The behavior of TCP transmission over an ADSL/ATM network

By

Bandula Vedage

Department of Teleinformatics
Royal Institute of Technology, Stockholm, Sweden

June 1999

Abstract

This paper describes performance measurements taken in the Telia's Test lab relating to the performance of TCP in ADSL/ATM environment. The behavior of TCP with and without cell loss and cell delay is studied. In particular, we focus on results that indicate that the TCP rate control mechanism alone is inadequate for congestion avoidance and control in ADSL environment. It also explains possible reasons for this poor performance and presents ATM flow control mechanisms that can be used with TCP to allow full utilization of bandwidth. Measurement results showing TCP throughput for various uplink and downlink configurations of the ADSL modem with and without cell loss and cell delay are presented.

Acknowledgements

This thesis project has been carried out with the collaboration between Telia ProSoft AB and the Department of Teleinformatics at the Royal Institute of Technology, Stockholm, Sweden.

I would like to thank my supervisors: Tomas Persson at Telia ProSoft AB and Viktoria Elek at the Department of Teleinformatics at KTH for giving me all the help during the thesis work.

I also like to thank Rune Wahlgren at Telia ProSoft AB and my fellow thesis workers at Telia ProSoft AB, Mathias Södergren and Magnus Lövqvist for their support.

I thank Professor Gunnar Karlsson at Department of Teleinformatics at KTH for his support.

Table of contents

Table of figures

1. Introduction

1.1 Background

The current applications such as video on demand, video conferencing, audio and image transmission etc. are demanding high bandwidth. The existing residential Internet accesses with 56.6Kb and 33.6kb are not enough to handle this ever-increasing demand for bandwidth. Is there a way of using the existing infrastructure to provide better performance? Asymmetric Digital Subscriber Line (ADSL) technology seems to provide the answer. With ADSL the existing Copper telephone lines can be used to deliver megabit services without having to invest on entirely new infrastructure. Currently ADSL can deliver upto 8 megabits per second. For households, this would mean an improvement in network access speeds upto about 200 times. Therefore ADSL will remove the last bottleneck for the high-speed access to the Internet.

TCP/IP is the most widely used protocol today for data transmission. Therefore the services Telia plans to provide will be based on TCP/IP. The data generated by these services will be carried over Telias existing ATM backbone network. ADSL technology will be used at the last link between the customer and the subscriber loop (Central Office), thus providing high-speed access to the Internet. This is a good solution from Telia’s point of view as the existing infrastructures can be used and therefore does not require heavy investments in infrastructure. With this background, we want to investigate the performance of TCP over ADSL/ATM. The tests were carried out to investigate the behavior of TCP with and without cell losses and cell delays. Project goals were formed with this background in mind.

1.2 Project goals

This project seeks to investigate the performance of TCP over ADSL and ATM. Therefore the goals of the project were:

- To study the behavior of TCP over ADSL over ATM. To investigate the parameters that affects the performance of TCP.
    - Up and down link speeds (configurations) of ADSL that gives the maximum throughput.
    - Throughput for different up and down link speeds of the ADSL modem.
    - How is throughput affected by TCP and network parameters?
    - Can throughput be improved by using different traffic classes?
    - Can throughput be improved by using traffic shaping?
- To study a performance evaluation tool that can be used to measure performance in future.
    - To select a performance evaluation tool.

- To document installation and how the tool can be used to make performance measurements.

1.3 Outline of the report

Chapters two to five provides the theoretical background needed to understand performance issues relating to TCP/IP and ATM. Chapter six handles tests and results. Chapter seven deals with future work. Chapter eight is about flow control mechanisms that can be used to improve the performance of TCP/IP over ATM and chapter 9 presents the conclusions that can be drawn from the project.

2. TCP/IP Protocol Architecture

The TCP/IP protocol suite, which is usually denoted by TCP/IP, is a result of a protocol research and development carried out by Advanced Research Projects Agency Network (ARPANET) of the U.S. Department of Defense. The main idea behind the research program was to allow cooperating computers to share resources across a network. TCP/IP consist of a large collection of protocols and its communication tasks are organized into five relatively independent layers. Each layer takes responsibility for handling a set of communication tasks. Figure 1 shows the TCP/IP protocol structure.

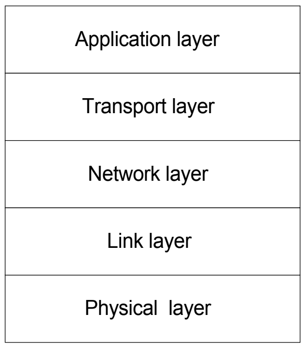

FIGURE 1. TCP/IP PROTOCOL SUITE.

Brief descriptions of the functions of the layers are given below.

Physical layer: This layer covers the characteristics of the underlying transmission medium, signaling and signal encoding schemes to be used.

Link layer: A link layer (network interface) normally consists of a device driver and a corresponding network interface card in the computer. This layer is responsible for accepting IP datagrams and

transmitting them over a network to which it is connected. The type of software used here depends on the type of network protocol to be used.

Network layer (Internet layer): This layer routes packets from source to destination host through intermediate networks connected by routers. It handles requests to send datagrams as well as incoming datagrams. For incoming datagrams, it decides whether to forward the datagram to a router or process it locally.

Transport layer: The transport layer of TCP/IP has two protocols, User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). This layer provides end-to-end data transfer services for the applications.

TCP provides a reliable data transfer service that assures that data arrives without error and in the same order. In order to provide reliability it has functions such as flow control, error control, acknowledgement of received packets etc.
UDP provides a simpler service than TCP. It just sends data from one host to the other without giving any guarantee about the other end receiving the data.

Application layer: This layer handles the logic needed to support a variety of applications. Examples of the most common TCP/IP applications are Telnet, FTP, SMTP and SNMP. An application makes use of TCP or UDP to send or receive data.

2.1 Layering in TCP/IP Internet Environment

The example below shows how TCP/IP operates in an internet environment. This internet consists of two networks: an Ethernet and a token ring connected by a router. Router connects the two networks at the network layer. The application layer in the example handles a user application (FTP) while the lower layers handle communication tasks. It can be seen that application layer and transport layers are using end-to-end protocols while the network layer using hop-by-hop protocol. As shown by Figure 2, message delivery uses two separate network frames: an Ethernet frame and a token ring frame.

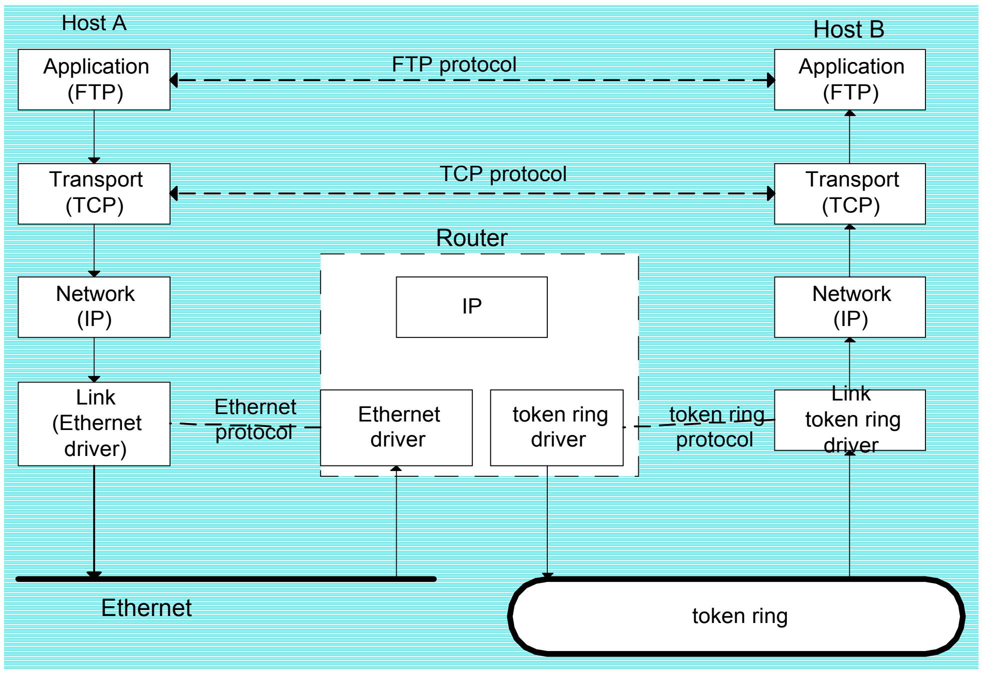

FIGURE 2. TCP OPERATION IN INTERNET.

2.2 Protocol Data Units (PDU) in TCP/IP

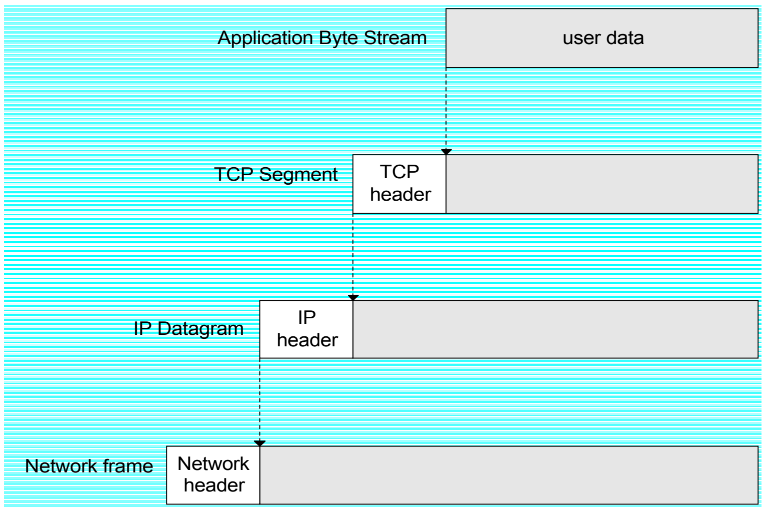

FIGURE 3. TCP OPERATION.

Suppose that a user wants to transmit data to another user using TCP and that data must pass through several subnetworks. The sending process passes users data to TCP. At the TCP layer, user data may be divided into smaller blocks that are manageable. To each of these blocks TCP appends its header, thereby forming a *TCP segment*.
Each of these segments is passed to the IP layer with the information about the receiver. At the IP layer, an IP header is appended to the TCP segment. The resulting piece of data is called an *IP datagram*. Routing decisions are also made at this layer and the IP datagrams are now delivered to the network access layer for transmission across the first subnetwork. The network access layer adds its header to the IP datagram, thereby forming a *frame*. TCP and IP headers are normally 20 bytes long.
Finally, this frame is sent across the subnetwork to the intermediate router. When the packet is received at the router, the frame header is removed and the resulting IP datagram is passed on to the IP layer. The IP layer makes a routing decision according to the destination address and passes the datagram to the appropriate network interface.
This process is repeated until the datagram reaches its destination. At the destination the reverse process occurs. At each layer, the respective header is removed and the remainder is passed on to the layer above. Finally, data is handed over to the receiving application.

2.3 Transmission Control Protocol (TCP)

The Transport layer of the TCP/IP protocol suite consists of Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).
TCP provides a *connection oriented*, *reliable*, *stream delivery* service between two application processes across a network.

Connection oriented means, application processes must make a TCP connection with each other before exchanging data. Although TCP is connection oriented it makes use of connectionless IP packets at the layer below. ***Reliable service*** means the data receiving process gets is without error and loss. ***Stream delivery*** services allow users to exchange streams of data.

TCP makes very few assumptions about the underlying network. This is because of the layering model adopted by TCP/IP. As a result TCP can be used over a complex Internet. The original specification for the TCP is given in RFC 793.

