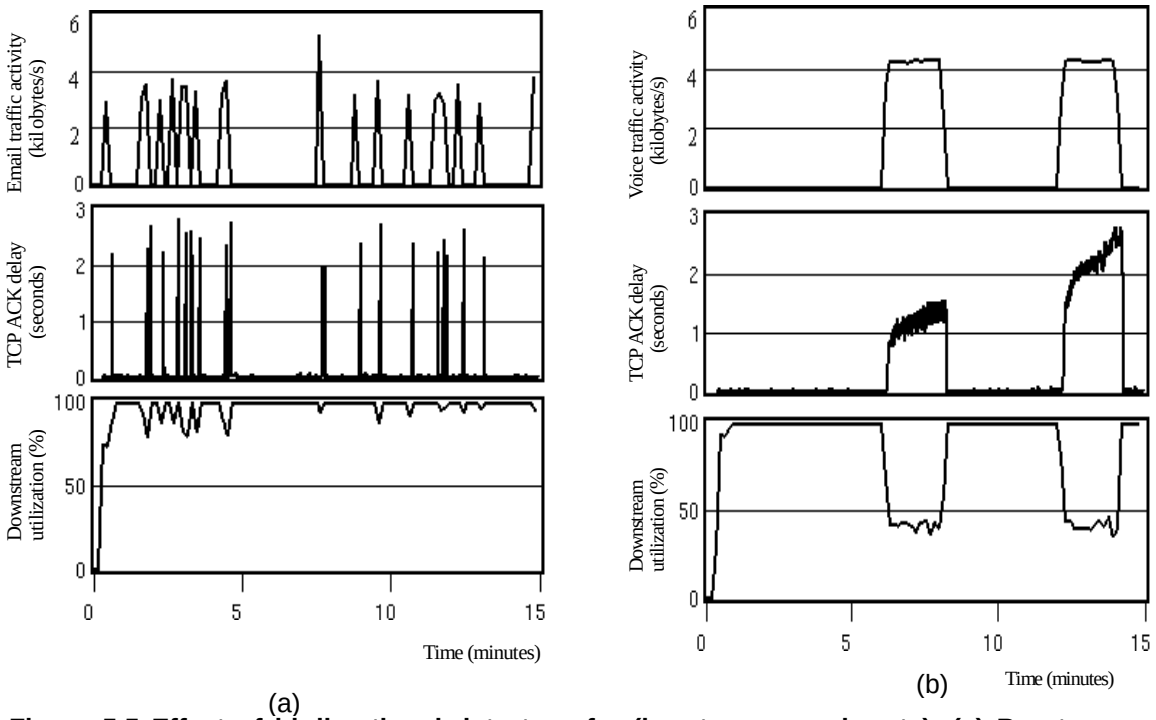

Figure 5.5 Effect of bi-directional data transfer (bursty vs. non-bursty); (a) Bursty case: FTP over downstream (1.544 Mb/s) and e-mail over upstream (96 kb/s), (b) Non-bursty case: FTP over downstream (1.544 Mb/s) and voice over upstream (96kb/s)

We did not observe any downstream throughput degradation in case of unidirectional traffic for the upstream bandwidth of 96 kb/s. However, bi-directional data transfer increases the effective degree of asymmetry causing TCP performance degradation. For roughly the same amount of traffic load generated in both cases, lesser downstream throughput degradation was observed in case of the bursty e-mail traffic as compared to the non-bursty digitized voice, as shown in Figures 5.5 (a) and 5.5(b) respectively.

This is attributed to the fact that in the case of non-bursty voice traffic there is a continuous flow of data packets for the entire duration of an ongoing call, unlike the bursty traffic. Hence, TCP ACKs are consistently caught behind the continuous stream of voice packets and experience large queuing delays for the duration of a call. As a result, the TCP ACKs experience much larger queuing (and hence end-to-end) delays resulting in greater throughput degradation in this case as compared to the case of bursty traffic. This effect is observed in Figures 5.5 (a) and 5.5 (b), where we compare the instantaneous upstream data (e-mail/voice) traffic activity, the corresponding delay encountered by the TCP ACKs and the downstream utilization.

It has been shown that although the TCP/IP header compression technique helps in reducing the transmission time of TCP ACKs, it does not eliminate all the problems in case of bi-directional data traffic [BPK97]. TCP ACKs still get caught behind larger data packets along the reverse link experiencing almost the same queuing delay as before. For similar reasons, the delayed TCP ACK technique does not prove as effective in case of bi-directional data transfer as it does for unidirectional traffic.

5.2 TCP/IP over ATM over ADSL

We kept all the simulation parameters (traffic models, TCP segment size, Reno version of TCP etc.) the same as those used for the *TCP/IP over ADSL* scenario (Section 5.1). The emphasis of these simulations is to study the effect of the ATM overhead on TCP throughput along the downstream in presence of high degree of asymmetry.

5.2.1 Effect of Normalized Asymmetry (k): Unidirectional Data Transfer (Single Flow)

We extend the definition of normalized asymmetry factor *k* to the *TCP/IP over ATM over ADSL* scenario. The segmentation of larger packets (in this case IP datagrams) at the AAL5 and transmission as ATM cells results in overhead commonly known as *cell tax*. This results in a higher degree of asymmetry as compared to the *TCP/IP over ADSL* case for a given set of packet sizes and raw bandwidth asymmetry. For example consider the case with 1.544 Mb/s downstream and 16 kb/s upstream, and 1540 byte data packets and 40 byte ACKs. To transmit a single 1540 byte packet, we will need 33 cells, which will involve a total overhead of 209 bytes (165 bytes for ATM headers, 36 bytes of padding and 8 bytes of AAL5 trailer). Thus, a single packet will now correspond to 1749 bytes instead of 1540 bytes. Similarly, a 40 byte TCP ACK will now be sent as a 53 byte ATM cell (overhead of 5 byte ATM header and 8 bytes of AAL5 trailer). Thus the new value of normalized asymmetry factor *k* would be 2.92 as against 2.5 for the *TCP/IP over ADSL* case, resulting in a further restricted upper bound for the downstream throughput.

Simulations were carried out, this time in presence of ATM, for a single TCP flow to benchmark the effect of ATM overhead or cell tax. Again, the variation in normalized asymmetry is effected using two cases: variation in raw bandwidth asymmetry for a given set of TCP data packet and ACK size, and variation in data packet size for a given set of raw bandwidth asymmetry.

- **Variation in raw bandwidth asymmetry**

Figure 5.6 shows the combined effect of asymmetry and ATM overhead on the link (upstream and downstream) utilization. Comparing these results with the ones in Figure 5.1, it is clearly seen that the ATM overhead or cell tax causes greater throughput degradation as compared to the *TCP/IP over ADSL* case. Comparison in Table 5.4 again shows that our simulation results closely match the theoretical values.

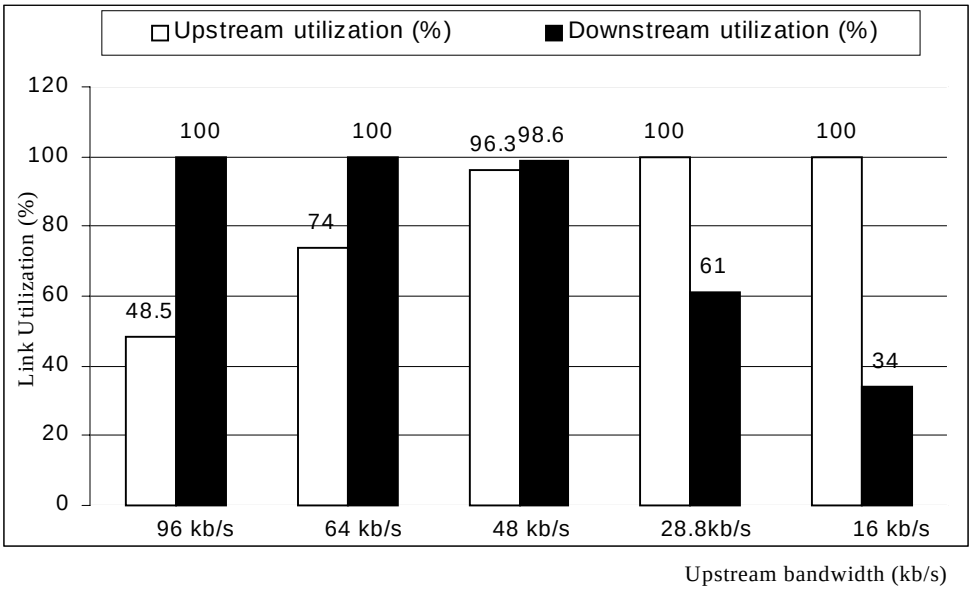

Figure 5.6 Effect of normalized asymmetry (variation in raw bandwidth asymmetry) in presence of ATM on TCP performance (Single TCP flow over 1.544 Mb/s downstream)

Table 5.4 Validation of simulation model in presence of ATM: effect of normalized asymmetry by varying raw bandwidth asymmetry (Downstream: 1.544 Mb/s)

Upstream bandwidth (kb/s)	Normalized asymmetry factor (k)	Downstream utilization (%)		FTP download response time (seconds)
		Simulation	Theoretical	
96	0.49	100	100	70.3
64	0.73	100	100	70.9
48	0.98	98.6	100	71.3
28.8	1.62	61	61.5	109
16	2.92	34	34.2	188

- **Variation in data packet size**

Simulation results in Figure 5.7 again exhibit the importance of normalized asymmetry rather than raw bandwidth asymmetry in analysis of network performance over asymmetric links. Different packet (IP MTU) sizes of 1540, 1000, 500 and 250 bytes are considered in presence of 1.544 Mb/s downstream and 64 kb/s upstream. The results also demonstrate a greater degradation

in downstream throughput (as compared to Figure 5.2) due to faster congestion of the upstream in presence of ATM overhead.

The simulation and theoretical results are listed in Table 5.5.

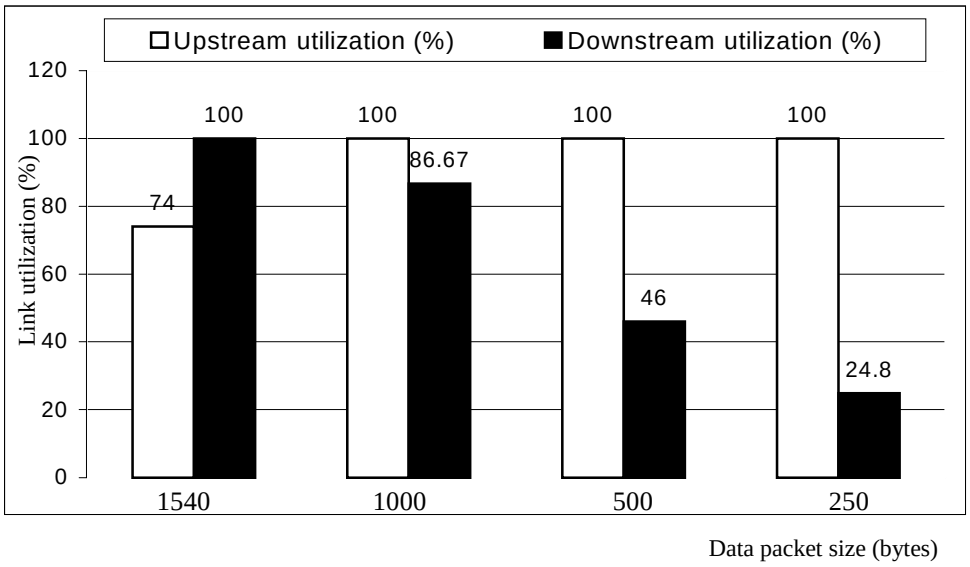

Figure 5.7 Effect of normalized asymmetry (variation in data packet size) on TCP performance in presence of ATM (Single TCP flow with 1.544 Mb/s downstream and 64 kb/s)

Table 5.5 Validation of simulation model in presence of ATM: effect of normalized asymmetry by varying the data packet sizes (Downstream: 1.544 Mb/s, Upstream: 64 kb/s)

Data packet size (bytes)	Normalized asymmetry factor (k)	Downstream utilization (%)		FTP download response time (seconds)
		Simulation	Theoretical	
1540	0.73	100	100	71.0
1000	1.15	86.67	86.95	80.0
500	2.19	46	45.66	154.2
250	4.02	24.8	24.87	325.0

5.2.2 Unidirectional Data Transfer (Multiple Flows)

We again chose upstream bandwidths of 96 kb/s and 16 kb/s for the simulations involving unidirectional data traffic comprising of multiple TCP flows. In addition to accounting for the effect of ATM, we further verify whether the solutions analyzed earlier for *TCP/IP over ADSL*, are now effective in the presence of ATM.

As seen in Figure 5.8, the addition of ATM does not cause any performance degradation in case of 96 kb/s upstream. However, we found that only in case of a very high degree of asymmetry (1.544 Mb/s downstream and 16 kb/s upstream) the overhead introduced by ATM causes further degradation. It must be noted that the main component of this overhead arises from

the fact that each 40-byte TCP ACK is now being sent as 53-byte ATM cell. In addition, the TCP requests for data during initial TCP connection set-up will also be segmented before transmission. This causes congestion over the slower upstream much faster than in the *TCP/IP over ADSL* case. Thus, for 16 kb/s upstream, the average downstream utilization is now limited to 35% (as seen in Figure 5.8) as opposed to 65% in case of *TCP/IP over ADSL* (Figure 5.3).

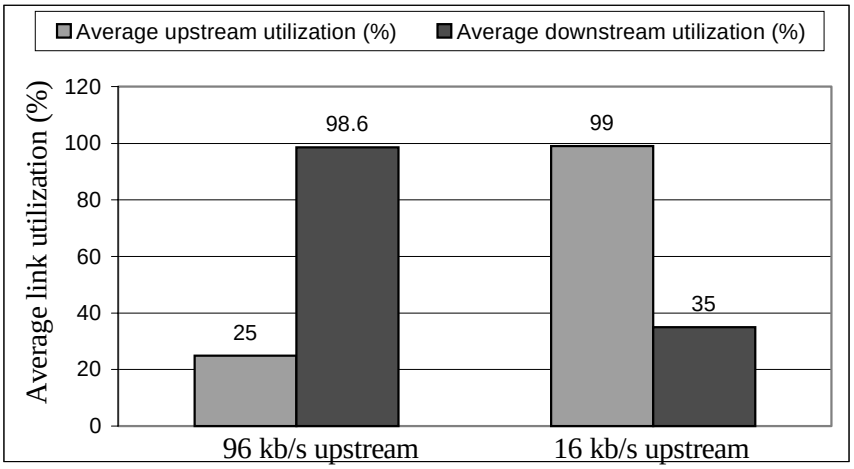

Figure 5.8 Degradation in average downstream utilization in presence of ATM for higher degree of asymmetry i.e. for upstream bandwidth = 16 kb/s (In both cases downstream bandwidth = 1.544 Mb/s)

- **Delayed TCP ACK Technique and TCP/IP Header Compression**

Table 5.6 Comparison of delayed TCP ACK and TCP/IP header compression techniques (Downstream: 1.544 Mb/s, Upstream: 16 kb/s)

Simulation scenario	Maximum TCP ACK delay	Average downstream utilization
16 Kbps_nodelay	0 ms	18%
16 Kbps	50 ms	35%
16 Kbps_delayed	200 ms	47%
16 Kbps_compress (with header compression)	50ms	40%

Unlike the *TCP/IP over ADSL* case, the TCP/IP header compression technique was not found to be as effective in presence of ATM. This is attributed to the fact that, in spite of the header compression, each TCP ACK is still being sent as an ATM cell (this time with additional padding). Thus, the header compression does not help much in alleviating the load over the upstream. Even delaying of the TCP ACKs (delay = 200ms) is not as effective in this case as compared to the *TCP/IP over ADSL* case. However, it results in better performance than the

header compression technique (unlike the *TCP/IP over ADSL* case) since it is able to ease the load over the upstream to a greater extent by reducing the rate at which ACKs are generated. The performance of these techniques is summarized in Table 5.6.

5.2.3 Bi-directional Data Transfer (Multiple Flows)

We used similar bi-directional traffic models as those used in the *TCP/IP over ADSL* case (Section 5.1.3), to study the effect of asymmetry and ATM overhead on the TCP performance. We again observed TCP behavior similar to that shown in Figures 5.5 (a) and 5.5 (b). However, as expected the degradation in downstream throughput was greater in presence of ATM than that in the *TCP/IP over ADSL* scenario. Again the non-bursty (digitized voice) traffic affected the downstream TCP throughput to a greater extent than the bursty (e-mail) traffic. The results have been summarized in Table 5.7 below.

Table 5.7 Effect of ATM in presence of bi-directional traffic (Downstream: 1.544 Mb/s, Upstream: 96 kb/s)

Scenario	***E-mail over upstream***	***Voice over upstream***
	Average downstream utilization	**Average downstream utilization**
TCP over ADSL	95%	81%
TCP over ATM over ADSL	87%	70%

5.3 TCP/IP over PPP over ATM over ADSL

From the viewpoint of quantifying system performance metrics such as throughput and delay, we focused mainly on the effect of the PPP encapsulation of IP datagrams before they get segmented at the ATM layer. The encapsulation of the IP datagrams is done in the LLC encapsulated PPP [GKL+98] frame format. The encapsulation adds 6 bytes of overhead for each IP datagram before transmission. For the purpose of simulation, a PPP maximum receive unit (MRU) of 1540 bytes is assumed, since the IP MTU for the network is maintained at 1540 bytes. This is done in order to avoid any need for padding at the PPP level.

Now with 6 additional bytes of PPP encapsulation (Section 4.5), an encapsulated TCP ACK occupies 46 bytes as opposed to the original 40 bytes. Along with the 8-byte trailer added at the AAL level, the resulting AAL5 data unit (CS-PDU) is 54 bytes long. Thus we now need two

ATM cells instead of one to transmit a TCP ACK. This will result in almost double the load over the upstream and adversely affect the downstream throughput.

We again consider a 1.544 Mb/s downstream and upstream bandwidth of 96 kb/s and 16 kb/s. Figure 5.9 compares the average upstream and downstream utilization with and without PPP encapsulation for the different scenarios in presence of unidirectional (FTP) data traffic.

The simulation results seen in Figure 5.9 validate our claim of having double the load over the upstream as a result of the PPP encapsulation (scenarios *96 kb/s: without PPP* and *96 kb/s: with PPP*). In case of 96 kb/s upstream, the additional load does not affect the downstream throughput. However, for a higher degree of asymmetry (16 kb/s upstream), it causes congestion over the upstream and corresponding downstream throughput degradation (scenario *16 kb/s: with PPP*). The corresponding average delay encountered by each TCP ACK is also shown.

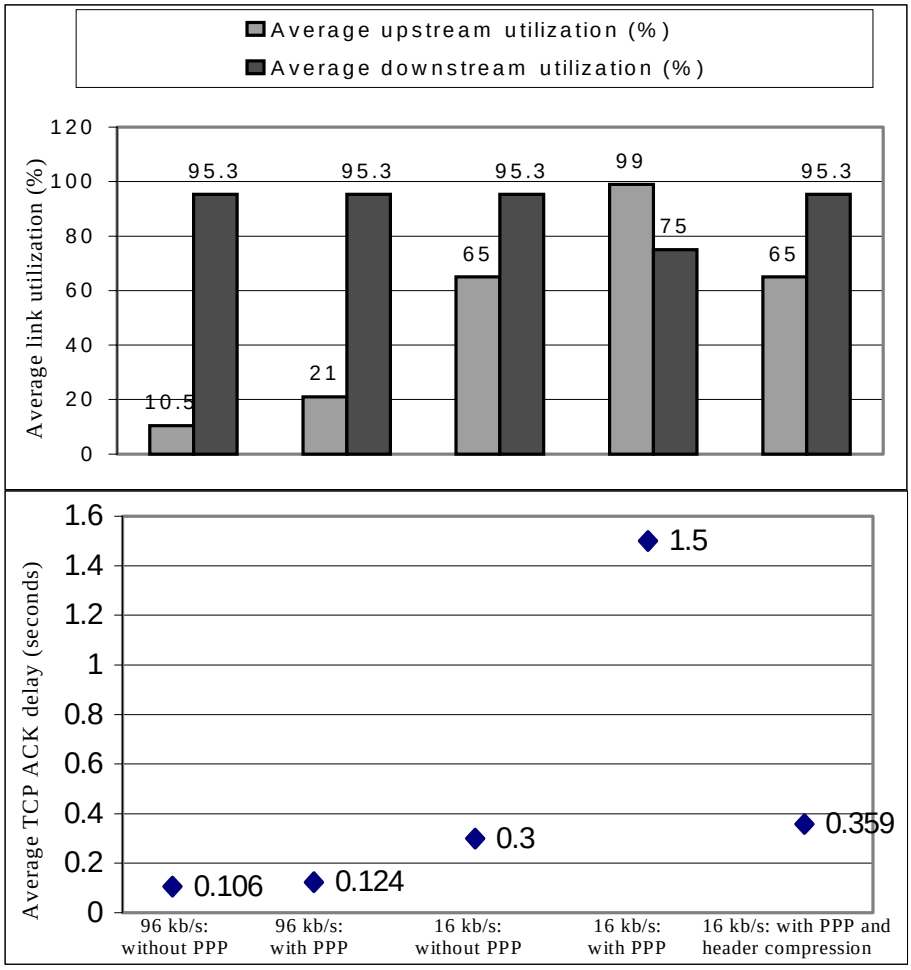

Figure 5.9. Average upstream and downstream utilization, and delay experienced by TCP ACK for 96 kb/s and 16 kb/s upstream speeds, with and without PPP encapsulation

- **TCP/IP Header Compression Technique**

To overcome the overhead problem introduced by PPP, we implemented the TCP/IP header compression technique. Using header compression, we now need only one ATM cell to transmit a TCP ACK instead of two cells, which completely nullifies the overhead due to PPP encapsulation, as seen in Figure 5.9 (scenario 16 kb/s: *with PPP and header compression*). However it must be noted that although TCP/IP header compression helps in alleviating the upstream congestion for unidirectional data transfer, it is not as effective in presence of bi-directional data transfer as already mentioned.