2.4 TCP Data Flow Control

2.4.1 The Sliding Window Concept

A transport protocol provides reliability by acknowledging the data received correctly and retransmitting the data that is lost. This mechanism is generally called *positive acknowledgement with retransmission*. A timer is set as each segment or packet is sent. If an acknowledgement is not received before the timer goes off, the TCP segment is considered lost and is retransmitted. If the transport protocol has to wait for an acknowledgement each time before sending next data, then the link utilization would be very poor. To avoid this, a sliding window protocol is used.
The Sliding window protocol concept explained below uses network capacity efficiently by allowing several packets to be transmitted before acknowledgements are received.
Figure 4 below shows a sliding window at its initial position and Figure 5 shows the position of it after some packets are sent and acknowledged. It should be noted that the window size is kept fixed.

Initial window

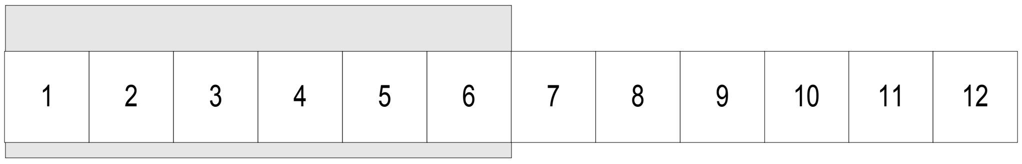

FIGURE 4. A SLIDING WINDOW WITH A SIZE OF SIX PACKETS AT ITS INITIAL POSITION.

A window size of six packets in the above example shows that the sender can transmit six packets before it receives an acknowledgement.

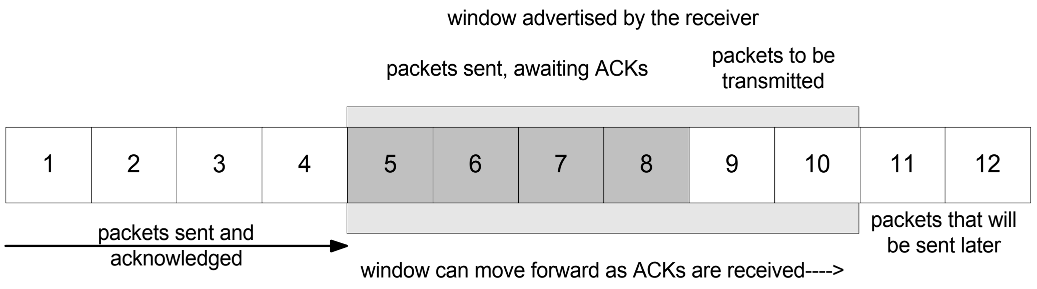
window advertised by the receiver
packets sent, awaiting ACKs
packets to be transmitted
1
2
3
4
5
6
7
8
9
10
11
12
packets sent and acknowledged
window can move forward as ACKs are received---->
packets that will be sent later

FIGURE 5. SLIDING WINDOW MOVED.

2.4.2 Sliding Window Protocol in Action

Figure 6 shows that the sender has sent and received acknowledgements for four packets. As a result, the window is shifted four steps to the right. When a sender receives an acknowledgement for the first packet within the window, the window is shifted to the right by one step. The darkened area within the window indicates that sender has transmitted four packets and the white area inside the window shows that transmission protocol has yet to transmit two packets and it would be sent as soon as possible. Packets to the right of the window would be sent later on as the window slides to the right. It should be noted that the window shown above is the window advertised by the receiver. The performance of sliding window protocol is dependent on window size and network speed. By selecting a suitable window size, network utilization can be increased so that sender can send packets as fast as the network can transfer them. TCP also has a similar sliding window protocol that is explained below.

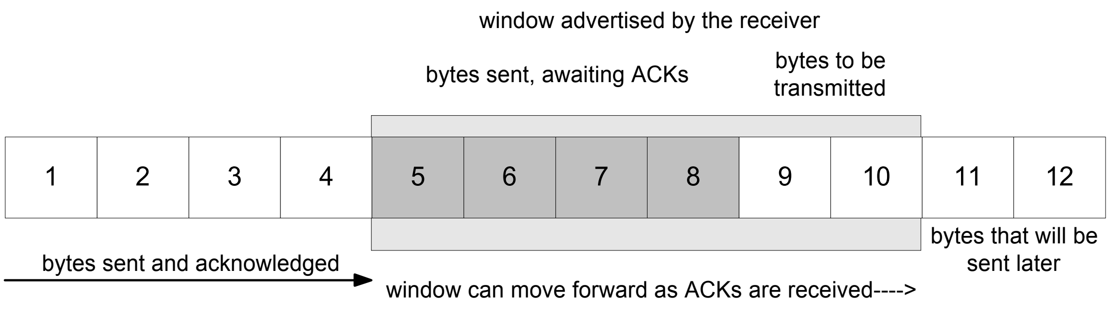
window advertised by the receiver
bytes sent, awaiting ACKs
bytes to be transmitted
1
2
3
4
5
6
7
8
9
10
11
12
bytes sent and acknowledged
window can move forward as ACKs are received---->
bytes that will be sent later

FIGURE 6 TCP SLIDING WINDOW.

2.5 TCP Window Size and Flow Control

The Sliding window protocol described in the previous section was explained in terms of packets and the window size was fixed. But TCPs sliding window protocol instead uses bytes. TCP uses its sliding window protocol to control data flow. The window seen in the above diagram is the window advertised by the receiver. TCP can also send multiple segments to the receiver before receiving acknowledgements for them. It uses window to control flow by allowing the receiver to send the window size in bytes it is willing to accept. If the receiver does not have enough buffer space, it can advertise a small window. If at any moment receiver runs out buffer space, a zero window size would be advertised by the receiver compelling the sender to stop transmission until buffer space is available. Therefore the TCP window is flexible, it can stretch or contract depending on the availability of buffer on the receiver side.
For a normal file transfer, it is the size of the transmission buffer of the sender and the size of the receiving buffer of the receiver that is important. Since TCP connections are full duplex, a TCP connection has four windows to maintain.

2.6 TCP Maximum Segment Size (MSS)

When a connection is established in TCP, each end advertises the maximum size of the segment it can transfer. This can be seen in the hand-shaking phase of connection establishment. If one end does not announce the MSS, *Maximum Segment Size*, the default value of 536 bytes is assumed. A larger MSS would utilize network capacity better than a smaller MSS up to a point where fragmentation occurs. In an Ethernet environment a MSS of 1460 bytes gives the best TCP throughput. This is because Ethernet has Maximum Transfer Unit (MTU) of 1500 bytes. Therefore, segment sizes greater than 1460 bytes in Ethernet environment would lead to fragmentation and hence, a decrease in performance. Optimum segment size for a TCP connection is the maximum size of the segment that can be transmitted between the sender and the receiver without being fragmented. This may be difficult to find in reality because the route between two points may differ from packet to packet.

2.7 Slow Start in TCP

In TCP, data flow is controlled by the receiving TCP's available buffer size. By sending a window advertisement to the sender with the acknowledgement for the segment received, the receiving end controls how many bytes the sender can transmit.
In an Internet environment data may have to flow through intermediate routers. The IP packet that carries TCP data is stored and routing decisions are made at each router. If buffer space were not available in one of these intermediate routers, then the packet that arrives at that time would be lost. This situation arises when the network is experiencing congestion.

TCP uses a mechanism called slow start to avoid congestion. TCP does not send data to match the window advertised by the receiver at the beginning of the transmission. TCP tries to transmit packets to the network at a speed that network can handle. Slow start adds a window to the senders TCP called a congestion window (cwnd). The Congestion window sets the upper bound for the amount of segments a sender can transmit at a given time. For a new connection, cwnd is set to 1 at the start, and each time an ACK is received cwnd increased by one segment.
The size of the window that is allowed to send data at a given time is given in the formula below.

Allowed window = minimum of (advertised window by the receiver, congestion window)

The maximum number of segments a sender can transmit at any time is the receivers advertised window.

2.8 Timeout and Retransmission

TCP provides reliability by requiring the receiving end to send an acknowledgement for received data. When TCP sends a data segment, a timer is set and an acknowledgement is expected within some precalculated time interval. If data is lost on its way to the receiver, an acknowledgement is not received and as a result timer would go off. Once the timer goes off, data is retransmitted. Figure 7 below shows an example of time out and retransmission.

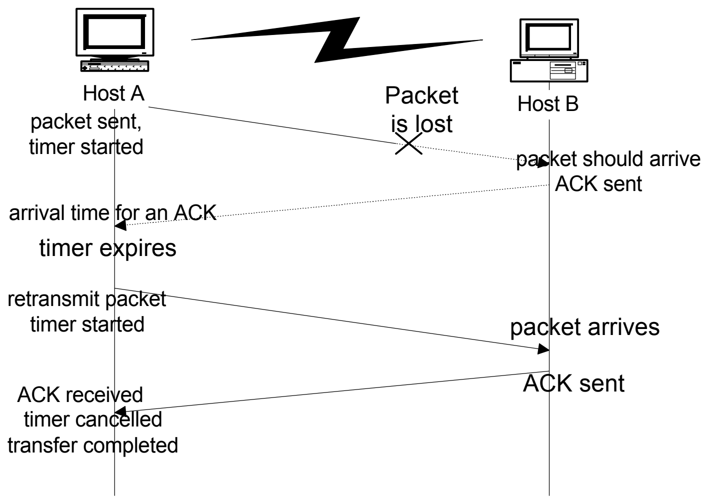

FIGURE 7. TIMEOUT AND RETRANSMISSION.

Calculation of the timeout period is not a simple issue. This is because the expected time of acknowledgement arrival can vary in an Internet environment. In an Internet, a packet would travel across multiple intermediate routers. If intermediate networks were congested, then the packet would experience delays due to queuing in buffers. Therefore it is a difficult task to predict when an acknowledgement will arrive. TCP uses an adaptive retransmission strategy to cope with varying delays in an Internet environment. TCP calculates timeout for a connection continuously and updates it accordingly.

2.9 Round-trip Time (RTT) Measurement

The time it takes from sending a segment and receiving an acknowledgement is called Round-trip time (RTT). Since TCP can acknowledge multiple segments, it is enough that an acknowledgement covers the segment that is used in RTT measurements. TCP measures RTT in a simple way. A timer is set when a segment is transmitted and is cancelled when the acknowledgement is received. The difference in time is the RTT for that instance and is called the sample round trip time. With time, network traffic and routes would change. This would lead to new values of RTT. TCP uses these new RTTs to calculate average round trip time for the connection. A formula for calculating RTT is given below. The resulting RTT is called estimated RTT or smoothed RTT.

$$\text{RTT} = \alpha * \text{Old RTT} + (1-\alpha)*\text{Measured RTT}$$

α is called a *smoothing factor*, where $0 \leq \alpha < 1$. If for example α is taken to be 0.8, then the calculated RTT would consist of 80% of the old RTT and 20% from the new measurements. RTT is updated each time a new measurement is made.
The value of RTT is then used to calculate the retransmission timeout (RTO), when a data segment is transmitted by the TCP. The retransmission timeout is given by the following formula.

$$\text{retransmission timeout} = \beta * \text{current RTT estimate}$$

β is called the *delay variance factor*. Having a value 1 for β would give a timeout close to the estimated RTT. This would let TCP detect a packet loss quickly and retransmit the lost packet. But on the other hand if the acknowledgement is only delayed and not lost, TCP would retransmit a packet that will not be used. The original TCP specification recommends a value of 0.9 for α and a value of 2 for β.