5.4 Smart ACK Dropper (SAD)

The SAD scheme deals with regulation of TCP acknowledgements. In these simulations we are interested in taking a close look at some of the TCP dynamics. Hence, we consider simulation scenarios with just a single TCP connection for the unidirectional case (wherein the data is transferred over the downstream) and one TCP connection in each direction for the bi-directional case. We analyze the SAD scheme with different degrees of asymmetry: T1 downstream -- 96 kb/s upstream and T1 downstream -- 64 kb/s upstream.

We again set the *maximum TCP ACK delay* to zero so that the receiver sends an ACK for each TCP segment received. This also allows us to focus on handling or regulating TCP acknowledgements distinct from the TCP data packets (i.e. no piggybacking) that could be flowing in the same direction in case of bi-directional data transfer.

5.4.1 Unidirectional Data Transfer

- **T1 downstream and 96 kb/s upstream**

We compare the performance of a single TCP connection, in presence and absence of SAD. A single 10 MB file transfer has been considered. As shown in Figure 5.10, the plot of TCP ACK numbers for the ACKs received at the TCP source, in presence of SAD, is found to have a steeper slope than that without SAD. This reflects the fact that, with SAD, ACKs arrive faster at the TCP source (since there are fewer of them to be transmitted) as compared to the scenario without SAD.

The suppressing of TCP ACKs using SAD considerably alleviates the load over the upstream and reduces the queuing delay encountered by each ACK. This in effect results in an increased TCP throughput over the downstream. The results are summarized in Table 5.8 below.

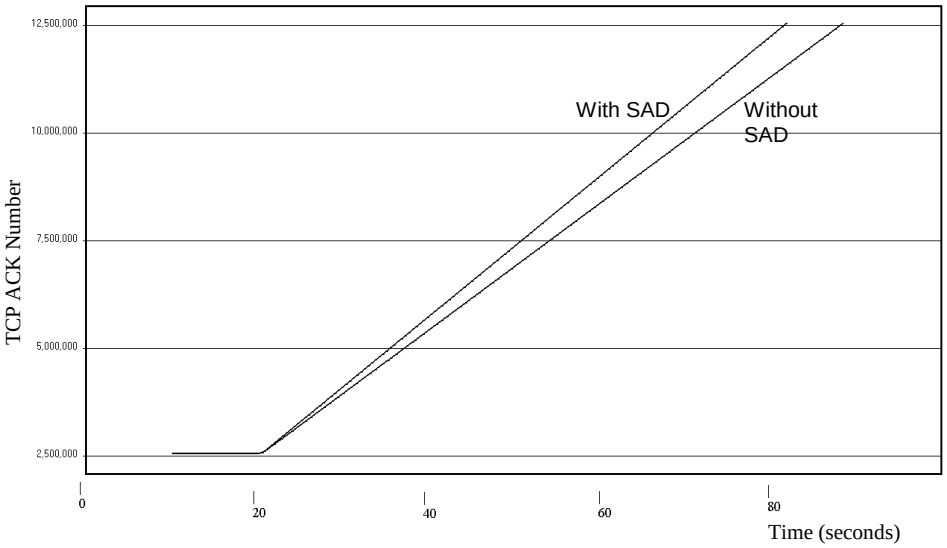

Figure 5.10. Plot of TCP ACK numbers of the ACKs received at the TCP source in presence and absence of SAD (Downstream: 1.544 Mb/s, Upstream: 96 kb/s)

- **T1 downstream and 64 kb/s upstream**

We now assess the performance of SAD for another scenario with the same TCP flow but in presence of a higher degree of asymmetry. As shown in Figure 5.11, a more pronounced effect of SAD is seen in this case as compared to the earlier scenario. The results for both the above scenarios are presented in Table 5.8.

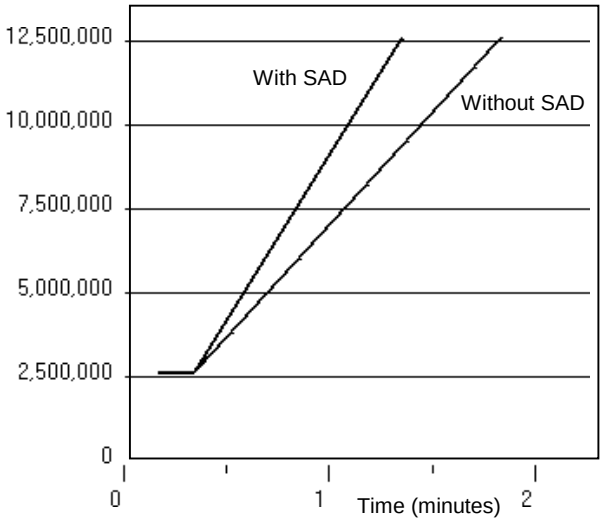

Figure 5.11. Plot of TCP ACK numbers of the ACKs received at the TCP source in presence and absence of SAD (Downstream: 1.544 Mb/s, Upstream: 64 kb/s)

Table 5.8 Performance of SAD (Downstream: 1.544 Mb/s)

Scenario		Upstream Throughput (kb/s)	Downstream Throughput (Mb/s)	Average queuing delay at CPE (seconds)	FTP download response time (seconds)	Total number of ACKs dropped
96 kb/s upstream	Without SAD	85	1.38	0.166	77.8	-
	With SAD	71	1.54	0.008	71	1657
64 kb/s upstream	Without SAD	63.8	1.04	0.233	100.1	-
	With SAD	33	1.54	0.019	71.2	4331

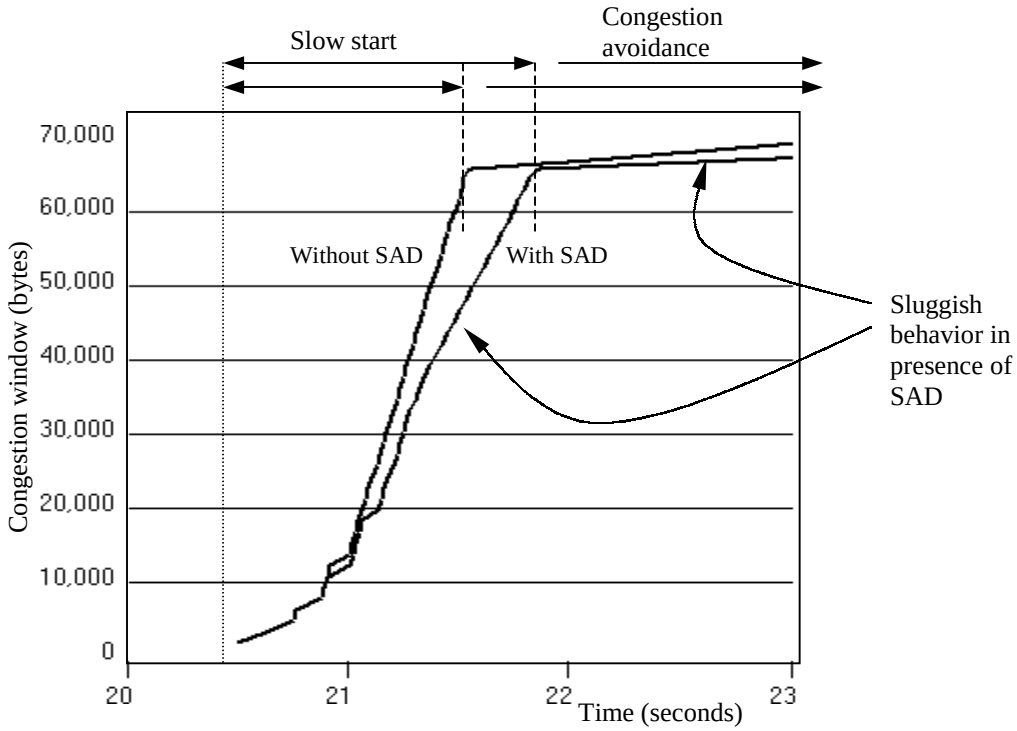

Figure 5.12 TCP congestion behavior in presence and absence of SAD (Downstream: 1.544 Mb/s, Upstream: 64 kb/s)

Another interesting phenomenon about TCP performance observed as a result of this ACK suppression is in the TCP congestion window behavior. The TCP slow start algorithm [Ste94] introduces what is called the congestion window, which allows the sender to perform congestion control as it transmits data. In slow start, the TCP congestion window is typically initialized to one segment (segment size announced by the other end). Every time the sender receives an ACK, the congestion window is increased by one segment. So, initially the sender transmits one segment and waits for the ACK. When it receives the ACK it increases the congestion window size to two segments and can transmit two segments. When ACK for each of these segments is received, the congestion window size is increased to four segments and the sender can now transmit four segments, and so on. This results in an exponential growth in the congestion window during slow start. So, basically during slow start, a TCP source sends one new segment

for each segment acknowledged and increases the congestion window size by one segment for each ACK received.

Now, at a given point during TCP slow start, suppose the TCP source sends W segments where W is the current congestion window size (expressed in number of segments). Let m ($0 < m < W$) be the number of ACKs suppressed by SAD on receiving the W segments, which means that the receiver will transmit (W-m) ACKs instead of W. Thus the new congestion window size will now be (2W-m) instead of 2W. This indicates that the growth of the congestion window size is slower in presence of SAD as compared to that for standard TCP, and this *sluggishness* of the congestion window is directly proportional to m, which will typically be a variable number.

This is found to be true even in case of TCP congestion avoidance, where in the congestion window size (W) is increased by at most one segment when the source receives an ACK for each of the W segments transmitted (i.e. the window size increases by 1/W for each ACK received). However, in presence of SAD (when m ACKs are suppressed for the W segments received), the increment in the congestion window size will now be (1/W) x (W-m) = 1-m/W.

This phenomenon is clearly seen in Figure 5.12, where in the congestion window behavior from our simulation (in presence and absence of SAD) has been depicted. Thus SAD is found to limit the rate at which the TCP congestion window increases during slow start as well as congestion avoidance. Inspite of this fact, SAD allows the TCP source to send data at a faster rate resulting in a much greater downstream throughput as compared to that without SAD. The transmission rate of the sender is determined by what is known as the *effective window* = Min [congestion window, receiver buffer window] – (last byte sent – last byte acked). In our simulations, the congestion window size is smaller than the receiver buffer size during slow start. Although the congestion window grows at a slower rate in presence of SAD, the factor *last byte sent – last byte acked* decreases much faster, since the ACKs reach the sender at a faster rate. This in turn causes the effective window to grow much quickly allowing the sender to send more data.

5.4.2 Bi-directional Data Transfer

- **T1 downstream and 96 kb/s upstream**

In this scenario, we set up bi-directional data transfer using two simultaneous TCP connections or flows, one in each direction: a single 10 MB file transfer along the downstream and a single 1 MB file transfer along the upstream. Both these file transfers are configured to start at the same time. It is found that SAD is not as effective in case of such bi-directional data transfer as compared to the unidirectional case. Although we reduce the amount of TCP ACK traffic using SAD, the ACKs still get trapped behind the larger data packets. Thus, we observe that for the

duration of the TCP data flow over the upstream (approximately for the first two minutes marked by dotted line in Figure 5.13), the plots for the TCP ACK numbers (with and without SAD) traversing the upstream have almost same slope, much smaller than the unidirectional traffic case. However, it must be noted that, as soon as the file transfer along the upstream is over, the network with SAD is able to immediately recover from the impact of bi-directional data flow. The steeper slope of the curve in presence of SAD exhibits this fact in Figure 5.13. This results in the FTP download response time (for the file transfer over the downstream) of 173.2 seconds in presence of SAD as against 299.5 seconds in absence of SAD. The results have been summarized in Table 5.9 below.

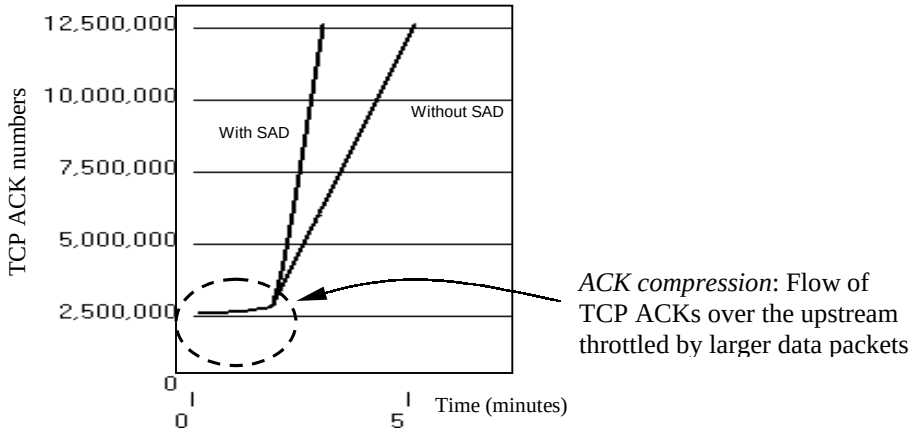

Figure 5.13 Plot of TCP ACK numbers corresponding to the TCP data flow in the forward direction in presence and absence of SAD for bi-directional data traffic (Downstream: 1.544 Mb/s, Upstream: 96 kb/s)

- **T1 downstream and 64 kb/s upstream**

Table 5.9 Performance of SAD in presence of bi-directional data traffic (Downstream: 1.544 Mb/s)

Scenario		Average downstream throughput (kb/s)	Average queuing delay at CPE (seconds)	FTP download response time (seconds)	Total number of ACKs dropped
96 kb/s upstream	Without SAD	330.49	0.578	299.5	-
	With SAD	501.86	0.194	173.2	2058
64 kb/s upstream	Without SAD	236.9	1.08	425	-
	With SAD	404.8	1.05	218.8	5008

A similar behavior for the plot of received TCP ACK numbers corresponding to the downstream TCP data flow was observed even in this case. A much faster recovery from the effect of ACK compression again results in a much better average downstream throughput in presence of SAD as compared to that without SAD. This in effect results in a much smaller download response time (for the downstream TCP flow) as shown in Table 5.9. However, for bi-

directional data transfer, the packets experience a considerably high average queuing delay even in presence of SAD (especially for a higher degree of asymmetry). As mentioned earlier this is mainly attributed to the fact that the ACKs transmitted over the slower upstream still get trapped behind the larger data packets resulting in large queuing delays even in presence of SAD.

5.5 Differentiated Service and Scheduling Mechanisms

In the simulation results presented so far, we found that although several techniques help mitigate the *asymmetry problem*, most of them do not prove to be as effective in presence of bi-directional data transfer. *ACK compression* [KVR98] is the key problem in case of bi-directional traffic, wherein the TCP ACKs (corresponding to the forward flow) get trapped behind larger data packets of the reverse flow over the upstream. Hence there is a need for certain scheduling/queuing mechanisms, which allow classification and differentiated servicing of packets (for example in this case classification between data packets and TCP ACKs sharing the upstream) into different traffic classes.

This section provides certain preliminary results for such differentiating queuing/scheduling techniques. We study and compare two types of queuing mechanisms: priority queuing (PQ) and custom queuing (CQ) for classification of upstream data flow and TCP ACKs. In addition we also study a policing technique (CAR: committed access rate), which allows effective bandwidth management when two or more types of traffic are competing for bandwidth. It is especially useful to control the amount of burstiness of a bursty traffic source allowing fair sharing of bandwidth as will be seen from our simulation results. Using CAR, we can define an average data rate, an acceptable burst size and excess burst size. Further policies can be defined to handle the traffic bursts differently depending on whether they are acceptable (transmit) or are in excess (drop or transmit with lower priority) of the defined burst size.

- **Priority Queuing (PQ)**

The basic idea [BPK97] behind PQ is to prioritize the TCP ACKs over the upstream so that they are no more trapped behind the larger data packets flowing along the upstream. We study PQ in combination with SAD for the bi-directional data traffic scenario with 1.544 Mb/s downstream and 64 kb/s upstream. It is found that prioritizing the TCP ACKs over the upstream data flow considerably improves performance for the file transfer over the downstream. However, this is done at the cost of the upstream TCP data flow, which is starved in this case. The results are presented in Table 5.10.

- **Custom Queuing (CQ)**

We saw that priority queuing or ACK-prioritization improves downstream throughput only at the cost of upstream data flow. On the other hand custom queuing implemented in OPNET can be looked upon as round robin weighted fair queuing, which allows fair sharing of bandwidth. We investigate the performance of custom queuing configured at both edges of the access network. Note that at any given time SAD allows only one ACK per flow to be in the queue. Hence, we configure CQ in a way so that we transmit one TCP ACK (corresponding to the downstream TCP data flow) for every data packet sent over the upstream. From Table 5.10 it is found that, although CQ improves the downstream throughput and FTP download response time (albeit to a lesser extent than PQ), it does not completely starve the upstream data flow resulting in a much smaller FTP upload response time than PQ.

Table 5.10 Performance of PQ with SAD in presence of bi-directional data traffic (Downstream: 1.544 Mb/s, Upstream: 64 kb/s)

Scenario	Average downstream throughput (kb/s)	FTP download response time -- downstream (seconds)	FTP upload response time – upstream (seconds)
Without SAD or PQ	236.9	425	151.8
With SAD	404.8	218.8	150.4
With SAD and PQ	855.9	98.9	642
With SAD and CQ	777.1	109.5	281.5

- **Committed Access Rate (CAR)**

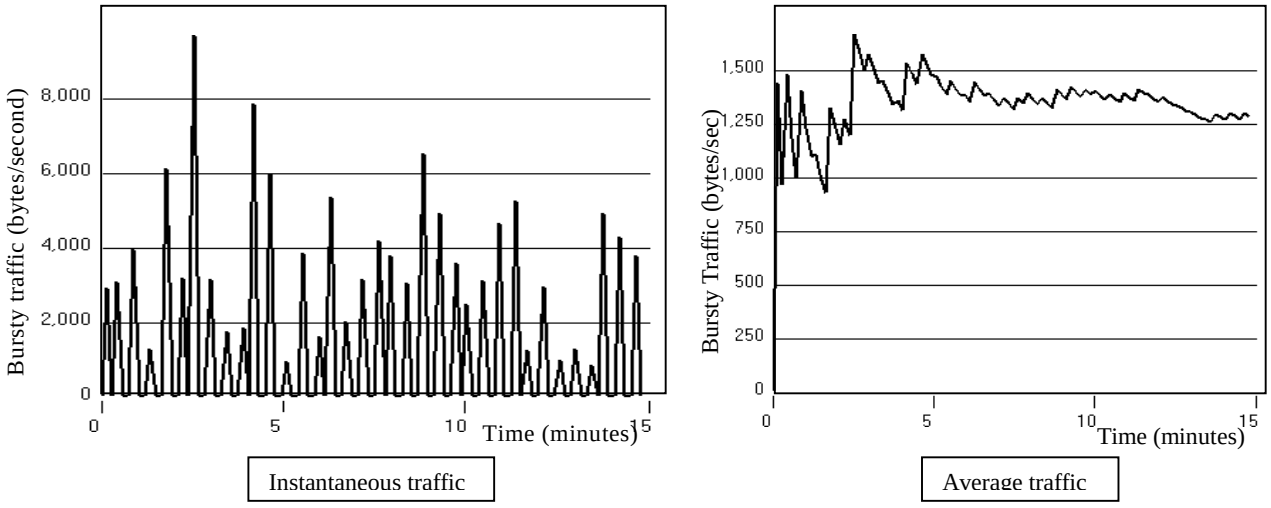

Figure 5.14 Instantaneous and average traffic activity of the bursty source

We consider bi-directional data transfer with a single 10 MB file transfer over the downstream (1.544 Mb/s) and traffic from a bursty source (e-mail) over the upstream (64 kb/s). We compare

the performance of the TCP flow over the downstream with and without CAR and also using both CAR and SAD. The instantaneous and average traffic activity of the bursty source is as shown in Figure 5.14 below.

Table 5.11 Performance of CAR in presence of bi-directional data traffic (Downstream = 1.544 Mb/s, Upstream: 64 kb/s)

Scenario	Average downstream throughput (kb/s)	FTP download response time (seconds)
Without CAR or SAD	232	444
With CAR	322.2	341.72
With SAD	557.2	172.4
With CAR and SAD	705.9	123.7

As seen in Figure 5.14, the average rate is around 1250 bytes/second. We set the average rate limit (in the CAR configuration) to 1250 bytes/second. Further using CAR we set the acceptable burst size to 1250 bytes and excess burst size to 2500 bytes. It uses these burst sizes to calculate the probability with which the incoming traffic exceeds the average rate limit (if at all). We set the CAR policing scheme to drop the traffic (from the bursty source), which exceeds the average rate limit. The simulation results for this experiment are summarized in Table 5.11.

5.6 DSLAM Oversubscription

The simulation results so far characterize the performance of TCP/IP over PPP over ATM in ADSL access network for a single client or small office/home office (SOHO) type of environment. The main focus of the simulations in this section is to characterize the combined effects of asymmetry and of multiple clients competing for available bandwidth on TCP performance. The simulation results that follow present our initial findings in a multiple client scenario and indicate the trend of various simulation metrics in such a scenario.

The simulations in this section are divided into two basic scenarios: *asymmetric* and *symmetric*. The access network links consist of T1 downstream and 96 kb/s upstream for the *asymmetric* scenario, and symmetric T1 links for the *symmetric* scenario. For each scenario, varied number of clients and different types of traffic mixes are used. FTP is used to model TCP traffic along the downstream for both unidirectional and bi-directional data transfer, while a combination of FTP, e-mail and voice traffic is used to model upstream traffic in case of bi-directional data transfer.

Figure 5.15 shows the downstream link utilization and client-to-server FTP traffic ratio as we vary the number of clients competing for the available bandwidth, under the *asymmetric* scenario.

Detailed results are shown in Table 5.12, for both *asymmetric* and *symmetric* scenarios. In both cases, as we increase the number of clients, the utilization of the link between the DSLAM and the gateway remains close to 100%. In case of symmetric links, all clients are found to equitably share the available bandwidth, as evidenced by fairness index almost equal to one.