2.10 Accurate RTT Measurements

Although it is easy to measure RTT, ambiguities arise when trying to measure RTT in some situations. Assume that a datagram is sent and the acknowledgement (ACK) is delayed for some reason. This would let the timer go off and TCP would retransmit data. A problem arise, when an ACK is received. The sender cannot tell whether the acknowledgement is for the first transmission or for the retransmission. This implies that sender cannot measure the correct RTT. This is called the *retransmission ambiguity problem*. If the sender assumed that the acknowledgement is for the first transmission when it is really not so, it would make the RTT estimate much longer than it should be. Assuming that the acknowledgement is for the retransmitted segment when it is not so would lead to a much smaller RTT and hence smaller timeout. This would trigger timeouts and unnecessary retransmissions. Therefore we see that we cannot associate the acknowledgement to the first transmission or the most recent retransmission, because both fail to provide correct round trip times.

Is there a way to get around the retransmission ambiguity problem? *Karn's Algorithm* provides a solution for the retransmission ambiguity problem.

2.11 Karn's Algorithm

Karn's Algorithm avoids retransmission ambiguity problem by adjusting RTT only for acknowledgements received for data sent once. This implies that RTT would not updated after receiving an ACK for a retransmitted segment. This can also fail in certain situations. Assume that the network load has gone up. This would inevitably lead to increased delay in the network. The timeout calculated just before the load increase would now be small compared to the delay experience by the network. As acknowledgements would take more time to reach the sender, timeout would occur and force retransmission of data. From now on, RTT values would not be updated by the TCP due to retransmission ambiguity problem. TCP would never update new timeout values, instead TCP would use old estimates, leading to more retransmissions.

Karn's algorithm introduces a timer back-off strategy to handle failures described in the previous paragraph. Until a retransmission occurs, the estimated RTT would be used for timeout calculations. Once a retransmission occurs TCP would increase the timeout. The new timeout is given by the formula below.

$$\text{New timeout} = \gamma * \text{old time out}$$

Multiplicative factor γ is normally set to 2. Although new timeout can increase indefinitely, most implementations set an upper limit for the calculated timeout. Karn's algorithm works well in networks with high packet losses.

2.12 Responding to High Delay Variance

Although Karn's algorithm works well for lower delay variations, it becomes insufficient for networks with wide fluctuations in the round trip time. Retransmission timeout (RTO) calculated in the previous section (timeout = β * current RTT estimate) assigned a value 2 for β. To accommodate wide fluctuations in the RTTs, TCP implementations now estimate variance in RTT in addition to the average round trip time. Instead of using just β in RTO calculations, β is replaced with the estimated variance. Calculating RTO with the help of following equations [6] has resulted in retransmission timeouts that are more sensitive to fluctuations in RTT.

The following equations are applied to each RTT measurement and next RTO is calculated accordingly.

$$\begin{aligned}\text{Difference (Diff)} &= \text{Measured RTT (M)} - \text{Estimated RTT (O)}\\ O &= O + g * \text{Diff}\\ \text{Dev} &= \text{Estimated Mean Deviation (Dev)} + h\,(\,|\text{Diff}| - \text{Estimated Mean Deviation (Dev)})\\ \text{RTO} &= O + 4 * \text{Dev}\end{aligned}$$

New values for Smoothed RTT and Deviation are calculated using their previous values and values obtained by last measurements. The values of g and h are chosen to be an inverse of a power of 2. Default value of g is set to 1/8 and h to 1/4.

2.13 Congestion

If the network load is high, then intermediate routers may not be able to queue all the incoming packets. As a result, congested routers start dropping packets until buffer space becomes available for the incoming packets.
The end points do not have any knowledge about the intermediate routers. Therefore when a packet is lost, the sender would get to know about it only if sender gets duplicate acknowledgements or its timer goes off when an acknowledgement is not received within a RTO. According to [Stephen] packet loss occurs mainly due to congestion in routers and not due to errors in the packet.
As a result of congestion, the packets have to stay in queues until they get the services from the routers and therefore acknowledgements may not reach the sender of data in time. Therefore, a timeout may not be due to a packet loss, but may be that the network was congested and the ACK got delayed. A timeout force TCP to retransmit data, thereby making network even more congested. If this process is not terminated in time, the sender would inject more and more packets into the network resulting the network to collapse. This situation is called *congestion collapse*. To avoid congestion collapse, TCP must slow down its transmission of packets into the network. To slow down its transmission, TCP uses congestion avoidance and slow start algorithms. These two algorithms work together to reduce transmission rate and thereby congestion.

2.14 Congestion Avoidance Algorithm in TCP

For the slow start and the congestion avoidance algorithms to work, TCP must maintain two variables for each connection: TCP must remember the size of the congestion window (cwnd) and slow start threshold size (ssthresh) [6].
The combined algorithm works as follows.

1. Congestion window (cwnd) is initialized to one segment size and start threshold size (ssthresh) is initialized to 65535.

2. Congestion window is the flow control mechanism imposed by the sender. Receiver’s advertised window is the flow control imposed by the receiver. The receiver’s advertised window is the available buffer space at the receiver. TCP transmits packets into the

3. network according to the formula given below. The allowed window gives the number of bytes TCP can transmit at a particular instance.

   Allowed window = minimum of (receivers advertised window, congestion window)

4. When TCP gets an indication about congestion in the form of duplicate ACKs or timeout, the value of ssthresh is set to one-half of the allowed window. This will reduce the network traffic considerably. If loss continues, TCP would continue to reduce traffic into the network exponentially until the allowed window size becomes one packet. If the congestion is due to delay in the network (that is, timeout occurred), then TCP also sets cnwd to one segment. TCP assumes that delay was caused by increased network load and therefore tries to minimize transmission volume.

5. When congestion ends, TCP would start injecting more and more segments into the network. For each acknowledgement received, congestion window would be increased. How it is increased depends on current values of cnwd and ssthresh.

   If cnwd > ssthresh, then TCP is said to be in the congestion avoidance phase. In this phase cwnd is increased by 1/cnwd for each acknowledgement received or at most 1 segment each round trip time without considering the number of acknowledgements received.

   If cnwd ≤ ssthresh, then cnwd is incremented by 1 segment for each acknowledgement received. This means congestion window would increase exponentially until it reaches ssthresh. When it reaches ssthresh, TCP would enter the congestion avoidance phase and act accordingly.

Figure 8 shows visually how slow start and congestion avoidance work. When a connection is established, TCP initialize cwnd to 1 segment, and is increased by one segment for each acknowledgement received. This implies that allowed window size increases exponentially until congestion occurs. At that point TCP sets ssthresh to one-half of the cwnd size existed just before congestion.

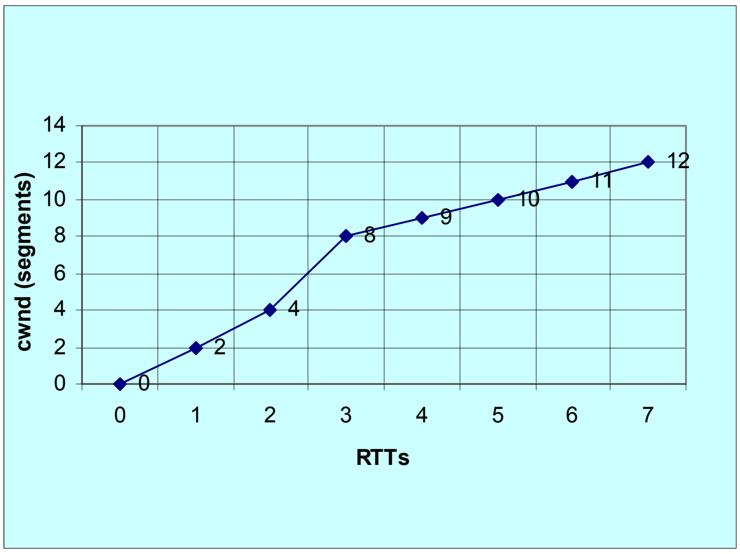

FIGURE 8. SLOW START.

In the graph above, it is assumed that a timeout occurred when cwnd had a value of 16 segments. Therefore ssthresh and cwnd get the values 8 and 1 respectively. Figure 8 shows what would take place afterwards. TCP sends one segment at time 0 and receives an ACK at time 1. cnwd is then set to 2 and two segments are transmitted. Two ACKs are received at 2. This indicates that cnwd can be increased to 4 and four segments are transmitted. At fourth RTT cnwd becomes equal to ssthresh and exponential increase is no longer possible. TCP now enters the congestion avoidance phase. The increase in cnwd would be 1 segment per RTT.

2.15 Fast Retransmit and Fast Recovery Algorithms

When TCP receives an out-of-order segment it sends a duplicate ACK immediately to the sender. This informs the sender the next sequence number expected and that an out-of-order segment has been received.

2.15.1 Fast Retransmit

If a segment is lost, on its way to the destination, all the other segments that follows would cause the receiving TCP to send duplicate ACK's. Normally TCP would wait for the timer to go off and then retransmit the missing segment. But *Fast retransmit* algorithm does not wait for the retransmission timer to go off, instead acts immediately by sending the missing segment immediately after receiving three or more ACKs in a row. It assumes that arrival of three or more duplicate ACKs in a row as an indication of packet loss.

2.15.2 Fast Recovery

If TCP begins with slow start each time a segment is lost, then it would affect the throughput considerably. Receiving duplicate ACKs is also an indication that data is still flowing between the end points. Therefore TCP avoids going back to slow start by performing *Fast recovery*. This means that TCP enters the congestion avoidance phase and starts form there.

2.16 Bandwidth Delay Product

Bandwidth –delay product for a connection is given by:

capacity (bits) = bandwidth (bits/sec) * round –trip time (sec)

Capacity in the formula above is the minimum size of the window receiver should have to get maximum throughput for the connection. It is dependent on both the bandwidth and the round trip time between the two ends.

3. ATM

ATM (asynchronous transfer mode) is the name given to a high-speed connection-oriented network technology, some times referred to as cell relay. An ATM cell is 53 bytes long. The cell contains 5 bytes of header followed by 48 bytes of data. ATM can switch data at gigabit speeds. ATM supports a variety of applications including real-time audio and video as well as data communications.
The ATM forum has defined several service categories for ATM. Those are Constant Bit Rate (CBR), Variable Bit Rate (VBR), Available Bit Rate (ABR) and Unspecified Bit Rate (UBR). The type of service category used depends upon the application requirements.
Figure 9 Shows the ATM protocol reference model [7]. The physical layer adapts ATM cells to the transmission environment. The ATM layer is responsible for multiplexing and switching the cells. The adaptation layer adapts the information flows to suit the ATM.

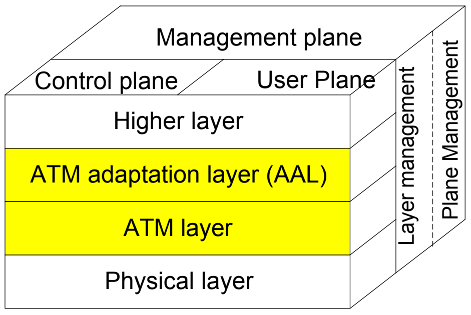

FIGURE 9. ATM PROTOCOL MODEL

3.1 Logical Connections of ATM

A Virtual Channel Connection (VCC) is the basic unit of switching in ATM. A VCC is setup between two end users and data is exchanged between them through this VCC using ATM cells. The VCCs that have the same end points are put together to form a Virtual Path Connection (VPC). A VCC is set up only if sufficient network resources are available to guarantee the quality of service (QoS) requested by the user. A user can negotiate traffic parameters such as Cell Loss Ratio (CLR), Cell Delay Variation (CDV), Peak Rate (PR), burstiness etc., with the network. Network may monitor a connection to ensure that it does not violate the agreements negotiated. In extreme situations an existing connections might be terminated.

3.2 ATM Adaptation Layer (AAL)

TCP/IP is designed for connectionless networks. ATM is by contrast connection oriented. In order to run TCP/IP over ATM, a mechanism must be there for integrating these connection oriented and connectionless worlds. For this purpose an ATM adaptation layer (AAL) is used. This layer is necessary to support the information protocols not based on ATM. The AAL adapts the data from the layer above to suit the ATM layer.

ITU-T has grouped all traffic flows into four service class categories. These service categories depends upon three main factors:

- *Synchronization between the sender and the receiver*
- *Constant or variable bit rate*
- *Connection oriented or connectionless mode*

The type of AAL protocol used at the AAL layer depends on the application requirements. Table 1 shows the relationship between the application requirements and the type of AAL used [7,8].

	Class A	Class B	Class C	Class D
Synchronization between source and destination	Required	Required	Not required	Not required
Bit rate	Constant	Variable		
Connection mode	Connection oriented			Connectionless
AAL protocol used	AAL-1	AAL-2	AAL3/4,AAL-5	AAL3/4

TABLE 1. THE RELATIONSHIP BETWEEN SERVICE CLASSIFICATION AND AAL PROTOCOLS.

We are concerned only with data transfer applications in our experiments. For data transfer applications, flows are made up of variable length data streams in connection oriented mode. These data flows does not require any synchronization between the source and destination. Therefore the AAL protocol suited for data transfer applications is AAL-5 and we limit our discussions only to that. RFC 1483[11] describes encapsulations methods for carrying network interconnect traffic over ATM AAL5.

The AAL5 can accept or deliver packets of varying size upto a maximum of 65535 bytes. This implies that an entire IP datagram can be transferred across an ATM network. But RFC 1483 standard restricts the maximum size of IP datagram that can be sent over ATM to 9180 bytes.

As the data being transferred may not fit exactly to a number of ATM cells, data has to be segmented on transmission and reassembled at the reception. [7] lists the services provided by ATM as:

- Handling of transmission errors
- Segmentation and reassembly
- Handling of lost and misinserted cells
- Flow control and timing control

AAL-5 consists of two logical sublayers:

- The Segmentation and Reassembly Sublayer (SAR):
  This layer is responsible for segmenting the user data received from CS to fit the ATM cells for transmission and reassembling cells received from the ATM layer before being sent to the user application. SAR also provide padding for the cells that does not have data to fill it.

- The Convergence Sublayer (CS): CS can provide end-to-end synchronisation, error handling, etc. This layer provides functions to support specific applications.

3.3 TCP/IP over ATM - AAL-5 adaptation layer

RFC 1483 [11] explains two encapsulation methods for carrying network traffic over ATM AAL-5 layer. The task of this layer is to provide functions to transfer data in connection oriented mode. One encapsulation method allows multiplexing of multiple protocols over a single virtual circuit. The other allows only one protocol over a single virtual channel. Figure 10 shows the operation of AAL-5[8].

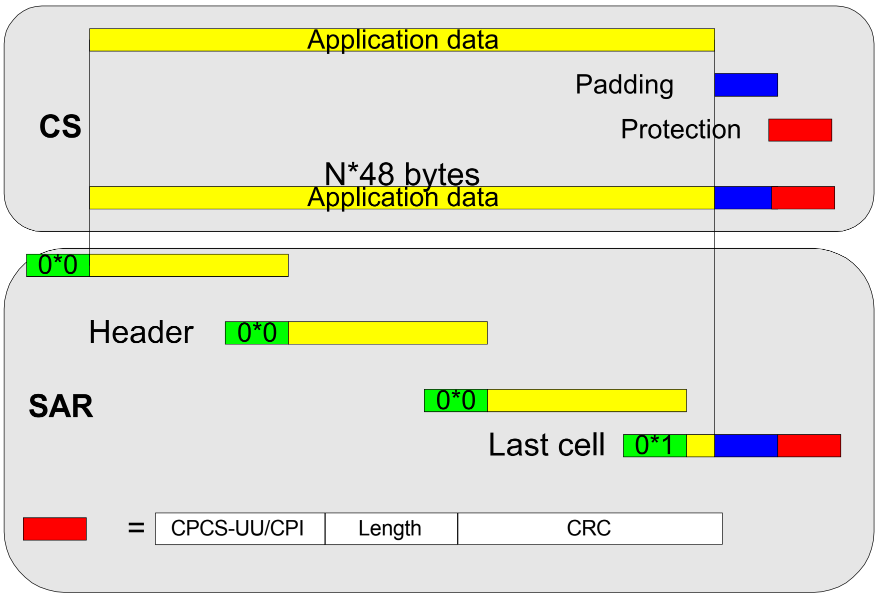

FIGURE 10. AAL-5 OPERATION

AAL-5 adds padding and a protection field to data sent by a user. The padding can vary from 0 byte to 47 bytes. The purpose of this is to obtain an integer number of ATM cells. The protection field consists of the following four fields [7]:

- Length field (16 bits), which indicates the size of the application data
- CRC (32 bits), which is used to discover errors in the received data
- User-to-User indication (CPCS-UU)which is used to transfer user information
- Common part indicator (CPI), which is used to interpret rest of the fields in the header.

As shown in the Figure 13, data occupies the ATM cells except for the last cell. Therefore AAL-5 cannot provide any indication of start and end of a data segment in individual cells. The PT field of the last cell of a data segment is set to 1 and this allows the receiver to differentiate two data segments. The receiver assumes that the cells preserve the sending order. It is not possible to interleave cells from different data segments, since there is no way in distinguishing them.

3.3.1 Encapsulation Methods

Ethernet encapsulation

This is the most widely used encapsulation method in local area networks. Ethernet uses 48-bit addresses. Figure 11 below shows the Ethernet encapsulation.

	destination address	source address	type	data	CRC
bytes	6	6	2	46-1500	4

FIGURE 11. ETHERNET ENCAPSULATION

The source and the destination addresses in the above are 48-bits long. The type field identifies the type of data that is included in the Ethernet frames. There are three types of data possible here: IP datagram, an ARP request/reply and a RARP request/reply. The Cyclic Redundancy Check (CRC) field is there to detect errors in the Ethernet frame. RFC 894 explains the Ethernet encapsulation.

IEEE encapsulation

The IEEE 802 encapsulation format shown in Figure 12 below, also uses 6 bytes for the source and destination addresses. As in the previous case it is used to carry three types of data. These are IP data, ARP and RARP data. The length field indicates the length of the
data to come (upto CRC).

destination addr	source addr	length	DSAP	SSAP	cntrl	org code	type	data	CRC

FIGURE 12. IEEE 802.3 ENCAPSULATION

3.4 Traffic and Congestion Control in ATM

The traffic control in ATM is based on the agreement that a user makes with the network. Without traffic control the users may overload the network leading to congestion collapse.

The network turns down a request for a new ATM connection if it cannot support the QoS requirements of the user. The user must specify the traffic parameters such as Peak Cell Rate (PCR), Cell Delay Variance (CDV), Sustainable Cell Rate (SCR) and Burst Tolerance [12]. If there are enough resources then the user and the network may enter into a traffic contract. As agreed, the network provides the QoS requested by the user, provided that the user does not violate the agreement. Traffic control functions such as Connection admission control, Priority Control, Network resource management etc. are used by the network to provide the QoS for the ATM connection [7].

Congestion control functions such as Selective Cell Discarding and Explicit Forward Congestion Indication are used in situations where congestion occurs. This would allow the network to recover from the congestion state.

3.5 Traffic and Congestion Control Mechanisms

Performance of TCP connections over ATM networks depends on whether ATM level congestion control is used or not. Congestion control is critical in ATM and non-ATM networks. When two or more bursts arrive simultaneously at a node, the queues may not be large enough to accommodate all the cells that arrive at the node resulting in buffer overflow. Another contributing factor is the link speeds of the incoming and the out going interfaces. If the total input rate is larger than the output link capacity, congestion problem may arise. To avoid congestion at ATM cell level different congestion avoidance strategies have been adapted successfully. These methods have been tested in research environment and are yet to come in to real networks. Some of the methods that can be used to control flow at ATM level are explained below.

Credit based congestion control is one such method. In a credit based flow control network, a sender is allowed to send data only when it is known that enough buffer space is available at the receiving switch. Flow control mechanism here reserves buffer space in each switch for each VC and this information is sent towards the sender of data in the form of credits. Therefore no data will be lost due to buffer overflow in these networks, thus achieving the maximum possible throughput [1]. A problem with this mechanism is that it requires a high degree of cooperation among switches. Therefore how this mechanism works in a complex network is yet to be seen.

ATM with Partial Packet Discard (PPD): [1,2] With PPD, once the switch drops a cell from a packet, it drops the rest of the packet (excluding the last cell). However the performance improvement here is limited because the switch begins to drop cells only when the buffer is full. Therefore more than one packet may be damaged due to buffer overflow resulting in timeouts at the packet level. Another problem is that cells of the damaged packets in the switch buffer will be still injected to the already congested link.

ATM with Early Packet Discard (EPD): [1,2] This mechanism is similar to PPD but more refined. With EPD, the switch drops an entire packet prior to the buffer overflow. Here the switch decides whether to drop the entire packet when the first cell arrives. If the switch does not have enough buffer space to accommodate the entire packet, the entire packet is dropped. This mechanism has many advantages over the earlier mechanisms. EPD prevents transmitting corrupted cells to an already congested link. Because only one packet is discarded, TCP's fast retransmission can be used. This was not possible in PPD if the switch discarded more than one packet. Unlike in Credit based method, EPD does not demand cooperation among the switches participating in the data transfer. EPD does not rely on end to end congestion control to prevent uncontrolled buffer overflows. As EPD drop only one packet at a time, TCP would react to this by reducing its congestion window, thereby reducing the rate of transmission to the network.

Experiments carried out with **Traffic Shaping** [3,4] have shown to improve the TCP performance even with significant differences in input/output link speeds at the ATM switch.

4. ADSL

ADSL stands for *Asymmetric Digital Subscriber Line*. This technology is intended for the last link between a customer's premises and the central office. ADSL transmits an asymmetric data stream, with a high bit rate channel (upto 8Mbps) towards the subscriber and much lower bit rate (upto 640Kbps) in the opposite direction. ADSL technique uses the existing Cu cables as the transmission medium.

It is estimated that there will be about 1 billion telephone lines at the end of this century. Therefore ADSL will have a great advantage over the other technologies. With down stream speeds as high as 8 Mbps, it is nearly 160 times faster than 56.6 modems. Another advantage is that it does not interfere with the existing telephone line, which implies that the telephone can be used and data can be transferred at the same time.
There are two types of modulation methods that are used by ADSL today. One is carrierless amplitude/phase (CAP) and the other is called discrete multi tone (DMT). ADSL upstream speeds range from 160 kbps to 640 kbps [14]. An example of the ADSL system structure is shown in Figure 13. The subscribers are connected to the Digital Subscriber Line Access Module (DSLAM), which in turn connects the subscribers to the ATM backbone.