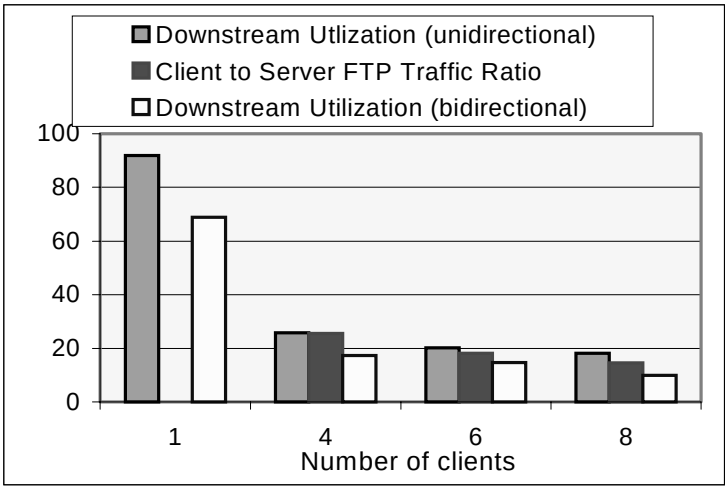

Figure 5.15 TCP performance in presence of multiple clients- asymmetric scenario

However, when asymmetric links are used, downstream throughput degradation for clients with bi-directional traffic is observed. This is mainly attributed to the fact that TCP acknowledgements (ACKs) share the slower upstream with data traffic. TCP ACKs get trapped behind larger data packets, thus encountering greater delay and degrading the downstream TCP throughput. This in turn causes decline in the fairness index as shown in Table 5.12. On the other hand the client-to-server FTP traffic ratio decreases as the number of clients is increased in both *symmetric* as well as *asymmetric* scenarios.

Table 5.12 First four rows of the table refer to *asymmetric* scenario; the last four rows refer to *symmetric* scenario

Number of clients	Downstream link utilization (%) - Unidirectional traffic	Downstream link utilization (%) – Bi-directional traffic	Client-to-server FTP traffic ratio (%)	Link utilization (Gateway → DSLAM)	Fairness Index
1	91.90	68.82	-	-	-
4	25.86	17.36	25.5	98.535	0.966
6	20.16	14.79	18.13	98.535	0.874
8	18.16	10.0	14.56	98.535	0.860
1	91.90	91.90	-	-	-
4	25.86	24.90	26.80	98.535	0.97
6	20.16	15.03	18.47	98.535	0.97
8	12.80	11.37	14.0	98.535	0.96

5.7 Chapter Summary

The simulation results from Section 5.1 through 5.3 characterized the performance of the new protocol stack TCP/IP over PPP over ATM proposed for ADSL networks. Using simulations we studied the effect of asymmetry on the performance of TCP in presence of both unidirectional and bi-directional traffic and portrayed the significance of the normalized asymmetry. We also analyzed the effect of ATM overhead and the effect of PPP encapsulation typically implemented in the ADSL access network. In addition, we compared certain proposed solutions namely the delayed TCP ACK technique [ZT99] and the TCP/IP header compression technique [BPK97] for improving TCP performance over asymmetric links, and tested their effectiveness in presence of ATM and PPP.

In Section 5.4, we characterized and analyzed the simulation results for the Smart ACK Dropper (SAD) technique [KSK99] which we implemented using OPNET Modeler™. To further improve performance especially in presence of bi-directional data transfer over asymmetric links, we study and compare the effectiveness of certain queuing/scheduling and policing mechanisms (Section 5.5), which allow classification of packets into different traffic categories facilitating differential servicing of the packets. Finally, in Section 5.6, we present certain preliminary simulation results studying the effect of asymmetry on TCP performance in presence of multiple clients competing for available bandwidth.

In the following chapter, we will summarize the major contributions of this thesis and present the main conclusions drawn from this research.

CHAPTER 6. Conclusions and Scope for Future Work

This chapter summarizes the primary points of this research and presents the conclusions drawn from the various simulation results obtained as a part of this thesis. It also provides suggestions for future work in this field.

6.1 Final Thoughts

6.1.1 Summary

This research aims at characterizing and analyzing TCP/IP performance in presence of a new protocol stack (TCP/IP over PPP and ATM over ADSL) being promoted for one of the ADSL network architectures.

Today, ADSL is among the top contenders for the broadband access market, the key feature being that it allows delivery of large bandwidth over the same (*omnipresent*) copper infrastructure being currently used for plain old telephone service (POTS). With high-speed Internet access being the primary market driver for ADSL, it is crucial to thoroughly understand the performance issues of TCP/IP used over PPP and ATM over asymmetric links.

Basically, network asymmetry affects the performance of reliable transport protocols like TCP because these protocols rely on feedback in the form of cumulative acknowledgements from the receiver to ensure reliable delivery of packets. Typically, for unidirectional flow of data in an ADSL access network, the TCP acknowledgements (*ACKs*) flow over the slower upstream regulating the flow of data packets or throughput over the downstream. The timely reception of the TCP ACKs when disrupted will cause throughput degradation along the faster downstream. Thus, a low bandwidth acknowledgement path may significantly throttle the downstream throughput, regardless of its high bandwidth. Besides, bi-directional data transfer is found to further exacerbate the *asymmetry problem* due to *ACK compression* [KVR98] (TCP ACKs getting bunched together behind larger data packets over the slower upstream). With this asymmetry problem in mind, we focus our research on the characterization of TCP/IP performance in ADSL access network.

Several researchers [BPK97, KVR98, LMS97] have probed into this asymmetry problem and also suggested ways to alleviate it, albeit only for TCP/IP directly implemented over asymmetric links. We further extend the effect of asymmetry on TCP performance in presence of the point-to-point protocol (PPP) and ATM. This considerably changes the scenario since we now have to also deal with the overhead resulting from the encapsulation of IP datagrams at the PPP layer followed

by the segmentation/overhead (*cell tax*) at the AAL5 and ATM layers. It is found that certain proposed solutions (delayed TCP ACK and TCP/IP header compression) that prove useful in improving TCP performance over asymmetric links, do not prove as effective when PPP and ATM enter the scenario.

Specifically, although TCP/IP header compression is not as useful in mitigating the ATM overhead, it does help nullify the overhead due to PPP encapsulation. An alternate approach is the use of the *Smart ACK Dropper (SAD)* [KSK99], an ACK regulation technique, which is found to considerably improve TCP/IP performance over asymmetric links even in presence of PPP and ATM. The fundamental initiative behind SAD is to take advantage of the cumulative nature of TCP ACKs (dropping the *redundant* ACKs), thus considerably decreasing the load over the constrained upstream. The positive impact of SAD is observed especially for unidirectional data transfer (i.e. TCP data flow over the downstream) over asymmetric links. However, even in presence of bi-directional data transfer, SAD allows a much faster recovery from the effect of ACK compression improving downstream TCP throughput.

The limited success of the various solutions considered in improving downstream TCP throughput in presence of bi-directional data transfer paves way for research into differentiated service schemes. Use of SAD along with certain differentiated (queuing/scheduling and policing) service mechanisms proves to be very effective in presence of bi-directional data transfer, considerably improving downstream throughput. The idea behind these differentiated service mechanisms is to classify TCP ACKs and data packets into different traffic classes, hence queuing and servicing them differently. This in effect helps in mitigating ACK compression. However, simply prioritizing one traffic flow over another may gain performance improvement for the high priority flow, but only at the expense of completely starving the low priority traffic flow. Hence, fair sharing of bandwidth among the upstream and downstream flows is crucial and is one of the main challenges for such schemes. We propose a round-robin weighted fair queuing mechanism, which allows fair bandwidth allocation between the downstream and upstream TCP flows. In addition, we propose a policing scheme to mitigate ACK compression especially in presence of bursty data traffic over the upstream.

6.1.2 Simulation Study

The research done as a part of this thesis was based on a simulation study. Simulations were carried out using OPNET Modeler™. By modifying the already existing protocol stack (TCP/IP over ATM), we implemented the model for PPP encapsulation of IP datagrams over AAL5. This newly developed protocol stack (TCP/IP over PPP and ATM over ADSL) was used to carry out

extensive simulation experiments and gain insight into the various performance issues related to the ADSL network architecture being considered. We implemented and compared the delayed TCP ACK and TCP/IP header compression techniques, both in presence and absence of ATM. Further, we developed a process model implementing the Smart ACK Dropper (SAD) scheme by extensive coding and modifications to the existing IP layer process model. Some preliminary modeling of the ACK regeneration process has also been done (further comments on this process model have been provided in Section 6.3). All the process models developed, involved extensive C/C++ programming and the use of Proto-C (OPNET defined simulation kernels). The relevant code has been provided in Appendix A.

6.2 Conclusions

In asymmetric networks (such as the ADSL access network), the slower upstream channel is the primary bottleneck for the TCP throughput over the faster downstream channel. It is the normalized asymmetry, rather than the raw bandwidth asymmetry that determines the degradation in performance. The amount of throughput degradation increases with the normalized asymmetry factor *k*.

For a given set of raw bandwidth asymmetry and packet sizes (in the forward and reverse directions), the value of *k* is greater in presence of ATM as compared to the *TCP/IP over ADSL* scenario. This is attributed to the overhead at the AAL and the ATM layer. Thus, throughput degradation is found to be greater in case of *TCP/IP over ATM over ADSL* as compared to *TCP/IP over ADS*L.

For unidirectional data transfer, the *delayed TCP ACK* mechanism helps in reducing the congestion over the slower upstream resulting in better TCP throughput over the downstream. This technique was found to be effective even in case of *TCP/IP over ATM over ADS*L, but to a lesser extent as compared to the scenario without ATM. There is a *break- even poin*t beyond which the delaying of the TCP ACKs hampers the throughput, limiting the use of this technique.

The TCP/IP header compression technique considerably improves the performance of *TCP/IP over ADSL*. In presence of ATM, each TCP ACK is sent as a single ATM cell. Hence, compressing of the ACKs will only result in more padding in the ATM cell, limiting the effectiveness of this technique. The delayed TCP ACK technique (200 ms delay) is thus found to be more effective in presence of ATM, while TCP/IP header compression gives better performance in absence of ATM.

Bi-directional or two-way data transfer further exacerbates the asymmetry problem due to ACK compression. The TCP ACKs experience greater queuing (and hence end-to-end) delays since they share the slower upstream with larger data packets. This effect was seen more prominently in case of non-bursty digitized voice traffic as compared to bursty e-mail traffic. Both the delayed TCP ACK and TCP/IP header compression techniques do not prove as effective in case of bi-directional data transfer, since the TCP ACKs still get trapped behind larger data packets along the reverse link experiencing almost the same queuing delay as before. Thus, there is a need to develop solutions to mitigate this problem, especially in presence of ATM.

The PPP encapsulation of IP datagrams over AAL5 almost doubles the upstream traffic for unidirectional data transfer. The main component of the overhead arises from the fact that each TCP ACK now requires two ATM cells for transmission as compared to just one ATM cell in absence of PPP. This may aggravate upstream congestion, degrading the downstream throughput for higher degrees of asymmetry.