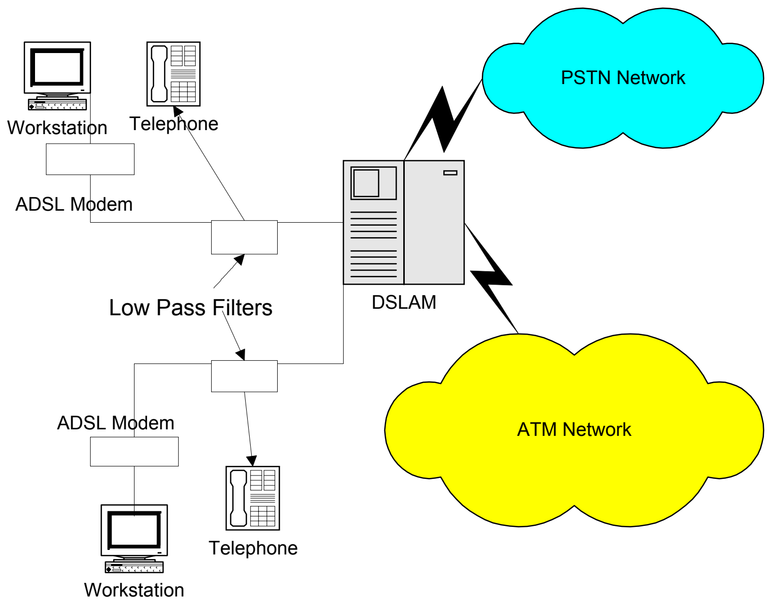

FIGURE 13. ADSL SYSTEM STRUCTURE

5. The Performance Testing Tool

The main goal of the project was to study the behavior of TCP over ADSL/ATM. Throughput measurements were to be taken in multiple connection, multiple hosts environment. The first column of Table 2 shows the basic properties that a testing tool should have in order to carry out the experiments successfully.

There were many performance evaluation tools available in the market. We have looked at three of them, namely, DBS, Netperf and NetSpec [5]. Netperf and NetSpec covered a part of the requirements set by us, whereas DBS covered all the requirements. Therefore we selected DBS as our testing tool.

Conditions	DBS	Netperf	NetSpec
Throughput measurements in TCP, UDP possible?	Yes	Yes	Yes
Multiple connections between two hosts possible?	Yes	No	Yes
Multiple connections between multiple hosts possible?	Yes	No	Yes
Can traffic pattern be specified?	Yes	Yes	Yes
Can throughput be measured continuously in a given time interval?	Yes	No	No
Can starting time, duration be specified for each connection?	Yes	No	No

TABLE 2. COMPARISON OF DBS WITH OTHER TOOLS

5.1 The System Structure

The DBS application consists of the DBS controller, the DBS daemon and the DBS view. The DBS controller is the main part of the DBS application. It controls the experiment by carrying out the commands given in the scenario file (command file). The DBS daemon is responsible for sending and receiving network traffic according to the commands given by the DBS controller. DBS viewer is used to view the results after a successful completion of an experiment.

Figure 14 shows the system structure of the DBS application. The experiment scenario is written in the *DBS configuration file* using DBS command language. This includes information about starting times, information about measurement hosts, traffic pattern, and duration of experiment etc. This information is passed to the respective hosts by the *DBS controlling host*. Then the data transmission is carried out at the specified times and the pattern by the *DBS daemons* at the respective measurement hosts. Once the experiment is done, the results are collected by the controller, which can then be viewed using DBS view. The installation information of DBS is given in Appendix A.

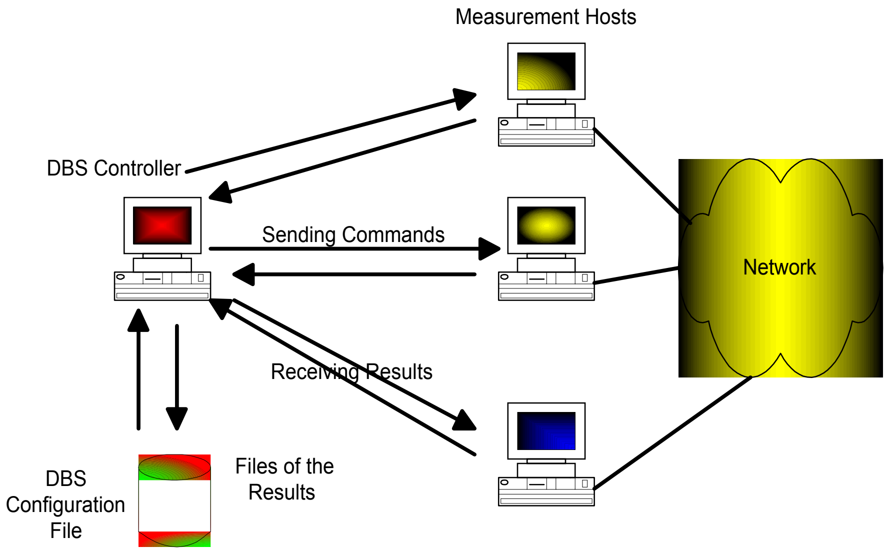

FIGURE 14. THE OPERATION OF DBS

5.2 Installation of Distributed Benchmark System

Installation of DBS is explained in Appendix A. In our test environment DBS deamon was installed in the client and the server computers. The DBS controller was installed in the server computer.

6. Performance Tests and Results

The tests described here have been performed at the Telia's research lab in Farsta. The software tool used for performance testing was DBS. This application allows us to generate TCP traffic between the client and the server machines and as well as to measure the throughput.

In each of the experiments discussed in the project except the last, the senders and the receivers buffer sizes were set to 9000 bytes. These numbers were selected so that they reflect the TCP buffer sizes of the home PCs. These experiments were supposed to give an understanding about the performance that a Telia's customer would get in the future with an ADSL system.

For the last set of tests, the sender and the receiver buffers were set to a size of 32000 bytes. This was done in order to investigate the impact of the buffer sizes in TCP performance.

6.1 The Goals of the Tests

The goals of the tests were:

- To study the behavior of TCP over ADSL/ATM. To investigate the parameters that affects the performance of TCP over ADSL/ATM.

- To study a performance evaluation tool that can be used to measure performance in the future.

Five types of tests were carried out:

Throughput measurements were taken by varying downlink/uplink speeds of the ADSL modem and by varying TCP segment sizes. These test results are described in section 6.3.

Throughput measurements were taken for different configurations of the ADSL modem with various cell losses introduced at ATM level. These results are explained in section 6.4.

Measurements were taken as in the case above, but with delay instead of cell losses. These test results are described in section 6.5.

Throughput measurements were done by combining cell loss and delay. These test results are described in section 6.6.

The final set of tests was done by increasing buffer size of the sender and receiver to 32000 bytes. These results are described in section 6.7.

Limitations:

When congestion occurs at a node (ATM switch) in an ATM network, the congested node drops ATM cells that have low priority. If these cells carry TCP/IP data, then the IP datagrams corresponding to these cells will be lost. If all the lost ATM cells belongs to one IP datagram, only one IP packet will be lost in the network and the receiver can use fast retransmission to recover the lost packet. Otherwise the sending TCP will timeout causing a considerable reduction in throughput. In our test environment we cannot model multiple cell losses.

6.2 The Test Environment

Figure 15 shows the test environment. The data flow between the server and the client is generated by the DBS application. The characteristics of the data flow depend upon the scenario information given in the DBS command file. The parameters that are set here are the size of data and the buffer sizes of the sender and the receiver. Data is transmitted from the server to the client through the network shown in figure 15. The traffic to and from the client is regulated by configuring the ADSL modem.

Data generated by the DBS application at the server machine is transmitted to Netedge via an Ethernet connection of 10Mbps. Netedge is configured to act as a router. At the Netedge, IP packets are converted to ATM cells and these cells are further directed through a ATM switch (APEX), through Digital Subscriber Line Access Multiplexer (DSLAM), and finally through ADSL modem to the receiver. At the ADSL modem ATM cells are converted back to IP packets and these are transmitted through an Ethernet link to the client machine.

Packet losses are introduced to the network by a HP instrument.

The client and the server computers are Intel-based computers running Linux.

For each configuration of the test setup, the measurements were done only once.

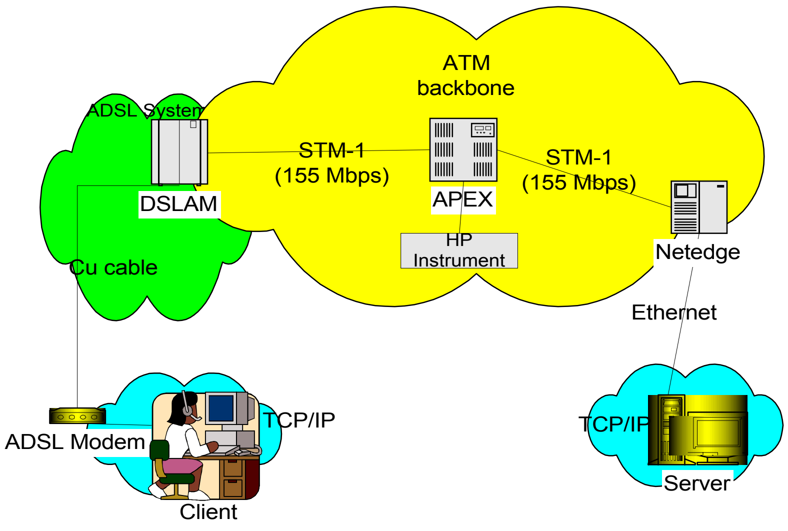

FIGURE 15. THE TEST ENVIRONMENT.

6.3 Throughput Measurements for Different Configurations of ADSL Modem and Data Sizes

The objectives of the following tests were to investigate the relationship that throughput has to downlink, uplink speeds and the data size.

Tests were carried out for various combinations of downlink and uplink speeds of the modem. Downlink speed of the modem was varied from 8Mbps down to 1Mbps at 1 Mbps intervals. Uplink speed was varied from 800kbps down to 200kbps at 200kbps intervals. The TCP data sizes used for the tests were 64, 128, 256, 512, 768, 1460 and 2048 bytes.

For each of the above combinations of downlink/uplink capacities, throughput measurments were carried out for various data sizes as indicated above. A total of 224 tests were done. The results are given in tables 1 to 10 in appendix B.

6.3.1 Throughput Measurements with Different Downlink Speeds

The x-axis of Figure 16 corresponds to the data sizes (in bytes) used in the tests and the y-axis shows the resulting throughput in bits per second. Different curves have been obtained for various combinations of downlink and uplink speeds. Here the uplink speed is kept at 800kBps while downlink speed is varied.

Figure 16 shows that the maximum throughput that can be achieved is about 3.5Mbps. This is obtained for TCP data sizes of 1460 bytes. The minimum ADSL configuration that gives this maximum throughput is 5Mbps down link and 800kbps uplink speeds. With a data size of 1460 bytes, the corresponding length of the IP packet becomes 1500bytes. This is the maximum packet size of IP that is possible for an Ethernet connection. Therefore the maximum throughput is possible at this data size. For data sizes greater than 1460 bytes, fragmentation would occur and hence reduction in throughput as a result. This is also seen in Fig.16.

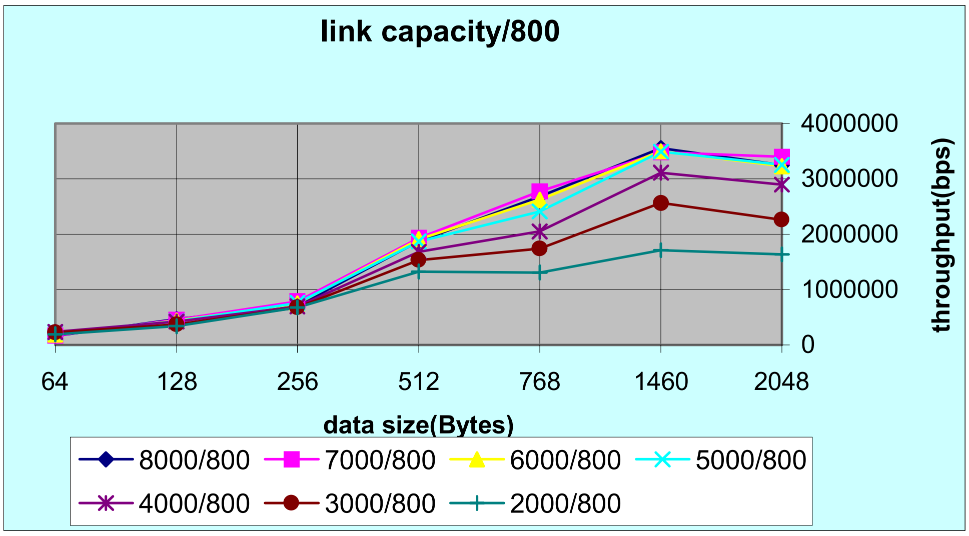

FIGURE 16. THROUGHPUT FOR VARIOUS DOWNLINK CAPACITIES OF ADSL.

For down link speeds of 5Mbps and upwards (i.e. 6, 7, 8Mbps), throughput values are almost identical and therefore for the next set of experiments we would consider only 5Mbps down link speed.

6.3.2 Throughput Measurements with Different Uplink speeds

The x-axis of Figure 17 corresponds to the data sizes used in the tests and the y-axis shows the resulting throughput in bits per second. Different curves have been obtained for combinations of downlink and uplink speeds. Here the downlink speed is kept at 5000kbps while uplink speed is varied as shown in the graph.

Figure 17 shows the throughput curves obtained for a downlink speed of 5Mbps and different uplink speeds and data sizes. One observation here is that for uplink speeds of 640 and 576 kbps, the throughput values are almost identical. This implies that the uplink is no longer a bottleneck for uplink speeds greater than 576kbps. But for uplink speeds of 192kbps and 384kbps, the throughput comes down to about 3Mbps, which is slightly less than the maximum throughput that was possible with higher link speeds. This is because the uplink becomes a bottleneck for the acknowledgements sent for the received data.

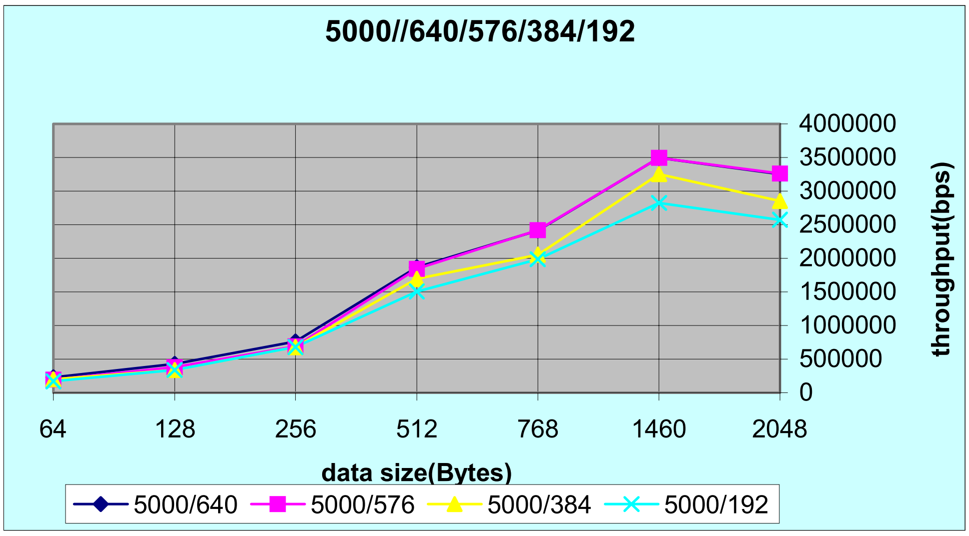

FIGURE 17. THROUGHPUT FOR DIFFERENT UPLINK SPEEDS.

6.3.3 Throughput Measurements for Data Size of 1460 bytes with Different Uplink speeds

The purpose of this experiment is to investigate the occurrence of bottlenecks.

The x-axis of Figure 18 corresponds to the uplink speeds used in the tests and the y-axis shows the resulting throughput in bites per second. Downlink capacity is kept at 5000kbps and a data size of 1460bytes is used for all the tests. Only the uplink speeds are varied.

As mentioned in the previous section, the throughput is almost equal for uplink speeds of 576 kbps and 640 kbps. Uplink speeds of 192kbps and 384kbps creates a bottleneck for the acknowledgements sent in that direction. Therefore a reduction in throughput is seen in Figure 18 for the said uplink speeds.

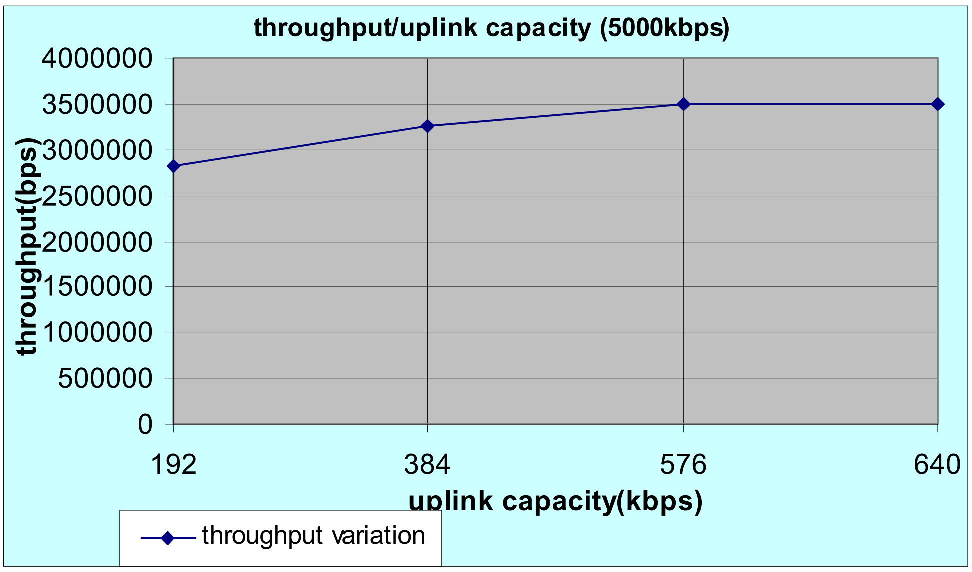

FIGURE 18. THROUGHPUT FOR DIFFERENT UPLINK CAPACITIES.

6.4 Throughput Measurements with Cell Losses

In this set of experiments we introduce cell losses at the ATM level. A HP instrument is used for this purpose. The data stream is now lead through the HP Instrument and it is configured to create the required cell losses with uniform distribution.

Figure 19 shows throughput values after cell losses were introduced. The x-axis of Figure 19 corresponds to the data sizes used in the tests and the y-axis shows the resulting throughput in bits per second. Different curves have been obtained for combinations of downlink and uplink speeds with a cell loss of 10E(-3) [read as one cell loss per thousand ATM cells]. Here the downlink transmission speed is kept at 5000kbps while uplink transmission speed is varied as shown in the graph.

The experiments without cell loss in the last section resulted in a throughput of 3.5Mbps. But with cell losses introduced, the throughput has been reduced to about 1.5Mbps. When a packet loss is experienced by the sending TCP, it would reduce the transmission rate. This can be given as the reason for throughput reduction.

The given level of cell loss (10E(-3)) at ATM level causes roughly an IP packet loss for every 32 IP packets sent. This implies that the sending TCP is unable to keep its maximum transmission rate for more than few RTTs (around 4RTTs), before it is interrupted either by timeout or duplicate acknowledgements. If the interruption was due to the retransmission timer going off (this implies that more than one packet is lost), then TCP would start transmission again using slow start algorithm [6]. Otherwise TCP would start with fast recovery. Whatever the case above, there would be a considerable throughput reduction. Please see sections 2.14 and 2.15.2 for more details.

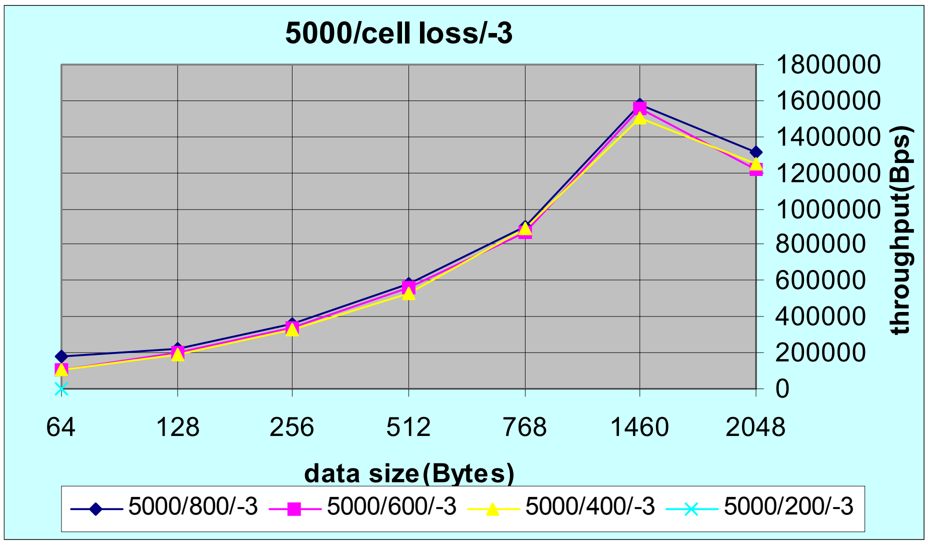

FIGURE 19. THROUGHPUT WITH CELL LOSSES.

In a real network, a node will drop many ATM cells in a congestion situation and these cells may belong to several IP packets. TCP recovers well only if one packet is lost. Otherwise it will timeout leaving a lot of bandwidth unused while it times out. [1,2] Explains strategies to control flow at ATM level.

6.4.1 Throughput Measurements with Varying Cell Losses

In this chapter we study how varying cell losses influence the throughput. For each cell loss value, a set of tests was carried out. The cell loss values used for the tests were 10E(-3), 10E(-4), 10E(-5), and 10E(-7).

The results of the tests are shown in Figure 20. The x-axis of Figure 20 corresponds to the packet sizes used in the tests and the y-axis shows the resulting throughput in bits per second. A downlink speed of 5000kbps and uplink speed of 640kbps is maintained for all the tests. Different curves in Figure 20 show the throughput measured for various cell loss values.

It was explained in the previous section why a cell loss of 10E(-3) resulted in very poor TCP throughput. Figure 20 shows that a cell loss of 10E(-4) or less does not lead to a bad performance. It was also mentioned in the previous section that a cell loss of 10E(-3) at ATM causes a retransmission of one IP packet for every 32 IP packets (roughly), but with a cell loss of 10E(-4), a retransmission would occur for every 320 IP packets. The sending TCP in the latter case can transmit its maximum allowable window for a reasonable amount of time (a minimum of 50RTTs) before retransmission timeout or duplicate acknowledgements interrupts it.