TCP/IP header compression helps in alleviating the overhead problem introduced by PPP. Using header compression we now need only one ATM cell to transmit a TCP ACK instead of two cells nullifying the overhead due to PPP encapsulation.

Regulation of the flow of TCP ACKs over the upstream using the Smart ACK Dropper (SAD) technique [KSK99] considerably improves performance especially in presence of unidirectional data transfer. Although the improvement is to a lesser extent in presence of bi-directional data traffic, SAD helps the network in quickly recovering from the impact of bi-directional data traffic unlike the scenario without SAD.

Using certain queuing/scheduling and policing mechanisms in conjunction with SAD can facilitate further improvement in downstream throughput. ACK prioritization using priority queuing improves performance of the TCP flow along the downstream, but only at the cost of starving the upstream data flow. Custom queuing or round robin weighted fair queuing allows fair bandwidth allocation between the downstream and upstream traffic considerably improving downstream throughput without completely starving the upstream data flow.

Committed Access Rate (CAR) is a policing scheme which allows bandwidth management and is found useful especially when TCP ACKs compete with bursty data traffic over the slower upstream. When used in conjunction with SAD, CAR was found to further enhance the downstream TCP throughput.

6.3 Suggestions for Ongoing and Future Work

6.3.1 ACK Regenerator

It was observed that although the Smart ACK Dropper (SAD) technique provides considerable improvements in TCP performance over asymmetric links, it might cause the TCP source to be bursty. This is due to the fact that, in presence of SAD, TCP source receives one ACK for every (n+1) data packets and hence sends n+1 packets for every ACK it receives (n being the number of ACKs suppressed by SAD). This phenomenon will be prominent for large values of n. This may result in problems such as buffer overflows (and possibly congestion) in the forward direction when several such bursty TCP sources send a large burst of packets. As mentioned earlier (Section 4.6.3) the use of the ACK regenerator scheme [KSK99] at the DSL access multiplexer (DSLAM) will avoid such a problem, since the end systems will be kept unaware of the ACK dropping and normal TCP behavior will be maintained. Hence, implementation of AR is crucial and can be the first step for future work.

The basic ACK Regenerator (AR) process has been modeled and can be used as a platform for further enhancements. The AR process model currently does not take into consideration the timing between the regenerated TCP ACKs. Some preliminary testing of this model can be done to gain an insight into the importance of maintaining the spacing between the regenerated ACKs and the way this may be implemented. Further, the trade off (processing versus performance) of regenerating all the ACKs dropped by SAD as against regenerating only a few of the dropped ACKs can be explored. We suggest two possible approaches in modeling ACK Regenerator. One option is to maintain a list of dropped ACKs in the SAD hash table at the customer premises equipment (CPE) and send this list out with the outgoing ACK for that connection. However, in practice this may require some header extension to current IP packet format (if use of certain unused fields is not practical), or alternately the use of some in-band signaling wherein the ACK numbers of the dropped ACKs are sent using a small packet with every outgoing ACK. The other option is to have the DSLAM regenerate and interpolate appropriate ACKs based on the ACK numbers of two consecutive ACKs arriving at the DSLAM.

6.3.2 Quality of service (QoS): differentiated service at IP layer, mapping IP and ATM guarantees and tunneling

The study of certain queuing/scheduling schemes done so far paves way for further investigation into other proposed queuing/differential service schemes (at the IP layer) such as class-based queuing (CBQ) that might be implemented in the ADSL access network. Further,

mapping between such IP QoS schemes with ATM traffic classes is crucial for providing end-to-end service guarantees.

The developed *TCP/IP over PPP over ATM over ADSL* simulation model can be further enhanced to carry various PPP sessions over an arbitrary network and to provide end-to-end connectivity using the Layer 2 Tunneling Protocol (L2TP) [ADS98, Bla99, TVR+99]. This simulation model will allow us to gain further insight into end-to-end quality of service (QoS) issues especially for real-time applications such as voice over IP (VoIP), video streaming, video conferencing etc. run over such a network.

6.4 Thesis summary

We motivated the research problem to be investigated and listed our research goals in Chapter 1. We provided an overview of the digital subscriber line (DSL) family in Chapter 2 and also presented a bird's eye view of a typical ADSL network architecture. In chapter 3, we discussed the various performance issues related to the different network protocols being used over ADSL. In chapter 4, we introduced OPNET Modeler™, a network simulation software that was used for this simulation study. Further, our simulation methodology, the different simulation scenarios considered and the various simulation models developed at every stage of this research project were described in detail. We presented our simulation results and analysis of these results in chapter 5. Finally, in this concluding chapter, we presented our final thoughts about this research, the conclusions drawn from this extensive simulation study and vision for future work.

References

[ADS] ADSL Forum Web Site: http://www.adsl.com.

[ADS98] ADSL Forum, "Core Network Architectures for ADSL Access Systems," white paper 98-017, version 1.01, March 1998.

[ADS99] C. Summers, *ADSL: standards, implementations and architectures*, CRC Press, Boca Raton, FL 1999.

[ALH+99] A. Azcorra *et al.*, "IP/ATM Integrated Services over Broadband Access Copper Technologies," *IEEE Communications Magazine*, vol. 37, no. 5, pp. 90-97, May 1999.

[ATM] ATM Forum Web Site: http://www.atmforum.com.

[ATM99] The ATM Forum Technical Committee, *Traffic Management Specification Version4.1*, March 1999; ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0121.000.pdf.

[Bai97] R. Baines, "Discrete Multitone (DMT) vs. Carrierless Amplitude/Phase (CAP) Line Codes," Analog Devices Inc. whitepaper, May 1997; http://www.analog.com/publications/whitepapers/whitepapers.html.

[Ber00] Y. Bernet, "The Complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network," *IEEE Communications Magazine*, vol. 38, no. 2, pp.154-162, February 2000.

[BPK97] H. Balakrishnan, V.N. Padmanabhan and R.H. Katz, "The Effects of Asymmetry on TCP Performance," in *Proc. of ACM Mobicom'97*, September 1997.

[Bha99] V. Bhagavath, "Emerging High-Speed xDSL Access Services: Architectures, Issues, Insights, and Implications," *IEEE Communications Magazine*, vol. 37, no. 11, pp.106-114, November 1999.

[Bla95] U. Black, *ATM: Foundation For Broadband Networks*, Prentice Hall PTR, Englewood Cliffs, NJ, 1995.

[Bla99] U. Black, *PPP and L2TP: Remote Access Communications*, Prentice Hall PTR, Upper Saddle River, NJ, 1999.

[BP97] J. Bingham and F. van der Putten, Eds., "Interface Between Network and Customer Installation, Asymmetric Digital Subscriber Line (ADSL) Metallic Interface," ANSI T1 draft, September 1997.

[Car98] J. Carlson, *PPP Design and Debugging*, Addison Wesley, Reading, MA, 1998.

[Coh99] R. Cohen, "Service Provisioning in an ATM-over-ADSL Access Network," *IEEE Communications Magazine*, vol. 37, no. 10, pp.82-87, October 1999.

[CKB+99] J. Cook *et al.*, "The Noise and Crosstalk Environment for ADSL and VDSL Systems," *IEEE Communications Magazine*, vol. 37, no. 5, pp.73-78, May 1999.

[Dif] DiffServ Working Group Web Site: http://www.ietf.org/html.charters/diffserv-charter.html

[Gor99] W. Goralski, "xDSL Loop Qualification and Testing," *IEEE Communications Magazine*, vol. 37, no. 5, pp.79-83, May 1999.

[GKL+98] G. Gross *et al.*, "PPP over AAL5," IETF RFC 2364, July 1998.

[HHS98] R. Handel, M. Huber and S. Schroder, *ATM Networks: Concepts, Protocols, Applications*, Addison-Wesley, 1998.

[IET] Internet Engineering Task Force (IETF) Web Site: http://www.ietf.org.

[Int] IntServ Working Group Web Site: http://www.ietf.org/html.charters/intserv-charter.html.

[ITU99] ITU-T Rec. G.992.2, 1999.

[Jac90] V. Jacobson, "Compressing TCP/IP headers for low speed serial links," IETF RFC 1144, February 1990.

[KDM98] K. Kidambi, L. DaSilva and S. Midkiff, "QoS and ADSL," White paper, unpublished, 1998.

[KVR98] L. Kalampoukas, A. Varma and K.K. Ramakrishnan, "Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions," in *Proc. of Sigmetrics'98*, June 1998.

[KSK99] S. Kalyanaraman, D. Shekhar, K. Kidambi, "TCP/IP Performance Optimization over ADSL", ECI Telecom Inc. and Rensselaer Polytechnic Institute Technical Report 1999, unpublished.

[Kwo99] T.C. Kwok, "Residential Broadband Architecture over ADSL and G.Lite (G.992.2): PPP over ATM," *IEEE Communications Magazine*, vol. 37, no. 5, pp.84-89, May 1999.

[LMS97] T. V. Lakshman, U. Madhow and B. Suter, "Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance," *in Proc. Of Infocom'97*, April 1997.

[Lan98] J. Lane, *Personal Broadband Services: DSL and ATM*, Virata's Primer, USA, 1998.

[Max96] K. Maxwell, "Asymmetric Digital Subscriber Line: Interim Technology for the Next Forty Years," *IEEE Communications Magazine*, vol. 34, no. 10, pp. 100-106, October 1996.

[Nor] Nortel Networks Inc., *Asymmetric Transfer Mode (ATM) Fundamentals Tutorial*, Web ProForum Tutorials (http://www.webproforum.com/atm_fund/index.html) for The International Engineering Consortium http://www.iec.org.

[OPN] OPNET Technologies Inc. web site: http://www.opnet.com.

[PD96] L. L. Peterson and B. S. Davie, *Computer Networks: A System Approach*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[PDK00] K. Phanse, L. DaSilva and K. Kidambi, "Characterization of Performance of TCP/IP over PPP over ATM over Asymmetric Links," to appear at the IEEE International Conference on Computer Communication and Network (ICCCN 2000).

[PDK(1)00] K. Phanse, L. DaSilva and K. Kidambi, " Effects of Competing Traffic on the Performance of TCP/IP over Asymmetric Links," to appear at the 25th Annual IEEE Conference on Local Computer Networks (LCN 2000).

[PS99] Z. Papir and A. Simmonds, "Competing for Throughput in the Local Loop," *IEEE Communications Magazine*, vol. 37, no. 5, pp.61-66, May 1999.

[QOS] QoS Forum Web Site: http://www.qosforum.com.

[RF95] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks," *IEEE JSAC*, vol. 13, no. 4, pp. 633-641, May 1995.

[Sim94] W. Simpson, Ed., "The Point-to-Point Protocol (PPP)," IETF RFC 1661, July 1994.

[Sim(1)94] W. Simpson, Ed., "PPP in HDLC-like Framing," IETF RFC 1661, July 1994.

[Ste94] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley Publishing Company, Reading, MA, 1994.

[TK98] M. Tatipamula and B. Khasnabish, *Multimedia Communications Networks: Technologies and Services*, Artech House Inc., Norwood, MA, 1998.