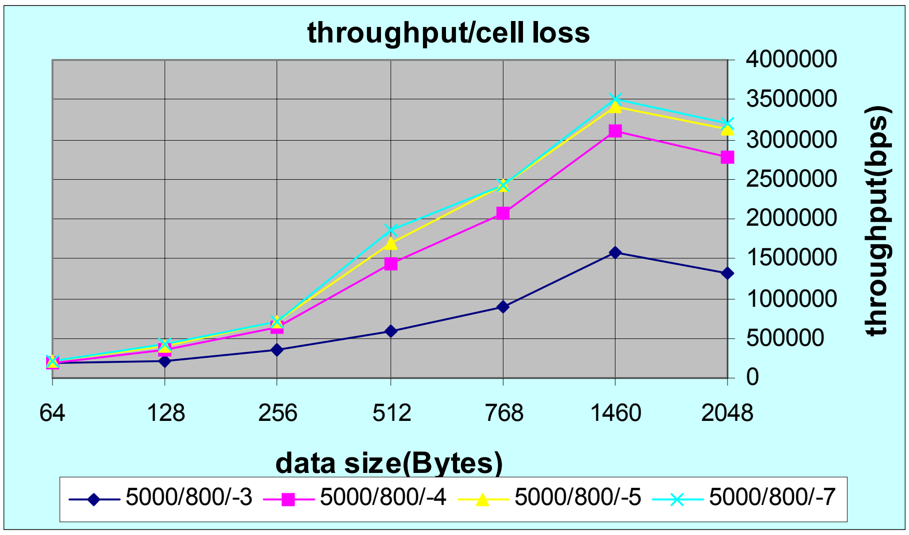

FIGURE 20. THROUGHPUT MEASUREMENTS WITH VARYING CELL LOSSES.

6.5 Throughput Measurements with Delay

How is throughput influenced by delay?

According to the bandwidth-delay product formula (see section 2.16) a delay in the network would lead to a decrease in throughput. The relationship between the bandwidth and the delay resulted from the experiments can be seen in Figure 21. The x-axis of Figure 21 corresponds to the packet sizes used in the tests and the y-axis shows the resulting throughput in bits per second. A downlink speed of 5000kbps and uplink speed of 640kbps is maintained for all the tests. Different curves in Figure 24 shows the throughput measured for various cell delay values.

The HP instrument created cell delays of 2ms, 4ms, 6ms, 8ms and 10ms. A delay of 2ms causes a reduction in throughput by about 300kbps. But a delay of 10ms causes a throughput reduction of about 1Mbps.

The results show that an increase in delay in the network leads to a reduction in the throughput.

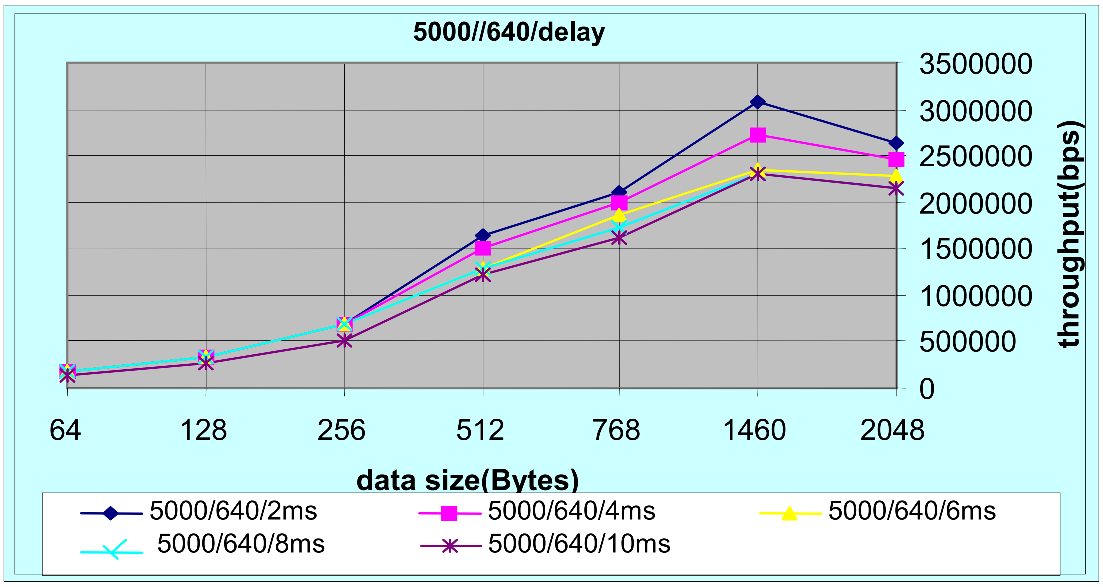

FIGURE 21. THROUGHPUT WITH DELAY.

6.6 Performance Measurements with Cell Loss and Cell Delay

In congestion situations, cells would be delayed due to the queuing at buffers, and cell losses would occur due to buffer overflows. Therefore the objectives of the experiments in this section were to investigate the behavior of TCP in the presence of both cell losses and delays.

Figure 22 shows the results of the tests carried out by combining cell loss and cell delay. The x-axis of Figure 22 corresponds to the data sizes used in the tests and the y-axis shows the resulting throughput in bits per second. A downlink speed of 5000kbps and uplink speed of 640kbps is maintained for all the tests. Different curves in Figure 22 shows the throughput measured for various cell losses (10E(-3), 10E(-5)) in combination with a cell delay of 4ms.

The throughput measurements show that a combination of delay and cell loss would lower the throughput by a considerable amount. The cell loss has a stronger contribution to the throughput reduction than cell delay.

Figure 22 also show that the network should not be loaded to such an extent where cell losses of 10E(-3) would occur.

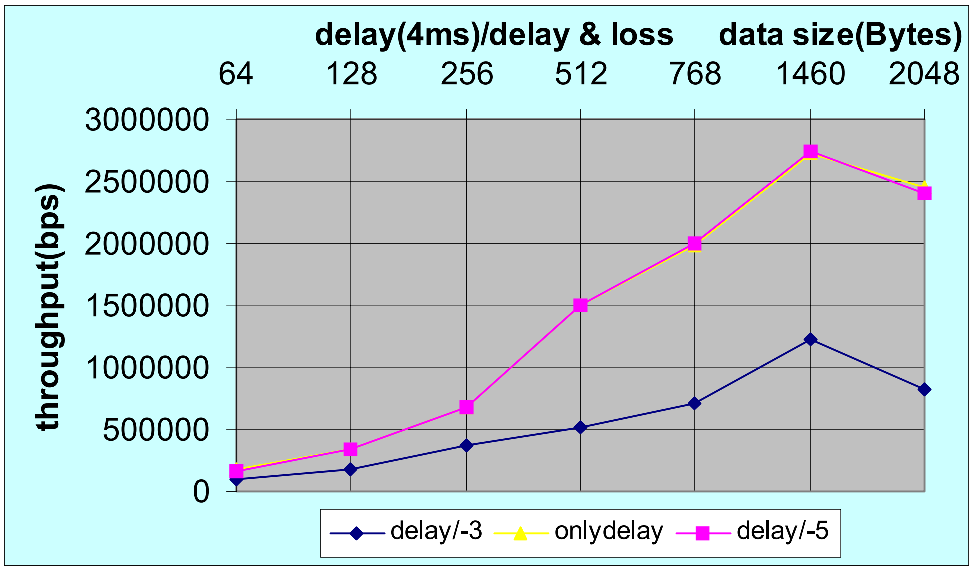

FIGURE 22. THROUGHPUT WITH CELL LOSSES AND DELAY COMBINED.

6.7 Performance Measurements with Varying Buffer Sizes

In all the previous tests that were carried out, the sender and the receiver had a buffer size around 9000 bytes. For the tests in this section a buffer of 32000 bytes has been used for both the sender and the receiver.

Figure 23 shows the results of the experiments carried out with the new buffer sizes. The x-axis of Figure 23 corresponds to the packet sizes used in the tests and the y-axis shows the resulting throughput in bits per second. A downlink speed of 5000kbps and uplink speed of 640kbps is maintained for all the tests. Different curves in Figure 23 shows the throughput measured for buffers of 9000 bytes and 32000 bytes respectively.

Results show that the throughput increased from 3.5Mbps to about 4.3Mbps with the new buffer sizes. This increase can be explained by using the bandwidth delay product formula given below.

capacity (bits) = bandwidth (bits/sec) * round –trip time (sec)

When capacity on the left-hand side of the equation (bandwidth-delay product) is increased, for the equation to hold bandwidth should increase.

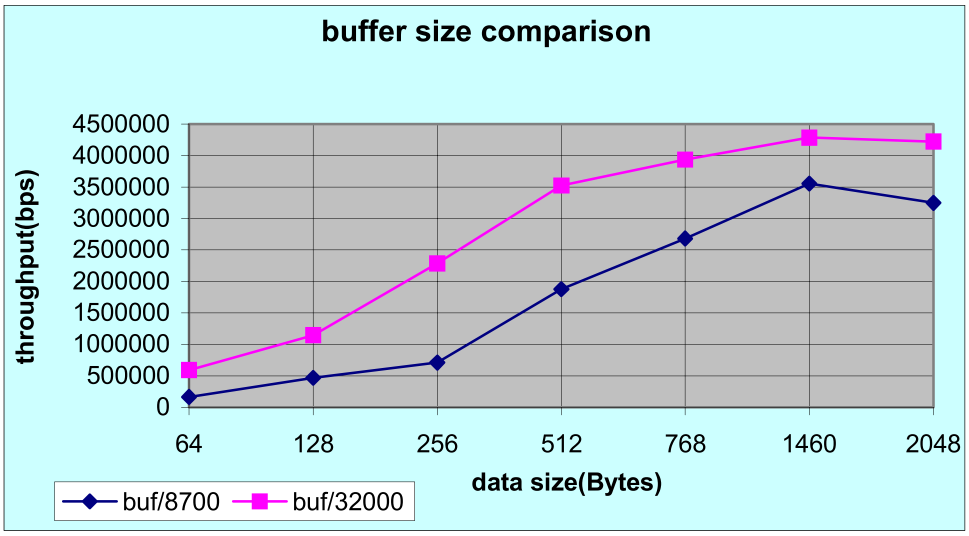

FIGURE 23. THROUGHPUT WITH A LARGE BUFFER.

There have been studies [3, 4, 5] addressing the relation between the performance and the buffer sizes. These studies have noted performance improvements with increasing buffer sizes. Experiment conducted in [3,4,5] in WAN environment shows that large TCP windows contributed to greater TCP performance. Here the buffer sizes were varied from 0.5k to 128k and the resulting throughput varied from 0.47 Mbps to 119 Mbps.

6.8 Test Summary

The main objectives of the tests were to investigate the behavior of TCP over ATM/ADSL. Tests show that the maximum throughput that can be obtained depended upon the parameters such as; packet size, cell loss in the network, cell delay, uplink and downlink transmission speeds, and the buffer sizes of the sender and the receiver.

The maximum throughput is obtained when the data segments had a size of 1460 bytes. This is because Ethernet was used at the two ends. (Beyond this segment size fragmetation would occur)

A cell loss of one ATM cell in one thousand ATM cells resulted in a very low throughput. But cell losses of 10E(-5) or more did not affect the throughput very much. A cell loss rate of 10E(-3) in ATM level is roughly equivalent to a TCP level packet loss for every 32 Packets. This implies TCP is in the congestion avoidance phase most of the time and thereby resulting very low throughput.

Delay has a negative influence on the performance. Delays of 2ms or 4ms did not affect the throughput very much. But if it was more than that value, the throughput was reduced by a considerable amount. This can be partly explained by the bandwidth delay product formula explained in section 2.16.

The default buffer size of the home PCs lie around 9000 bytes. That was the reason why buffer sizes around 9000 bytes were selected for the tests. A throughput of 3.5Mbps was possible with this buffer size. The throughput increased to 4.3Mbps just by increasing the buffer to 32000 bytes. This shows that the default buffer size around 9000 bytes for both sender and receiver is not an optimal value.

The tests also show that the uplink can be a bottleneck. The acknowledgements sent for the received data is “blocked” due to the low uplink speeds.