[TVR+99] W. Townsley *et al*., "Layer Two Tunneling Protocol (L2TP)," IETF RFC 2661, August 1999.

[ZT99] L. Zhou and I. M. Tam, "ADSL access network simulation," *Opnetwork'99*, August 1999.

Appendix A. Relevant code for PPP encapsulation and Smart ACK Dropper (SAD)

Code for PPP encapsulation

Note: The code that was added or the portion of the code that was modified has been shown as **bold type**

- Process model: ***ams_aal5_ksp_conn_v3***

---

to_atm state: enter executives

```
/* Data has arrived from the higher layer module.          */
/* The data is processed and sent to the ATM               */
/* as cells.                                               */

/* Determine whether the SDU is from the SAAL or           */
/* from the application layer.                             */

if (aal_argmem_ptr != OPC_NIL)
        {
        /* The SDU is from the SAAL.                       */

                                                *

                                                *

                                                *

else
        {
        /* The SDU is from the APL.                        */

                                                *

                                                *

                                                *

        /* Obtain the packet pointer.                      */
        pkptr = op_pk_get (op_intrpt_strm ());

        if (pkptr == OPC_NIL)
                ams_aal5_ksp_error ("Unable to get packet from input stream.", OPC_NIL,
                OPC_NIL);

        /* Obtain the length of the encapsulated packet (in bytes)*/
        data_len = op_pk_total_size_get (pkptr) / 8;

        /* Create a PPP frame and encapsulate the new arrival      */
        /* within its Information field. The Information field has  */
        /* a modeled size of zero. The bulk size attribute of the PPP*/
        /* frame will instead be used to model the size of the     */
```

```
        /* encapsulated data.                                   */

        /* Encapsulating the network layer packet into PPP frame */
        ppp_pkptr = op_pk_create_fmt ("ppp_pk_fmt");
        op_pk_nfd_set (ppp_pkptr, "Information", pkptr);


        /* Set the bulk size of the PPP packet to model the space */
        /* occupied by the encapsulated data.                   */
        op_pk_bulk_size_set (ppp_pkptr, data_len * 8);

        /* Encapsulate the packet in an ams_aal5_cpcs_pdu.      */
        cpcs_pdu_ptr = ams_aal5_conn_cpcs_pdu_create (ppp_pkptr, user_to_user);
    }

/* Send the CPCS PDU to the ATM layer. */
ams_aal5_conn_cpcs_pdu_send (cpcs_pdu_ptr, payload_type);
```

fr_atm state and fr_atm_rel state: enter executives

```
/* Data has arrived from the ATM layer and may be ready to    */
/* be forwarded to the higher layer process.                  */

if (LTRACE_ACTIVE)
        {
        op_prg_odb_print_major (pid_string, "Received DATA SEGMENT from ATM.", OPC_NIL);
        }

/* Obtain the data segment.                                   */

ppp_pkptr = op_pk_get (op_intrpt_strm ());
if (ppp_pkptr == OPC_NIL)
        ams_aal5_ksp_error ("Unable to get packet from input stream.", OPC_NIL, OPC_NIL);

/* Insert the segment into the reassembly buffer.             */

op_sar_rsmbuf_seg_insert (reassembly_buffer, ppp_pkptr);

/* Determine if a higher layer packet or packets were         */
/* completed.                                                 */

while (op_sar_rsmbuf_pk_count (reassembly_buffer) > 0)
        {
        /* A PDU has been completed.                          */
        if (LTRACE_ACTIVE)
                {
                op_prg_odb_print_minor ("AAL5 CPCS PDU Reassembled.", OPC_NIL);
                }

        /* Obtain the completed packet.                       */

        cpcs_pdu_ptr = op_sar_rsmbuf_pk_remove (reassembly_buffer);
```

```
        if (cpcs_pdu_ptr == OPC_NIL)
                ams_aal5_ksp_error ("Unable to get PDU packet from nonempty reassembly
                buffer.", OPC_NIL, OPC_NIL);

        /* De-encapsulate SDU.                          */

        if (op_pk_nfd_get (cpcs_pdu_ptr, "payload", &ppp_pkptr) ==
OPC_COMPCODE_FAILURE)
                ams_aal5_ksp_error ("Unable to get SDU payload from PDU packet.", OPC_NIL,
                OPC_NIL);

        /* Get user_to_user indication.                 */
        /* This indication is to determine whether the SDU          */
        /* is destined for the SAAL or the APL.         */
        if (op_pk_nfd_get (cpcs_pdu_ptr, "UU", &user_to_user) ==
OPC_COMPCODE_FAILURE)
                ams_aal5_ksp_error ("Unable to get user-to-user indication from PDU packet.",
                OPC_NIL, OPC_NIL);

                                                        *

                                                        *

                                                        *

        /* Decapsulate the contained network level data.        */
        if (op_pk_nfd_get (ppp_pkptr, "Information", &pkptr) == OPC_COMPCODE_FAILURE)
                ams_aal5_ksp_error ("Unable to get data from PPP frame!", OPC_NIL,
                OPC_NIL);

                /* Send the completed packet to the higher layer. */

                op_pk_send_delayed (pkptr, to_apl_stream_index, AMSC_AAL_REASS_DELAY);
                        }
                }
```

- Process model: ***ams_aal_ksp_disp_v3***

Function Block (FB)

```
static Prohandle
ams_aal_disp_conn_create (int requested_aal_type, AalT_Ptc* ptc_mem_ptr)
        {
        Prohandle           aal_prohandle;

        /* This procedure creates and invokes the SAAL process that     */
        /* will handle all the signalling events for this AAL VC connection.    */

        FIN (ams_aal_disp_conn_create (requested_aal_type, ptc_mem_ptr));

        switch (requested_aal_type)
                {
```

```
        case AMSC_AAL_1:

                                        *

                                        *

                                        *

        case AMSC_AAL_5:
                /* Pass in the neighbor data information on creation. */

                aal_prohandle = op_pro_create ("ams_aal5_ksp_conn_v3", ptc_mem_ptr);

                if (op_pro_valid (aal_prohandle) == OPC_FALSE)
                        ams_aal_disp_error ("Unable to create AAL connection process.",
                        OPC_NIL, OPC_NIL);

                                        *

                                        *

                                        *
FRET (aal_prohandle);
}
```

Code for Smart ACK Dropper (SAD)

- Process model: ***ip_ksp_SAD_rte_v4***

Header block (HB)

```
/* Connection Table for SAD scheme */
typedef struct
{
int             source_port;
int             destination_port;
IpT_Address     source_address;
IpT_Address     destination_address;
int             ACTIVE_FLAG;
int             ACKs_dropped;
int             ACK_number;
List*           dropped_ACKs_num_ptr;
}
Connection_Table;
```

arrival state: enter executives

```
/* Acquire the arriving packet. The packet could have arrived via a stream      */
/* interrupt, or from one of the child process (ip_basetraf_src or ip_icmp)     */
/* A packet from the ip_basetraf process indicates that the child had a         */
/* self-interrupt and generated a tracer packet to send bandwidth usage         */
/* information across the network. A packet from ip_icmp process indicates      */
/* a IP control message needs to be sent.                                       */

if (pk_from_child_process == OPC_TRUE)
        {
                                                *
                                                *
                                                *
        }
else
        {
        /* Get the packet from the strm.  Multiple arriving streams are supported */
        instrm = op_intrpt_strm ();
        pkptr = op_pk_get (instrm);
        if (pkptr == OPC_NIL)
                ip_rte_error ("Unable to get packet from input stream.");
        }

/*      Obtain the format of the packet to verify that it is an  IP datagram.   */
op_pk_format (pkptr, ip_packet_format);

/* Ignore if this is not a valid IP datagram.    */
if (strcmp (ip_packet_format, "ip_dgram_v4") != 0)
```

```
                                                    *
                                                    *
                                                    *
else
{
/* Check if the packet is from lower layer */
if (!((instrm == instrm_from_ip_encap) || (instrm == IpC_Pk_Instrm_Child)))
        {
        op_prg_odb_bkpt ("lower_layer");
        ip_client_address = ip_address_create ("1.1.1.1");

        /* Getting source and destination address from IP header*/

        op_pk_nfd_access(pkptr, "fields", &ip_fields_ptr);
        src_addr_temp = ip_fields_ptr→src_addr;
        dest_addr_temp = ip_fields_ptr→dest_addr;

        /* Check if the packet is from client */
        if (src_addr_temp == ip_client_address)
        {
        op_prg_odb_bkpt ("Client");

        /* Smart ACK Dropper Scheme */

        /* Check if the IP datagram carries a TCP segment */

        if (ip_fields_ptr →protocol == IpC_Protocol_Tcp)
                {
                op_prg_odb_bkpt ("TCP");

                /* Create dupilcate copy of the packet */

                pkptr_1 = op_pk_copy (pkptr);

                if (pkptr_1 == OPC_NIL)
                        ip_rte_error ("Unable to create copy of the packet.");

                /* Getting connection and ACK information from TCP segment header */

                if (op_pk_nfd_get (pkptr_1, "data", &tcp_ptr) == OPC_COMPCODE_FAILURE)
                                ip_rte_error ("Unable to get TCP information from IP datagram
                                for SAD!!!");

                /* Destroy the copy */
                op_pk_destroy (pkptr_1);

                /* Get TCP information */

                op_pk_nfd_access(tcp_ptr, "fields", &tcp_fields_ptr);
                src_port_temp = tcp_fields_ptr→src_port;
                dest_port_temp = tcp_fields_ptr→dest_port;
                IF_ACK = tcp_fields_ptr→ack;
                ACK_number = tcp_fields_ptr→ack_num;
                op_pk_destroy (tcp_ptr);
```

```
/* Check if the packet is an ACK */
if (IF_ACK == 1)
        {
        op_prg_odb_bkpt ("ACK_1");

        connection_table_size = op_prg_list_size (connection_table_list_ptr);

        /* Scan the connection table */

        for (row_i=0; row_i < connection_table_size; row_i++)
        {
        connection_ptr = op_prg_list_access(connection_table_list_ptr, row_i);
        table_scan_ptr = op_prg_list_access (connection_table_list_ptr, row_i);


        if ((table_scan_ptr → source_address == src_addr_temp) &&
        (table_scan_ptr → destination_address == dest_addr_temp) &&
        (table_scan_ptr → source_port == src_port_temp) &&
        (table_scan_ptr → destination_port == dest_port_temp))
                {
        /* Check if an ACK belonging to this connection is already present
        in the queue */
        if (connection_ptr→ ACTIVE_FLAG == 0)
                {
                /* No ACK belonging to this connection is in the queue */
                /* So set the ACTIVE_FLAG for this connection, enter the
                /* ACK Number in the table */
                /* Enqueue this ACK */

                connection_ptr → ACTIVE_FLAG = 1;
                connection_ptr → ACK_number = ACK_number;
                goto CONTINUE;
                }
        else
                {
                /* ACK belonging to this connection is already in the
                /* queue */

                /* Check the ACK number!!!...apply ACK dropping only if the
                ACK number */
                /* of the new ACK is greater than that of the ACK present
                /* in the queue */
                if (ACK_number ==  connection_ptr→ACK_number)
                {
                op_prg_odb_bkpt ("ACK_drop");

                /* Record the ACK number of the ACK to be dropped */
                ACK_num_ptr = (int*) op_prg_mem_alloc (sizeof (int));
                *ACK_num_ptr = connection_ptr → ACK_number;
                op_prg_list_insert (connection_ptr →dropped_ACKs_num_ptr,
                ACK_num_ptr, OPC_LISTPOS_TAIL);

                /* Enter the latest ACK number for this newly arrived ACK in */
                /* the table for appropriate connection */
                /* Drop this ACK and increment the ACKs_dropped in table */
                connection_ptr → ACK_number = ACK_number;
```

```
                                (connection_ptr → ACKs_dropped)++;

                                op_pk_destroy(pkptr);
                                insert_ok = 0;

                                goto EXIT;
                                }
                                else
                                {
                                goto CONTINUE;
                                }
                        }
                }
        }       /* End of for loop (i.e. Finished scanning the connection table) */

        /* There is no entry in the connection table for this newly arrived ACK */
        /* Create a new entry for this connection */
        /* Re-initializing and allocating memory space to the new connection pointer */
        new_connection_ptr = (Connection_Table *) op_prg_mem_alloc (sizeof(Connection_Table));


        new_connection_ptr → source_address = src_addr_temp;
        new_connection_ptr → destination_address = dest_addr_temp;
        new_connection_ptr → source_port = src_port_temp;
        new_connection_ptr → destination_port = dest_port_temp;
        new_connection_ptr → ACK_number = ACK_number;
        new_connection_ptr → ACTIVE_FLAG = 1;
        new_connection_ptr → dropped_ACKs_num_ptr = (List *) op_prg_mem_alloc (sizeof
        (List));
        op_prg_list_init (new_connection_ptr →dropped_ACKs_num_ptr);

        op_prg_list_insert (connection_table_list_ptr, new_connection_ptr, OPC_LISTPOS_TAIL);

                                }       /* end if: IF_ACK */
                        }       /* end if: IF TCP */
                }       /* end if: IF from client */
        }       /* end if: IF lower layer packet */

CONTINUE:

                                        *

                                        *

                                        *

EXIT:
```

svc_compl state: enter executives

```
/* Extract packet at head of queue - this is the packet just finishing service. */
/* First we need to determine which CPU has completed the service and then */
/* extract the packet from  the queue of that CPU. The code of the self interrupt  */
/* contains the information about which CPU has completed the service. */

cpu_code = op_intrpt_code ();
if ((cpu_code == CENTRAL_CPU) || (cpu_code == SLOT_TO_CENTRAL_FORWARD))
        {
        /* The central processor completed the service. Extract the packet from the */
        /* module's queue which is the central processor's buffer. */
        pkptr = op_subq_pk_remove (0, OPC_QPOS_HEAD);

        if (pkptr == OPC_NIL)
        ip_rte_error ("Unable to get packet from head of queue.");

        if (pkptr == OPC_NIL)
        ip_rte_error ("Unable to get packet from head of queue.");

        ip_client_address = ip_address_create ("1.1.1.1");

        /* Getting source and destination address from IP header*/

        op_pk_nfd_access(pkptr, "fields", &ip_fields_ptr);
        src_addr_temp = ip_fields_ptr→src_addr;
        dest_addr_temp = ip_fields_ptr→dest_addr;

        /* Check if the packet is from client */
        if (src_addr_temp == ip_client_address)
        {
        /* Smart ACK Dropper Scheme */
        /* Getting connection and ACK information from TCP segment header */
        /* Check if the IP datagram carries a TCP segment */

        if (ip_fields_ptr →protocol == IpC_Protocol_Tcp)
        {
        /* Create dupilcate copy of the packet */
        pkptr_1 = op_pk_copy (pkptr);

        if (pkptr_1 == OPC_NIL)
        ip_rte_error ("Unable to create copy of the packet.");

        /* Getting connection and ACK information from TCP segment header */
        if (op_pk_nfd_get (pkptr_1, "data", &tcp_ptr) == OPC_COMPCODE_FAILURE)
        ip_rte_error ("Unable to get TCP information from IP datagram for SAD!!!");


        /* Destroy the copy */
        op_pk_destroy (pkptr_1);

        op_pk_nfd_access(tcp_ptr, "fields", &tcp_fields_ptr);
        src_port_temp = tcp_fields_ptr→src_port;
        dest_port_temp = tcp_fields_ptr→dest_port;
```

```
IF_ACK = tcp_fields_ptr→ack;

/* Check if the packet is an ACK */
if (IF_ACK == 1)
        {
        connection_table_size = op_prg_list_size (connection_table_list_ptr);

/* Scan the connection table */
for (row_i=0; row_i < connection_table_size; row_i++)
        {
        connection_ptr = op_prg_list_access (connection_table_list_ptr, row_i);
        table_scan_ptr = op_prg_list_access (connection_table_list_ptr, row_i);

        if ((table_scan_ptr →source_address==src_addr_temp) &&
        (table_scan_ptr → destination_address==dest_addr_temp) &&
        (table_scan_ptr → source_port==src_port_temp) &&
        (table_scan_ptr → destination_port==dest_port_temp))
                {
                op_prg_odb_bkpt ("ACK_leaving");

                /* Copy latest ACK number into the TCP header of the outgoing ACK */
                tcp_fields_ptr → ack_num = connection_ptr → ACK_number;

                if (op_pk_nfd_set (pkptr, "data", tcp_ptr) ==
                OPC_COMPCODE_FAILURE)
                ip_rte_error ("Unable to set data in IP datagram for AR!.");

                /* Calculating the number of ACKs dropped by SAD */
                /* before this ACK was sent */
                SAD_ACKs_dropped = connection_ptr → ACKs_dropped;
                op_stat_write (ksp_ACKs_dropped, SAD_ACKs_dropped);

                /* Reset the ACTIVE_FLAG and number of ACKs dropped to zero */
                connection_ptr → ACTIVE_FLAG = 0;
                connection_ptr → ACKs_dropped = 0;

                /* Set the dropped ACK information in the IP header to be used for AR */
                ip_fields_ptr → SAD_AR_table_ptr = (List *) op_prg_mem_alloc (sizeof
        (List));
                op_prg_list_init (ip_fields_ptr → SAD_AR_table_ptr);

                dropped_ACKs_list_size = op_prg_list_size (connection_ptr →
                dropped_ACKs_num_ptr);

                ip_fields_ptr→number_of_dropped_ACKs = dropped_ACKs_list_size;

                for (list_index_i=0; list_index_i < dropped_ACKs_list_size; list_index_i++)
                        {
                        temp_ACK_num_ptr = (int*) op_prg_mem_alloc (sizeof (int));
                        temp_ACK_num_ptr = (int*) op_prg_list_access (connection_ptr→
                        dropped_ACKs_num_ptr, list_index_i);
                        op_prg_list_insert (ip_fields_ptr→ SAD_AR_table_ptr,
                        temp_ACK_num_ptr,
                        OPC_LISTPOS_TAIL);
                        } /* End of for loop  (i.e. finished copying list of dropped ACKs*/
```

```
                op_prg_list_elems_copy (connection_ptr →dropped_ACKs_num_ptr,
                        ip_fields_ptr → SAD_AR_table_ptr);

                op_prg_list_free (connection_ptr → dropped_ACKs_num_ptr);
                op_prg_mem_free (connection_ptr → dropped_ACKs_num_ptr);
                connection_ptr → dropped_ACKs_num_ptr = (List *) op_prg_mem_alloc
                (sizeof (List));
                } /* End of for loop (i.e. finished scanning the connection table) */

                /*      Set the fields structure inside the ip datagram.        */
                 op_pk_nfd_set (pkptr, "fields", ip_fields_ptr, ip_dgram_fdstruct_copy,
                ip_dgram_fdstruct_destroy, sizeof (IpT_Dgram_Fields)); */
                }       /* end if ACK */
        }       /* end if TCP */
}       /* end if from client */
```

*

*

*

Glossary of Acronyms

AAL5: ATM Adaptation Layer 5

ABR: Available Bit Rate

ACK: Acknowledgement

ADSL: Asymmetric Digital Subscriber Line

AR: ACK Regenerator

ATM: Asynchronous Transfer Mode

ATU-C: ADSL Termination Unit at Central site

ATU-R: ADSL Termination Unit at Residence

BT: Burst Tolerance

CAR: Committed Access Rate

CAP: Carrierless and Amplitude Phase

CBR: Constant Bit Rate

CDV: Cell Delay Variation

CLR: Cell Loss Ratio

CO: Central Office

CPE: Customer Premises Equiment

CQ: Custom Queuing

CS-PDU: Convergence Sublayer Packet Data Unit

CTD: Cell Transfer Delay

DiffServ: Differentiated Services

DMT: Discrete Multitone

DSL: Digital Subscriber Line

DSLAM: Digital Subscriber Line Access Multiplexer

FEXT: Far End Cross Talk

GFR: Guaranteed Frame Rate

HDSL: High-speed Digital Subscriber Line

IntServ: Integrated Services

IP: Internet Protocol

ISDN: Integrated Services Digital Network

L2FP: Layer 2 Forwarding Protocol

L2TP: Layer 2 Tunneling Protocol

LLC: Logic Link Control

MBS: Maximum Burst Size

MCR: Minimum Cell Rate

MCTD: Mean Cell Transfer Delay

MFS: Maximum Frame Size

MRU: Maximum Receive Unit

MSS: Maximum Segment Size

MTU: Maximum Transfer Unit

NEXT: Near End Cross Talk

NRT-VBR: Non-Real time Variable Bit Rate

PCR: Peak Cell Rate

POTS: Plain Old Telephone Service

PPP: Point-to-Point Protocol

PPTP: Point-to-Point Tunneling Protocol

PQ: Priority Queuing

QoS: Quality of Service

RADSL: Rate Adaptive Digital Subscriber Line

RT-VBR: Real time Variable Bit Rate

SAD: Smart ACK Dropper

SCR: Sustainable Cell Rate

SDSL: Single-line Digital Subscriber Line

SOHO: Small Office Home Office

SONET: Synchronous Optical Network

TCP: Transmission Control Protocol

UBR: Undefined Bit Rate

UDP: User Datagram Protocol

VBR: Variable Bit Rate

VC: Virtual Circuit

VDSL: Very-high-rate Digital Subscriber Line

Vitae

Kaustubh Phanse was born on March 25, 1976 in the city of Mumbai (erstwhile Bombay), India. He earned his Bachelors degree in Electronics and Telecommunications from the Vivekananda Education Society's Institute of Technology, affiliated to the University of Mumbai. He then joined the MSEE program at the Bradley Department of Electrical and Computer Engineering, Virginia Tech in Fall 1998. After spending three semesters in the Blacksburg campus, he moved to Virginia Tech's Alexandria Research Institute (ARI) in Spring 2000, where a part of this thesis work was completed. After completing his Masters degree in Fall 2000, he will continue for his PhD at the Alexandria Research Institute.

Kaustubh is a student member of the IEEE and its technical society for Communications.