Although we can draw the above conclusions from the tests we have done, there are some questions that are left unanswered. Much time was spent on selecting a test tool, installing it and in learning how to use it, initial test plans had to be limited. Initial plans to measure throughput by using different ATM service classes and functions such as shaping at the routers had to be left for future work.

The DBS application program that was selected to measure the throughput turned out to be a powerful tool. DBS tool is easy to use and therefore Telia should be able to use it to measure performance without any difficulty in future tests.

7. Future Work

As DBS tool can be used for the throughput measurements and that the knowledge about installing and using the tool is available now, one could extend the current project to cover a greater area without much difficulty. The following can be suggested as the starting point.

1. The thesis focused on the behavior of TCP in an environment with only one server and one client. This can be easily extended to multiple server, multiple client connections. It would be interesting to investigate the behavior of TCP in a network environment that resembles a real network.

2. To investigate the behavior of TCP as in the above environment with traffic shaping function at the routers.

3. To investigate the behavior of TCP as in 1 and 2, with different ATM traffic classes.

4. To perform all the tests as above in a real network environment using more realistic cell loss models.

8. Conclusions

The primary goal of the thesis project was to investigate the behavior of TCP over ADSL/ATM.

The report contains descriptions of the technologies involved, namely TCP/IP, ATM, and ADSL. It also gives an introduction to the test tool DBS and presents performance evaluation results obtained using this test tool.

The technologies TCP/IP, ADSL and ATM are of great interest to an Internet service provider like Telia. These technologies have the following advantages from Telia's point of view:

TCP/IP – Most widely used protocol for data transmission.

ATM – Telia has already invested heavily in an ATM network that covers the whole country.

ADSL – gives about 150 times more data transmission speeds than the existing modems. Do not need any infrastructure investments as ADSL uses existing telephone lines.

We have looked at some of the parameters of TCP and the underlying network. Tests show that the throughput depends very much on the data size. Smaller data sizes resulted in lower throughput and the maximum throughput was obtained for packets with a data size of 1460 bytes.

We introduced congestion to the network by using a HP instrument. The cell losses and the delays created by the HP instrument can be thought as congestion created due to high network load in a real network. Throughput measurements were taken by varying the cell losses and cell delays. A cell loss ratio of one to thousand or ten thousand, and cell delays above 4ms resulted in poor throughput. The reason for that is the ATM cells that carry TCP/IP data may belong to one or more IP datagrams. If all the cells that are lost belong to one IP datagram, then TCP/IP can use the fast retransmission mechanism to recover the lost data. But if the lost cells are spread among several IP datagrams, then TCP will timeout resulting low throughput. Therefore over booking of ATM networks that leads to cell losses and cell delays as above should be avoided. Experiments [3,4] done by introducing congestion control at ATM level has shown to improve TCP performance to very good levels. The matter has been left for further studies.

By increasing the buffer size of the sender and the receiver to 32000 bytes rather than the default value of 9000 bytes, the throughput increased to 4.3Mbps from 3.5Mbps. Large buffers should be used in order to get a better throughput.

The results show that the TCP throughput depends on many factors. TCP throughput can be improved by two possible ways; either by implementing congestion control mechanisms such as PPD, EPD, Traffic shaping etc., at the network level or by selecting the appropriate TCP parameters such as packet size, buffer size etc., or by combining both.

It is reasonable to state that TCP/IP over ADSL will remove the last bottleneck for high-speed access to the Internet.

9. References

[1] Kung, H T. And Morris, Robert. Impact of ATM switching and flow control on TCP performance: Measurements on an experimental switch. GLOBECOM'95, November 1995.

[2] Romanow Allen. And Floyd Sally. Dynamics of TCP Traffic over ATM Networks. Computer Communications Review, vol 24 no 4 pp 79-85, October , 1994.

[3] Lindberg, G., Olsson R., And Ståhl, G. Assorted experiments with TCP/IP over ATM wide area links. Report from COAST phase 1, September 1995.

[4] Benjamin, J. Ewy, Evans, B. Joseph et. Al. TCP/IP Experiences in the MAGIC Testbed. University of Kansas Technical report, 1994.

[5] Murayama, Y. Yamaguchi, S. DBS: a powerful tool for TCP performance evaluations, Graduate School of Information Science, Nara Institute of Science and Technology, Japan.

[6] Stevens, W. Richard. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley, 1997.

[7] Stallings, W. Data and Computer Communications. Prentice Hall, 1977.

[8] Boisseau Mark, Demange Michel and Munier Jean-Marie. ATM Technology: An Introduction. Thompson Computer Press, 1996.

[9] Manthorpe Sam. Implications of the Transport Layer for Network Dimensioning. Lausanne, EPFL 1997.

[10] Laubach, M. Network Working Group. Classical IP and ARP over ATM. RFC 1577, January 1977.

[11] Heinanan, J. Multiprotocol Encapsulation over ATM Adaptation Layer 5, RFC 1483.

[12] Comer Douglas. Internetworking with TCP/IP, Volume 1: Principles, Protocols and Architecture, Prentice Hall, 1995.

[13] Jain, R. Siu Kai-Yeung. A Brief Overview of ATM: Protocol Layers, Lan Emulation, and Traffic Management. Dept. Of Computer and Information Science, Ohio State University technical report, 1996.

[14] A 1000 ADSL- " Product Information manual ", ALCATEL.

Appendix A

Installing DBS

DBS is free software and can be run on various platforms. The home page for DBS is found at http://www.ai3.net/products/dbs. DBS dbs-1.1.5 version was used for the tests at Telia. The application program is provided as a tar.gz file.
Copy the file dbs-1.1.5.tar.gz, where you would like DBS directory to appear and decompress it using gunzip.
gunzip dbs-1.1.5.tar.gz

Then extract the directory and the files using tar.

tar xf dbs-1.1.5.tar

This will create a directory called dbs-1.1.5, which contains all the necessary DBS files.

A README file is also created in DBS directory containing this installation information. Installation can be done using install script.
To compile the application tool enter the dbs-1.1.5 directory and compile using *make* command. This is done using:
Type *make* in the source directory
Modify ‘CFLAGS LDFLAGS as follows
CFLAGS = -O –I/usr/include/libelf
LDFLAGS = - insl -lelf

Once the system is compiled there would be three programs that are needed to run DBS. The dbsd has to be installed in all the measurement hosts and dbdc and dbs_view in the controlling host. The application is ready to run at this stage. Study the examples in the Examples directory.

Appendix B

Tables 1-10 in Appendix B shows the results obtained for the experiments carried out in the project.

		Table 1		
Capacity Up	7296	7392	7136	7328
Capacity Do	640	576	384	192
Packet Size				
64	164800	169600	169400	169400
128	468200	379300	339400	339400
256	706700	732200	677700	677700
512	1873100	1915400	1881300	1837600
768	2681800	2570300	2562600	2430000
1460	3556100	3452600	3500200	3441900
2048	3248600	3226400	3243800	3106500

		Table 2		
Capacity Up	6976	6944		
Capacity Do	640	576	384	192
64	170500	169500		
128	453200	403100		
256	788300	731600		
512	1933900	1916500		
768	2771400	2561000		
1460	3480900	3500500		
2048	3400200	3370600		

		Table 3		
Capacity Up	5984	5984		
Capacity Do	640	576	384	192
64	189400	182900		
128	437700	400400		
256	747000	711500		
512	1908300	1906200		
768	2620700	2563600		
1460	3495700	3495400		
2048	3228400	3260500		

		Table 4		
Capacity Up	4992	4992	4992	4992
Capacity Do	640	576	384	192
64	227800	193400	199500	169100
128	426600	379400	338900	338900
256	760500	688200	677700	677800
512	1867600	1843000	1691800	1504300
768	2412800	2416200	2053500	1987300
1460	3494100	3495000	3251100	2821100
2048	3249700	3260800	2852000	2571100

		Table 5		
Capacity Up	4000	4000	4000	4000
Capacity Do	640	576	384	192
64	233300	206400	169400	159700
128	419800	376900	338900	339400
256	697200	677900	678800	678800
512	1680800	1679800	1553300	1453600
768	2050000	2023300	1970800	1875100
1460	3111900	3112300	2954600	2708400
2048	2894700	2776400	2521600	2401900
		Table 6		
Capacity Up	2976	2976	2976	2796
Capacity Do	640	576	384	192
64	226600	200100	169400	169400
128	365500	351100	339400	339400
256	678800	678800	678800	678800
512	1529600	1454100	1423600	1132600
768	1737300	1737100	1670400	1635400
1460	2565700	2543100	2565800	2521100
2048	2258300	2208800	2173700	2067000
		Table 7		
Capacity Up	1984	1984	1984	1984
Capacity Do	640	576	384	192
64	194000	182800	169400	169700
128	339300	339400	339400	339400
256	678900	678900	678800	678700
512	1322800	1323200	1205800	1191800
768	1307900	1301500	1302700	1265100
1460	1709600	1707400	1709300	1709800
2048	1634000	1634800	1598000	1571200
		Table 8		
Capacity Up	992	992	992	992
Capacity Do	640	576	384	192
64	169700	170700	170700	170700
128	251000	290800	303700	310700
256	515800	650900	621200	539800
512	525900	745400	738600	719500
768	762200	798600	798700	789900
1460	811900	856200	856500	866700
2048	711100	841600	843000	842800

			Table 8.1			
packet size	Throughput	with cell	loss	minus 3	minus 4	minus 4
	4992/640	4992/576	4992/384	4992/192	4992/640	4992/384
128	220200	199800	193300	178500	353400	315000
256	361600	343900	327000	334000	625800	630600
512	581800	564600	528700	545500	1428000	1414400
768	899900	866300	886800	815700	2073300	1936200
1460	1579200	1559100	1498500	1375500	3105800	2973500
2048	1314600	1220800	1248300	1234200	2785300	2489400

		Table 8.2			
packet size		minus 5	minus 7	minus 7	
	4992/640	4992/384	4992/640	4992/384	
64	217000	169400	211400	169100	
128	405400	308300	419500	338100	
256	701100	676100	706800	676300	
512	1686900	1677100	1865200	1686600	
768	2430300	2161900	2413700	2122800	
1460	3407400	3122400	3495700	3250400	
2048	3135700	2883100	3200000	2952700	

		Table 9.1			
			cell delay		
packet size	2ms	4ms	6ms	8ms	10ms
	4992/640	4992/640	4992/640	4992/640	4992/640
64	169400	169400	169400	169400	139400
128	338300	338900	339000	338800	273000
256	677900	677200	677700	677800	507500
512	1641100	1501700	1282400	1291900	1219100
768	2097400	1985700	1865500	1732100	1620000
1460	3079100	2720800	2339500	2293000	2308200
2048	2637200	2452700	2282500	2142200	2151900

			Table 9.2	
packet size	cell loss	minus 3	minus 5	minus 5
delay	4ms	8ms	4ms	8ms
	4992/640	4992/640	4992/640	4992/640
64	101200	115400	169100	169000
128	180200	211700	337900	323700
256	366100	361800	676400	640800
512	521000	489500	1500500	1295900
768	714600	700600	2007300	1742000
1460	1229900	1113600	2741500	2293000
2048	828500	843700	2404700	2152500

	Table 9.3	
	no delay,	no loss
packet size	buffer	32120
	down/up	4992/640
64		591900
128		1143400
256		2288300
512		3526000
768		3934600
1460		4282500
2048		4221900

	Table 10	
	down/up	4992/640
buffer	8700	32000
64	164800	591900
128	468200	1143400
256	706700	2288300
512	1873100	3526000
768	2681800	3934600
1460	3556100	4282500
2048	3248600	4221900