Switch Lab
Ericsson Telecom Aktiebolag

TCP Analysis and Simulation

Johan Gärde

Royal Institute of Technology, Sweden

Master's Thesis in
Electrical Engineering
April 1997

Supervisors:	Harald Brandt Zsolt Haraszti SwitchLab Ericsson Telecom AB	Lars Gauffin Vladimir Vlassov Department of Teleinformatics Royal Institute of Technology
Examiner:	Björn Pehrson Department of Teleinformatics Royal Institute of Technology	

Abstract

The purpose of this master's thesis was to highlight the issues of TCP over ATM. The results presented have been attained from simulations of TCP traffic using the UBR and ABR services with VBR background traffic.

The performance of the Tahoe and Reno implementations of TCP have been examined, revealing the known problems with TCP. The buffer size required to get high performance using the UBR and ABR services have been studied for various configurations. The importance of the use of a discard policy for connections using UBR is demonstrated, along with a comparison between the early packet discard and selective drop schemes.

Table of Contents

1 Introduction

This is the report of a master's thesis performed at SwitchLab, a network research laboratory within Ericsson Telecom AB. The purpose of the thesis was to highlight the issues of TCP over ATM, including the characteristics of TCP, ATM and the interaction between them. The two service categories of ATM examined in this report are the Unspecified Bit Rate (UBR) service and the Available Bit Rate (ABR) service.

The UBR service has no congestion control mechanism of its own, to control the flow of data. Instead, it relies on higher level protocols, such as TCP, to handle the flow control. The ABR service has a flow control mechanism, and it will be used in addition to the one provided by TCP. The use of the ABR service for TCP traffic have to be studied in order to assure that the two flow control mechanisms will be able to coexist and still maintain high performance.

TCP traffic using the UBR service may experience that the available bandwidth is shared among the sources in a very unfair manner. To improve the fairness, discard policies can be used at the switches in the network to discard cells from sources that use more than their fair share of the bandwidth. The discard policies are also able to remove corrupt packets from the network, increasing the available bandwidth for the usable packets.

This report cover studies of TCP and the use of the UBR and ABR services for TCP traffic. The means for doing these studies was by simulation in the PLASMA simulation environment developed by Ericsson. The tasks included in the thesis was to implement the appropriate components to study into the PLASMA environment, carry out the simulations and do performance evaluations. The results from the work done are presented in the following chapters.

Chapter 2 of the report is an introduction to TCP. The functionality and characteristics of the TCP flow control mechanism are described. The Tahoe and Reno implementations are examined and their flow control algorithms presented.

Chapter 3 is a similar introduction to ATM. The service categories of ATM are presented and the flow control mechanism of the ABR service is described. The explicit rate marking algorithm ERICA+ is examined closely. The reason for having a discard policy is explained along with the description of the early packet discard and the selective drop policies.

Further, the model used for the simulations is described in chapter 4. The default parameter settings and network configurations used for most simulations are listed.

Chapter 5 holds the results from the simulations. A number of characteristics of TCP are revealed, including such dependent on TCP only or the interaction between TCP and the ATM service categories UBR and ABR. The importance of the use of a discard policy is demonstrated, and the early packet discard policy is compared to the selective drop policy.

A summary of the conclusions made from the simulations and recommendations of areas for further study are found in the final chapters.

2 TCP

The main objectives of the Transmission Control Protocol (TCP) [4] is to deliver segments of data from one application to another in correct order and without loss. To accomplish this, end-to-end flow control is used. Both source and destination have a sliding window to limit the amount of segments sent by the source. Upon reception of a segment, the destination will return an acknowledgement to the source indicating the next segment expected in order to get a consecutive sequence of received segments. When an acknowledgement for the sent segments is received by the source, new ones may be sent. If an acknowledgement is not received within a specified period of time, the segment is retransmitted.

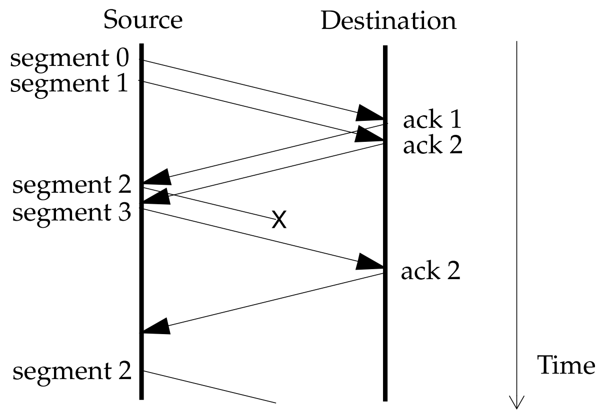


Figure 1: A TCP connection with a window of two segments. Segment 2 is lost somewhere in the network. When the acknowledgement, indicating that segment 2 is successfully delivered, has not been received within a specified period of time, segment 2 is retransmitted.

If there is no loss of segments, TCP will increase the window at the source, thus making it possible to send even more. As long as the window does not get bigger than the window at the destination, this will continue until the network becomes congested, resulting in lost segments. At this point, the window is decreased and the lost segments are retransmitted. These actions will be repeated throughout the duration of the connection. In this way TCP uses segment loss as a measure of the congestion.

The maximum size of each segment is decided upon setup of the connection. Both source and destination announce what they believe to be the maximum allowed segment size along the path through the network and the smallest size is used. The segment size announced is either set locally or found by path MTU (maximum transmission unit) discovery. The reason for selecting the smallest size is to avoid fragmentation of the segments. A larger segment size will decrease the relative overhead caused by the 40 byte TCP/IP header and generally give higher throughput.

Common segment sizes are 512 bytes, which is the default setting for many TCP implementations, 1460 bytes for Ethernets, 9140 bytes for pure ATM networks. They are often limited by the MTU of the link layer. Ethernets, for example, have a MTU of 1500 bytes. Subtract the 40 byte TCP/IP header and you get 1460 bytes as the largest segment possible to fit into one Ethernet frame.

There are a number of algorithms to control how and when to retransmit segments or change the size of the source's window. Out of the ones described below, "Slow start", "Congestion avoidance" and "Retransmit timeout" are required in a TCP implementation, whereas "Fast retransmit" and "Fast recovery" are optional. "Slow start" and "Congestion avoidance" control the growth of the source's window and the others are invoked at retransmissions.

2.1 Slow start

When a connection is established, TCP starts in the slow start state. The size of the source's window, called the congestion window (`cwnd`), is set to one segment. For each segment acknowledged, `cwnd` is increased by one segment. When the first acknowledgement is received, the size of `cwnd` becomes two segments, and when the acknowledgement for these two sent segments is received, `cwnd` becomes four segments and so on. This is an exponential increase of the window size, since `cwnd` is doubled each round-trip time as seen in step 1 of figure 2.

The size of `cwnd` is limited by the fixed amount of memory allocated for the TCP send window. Common sizes of the TCP send window are 4, 8 and 16 kBytes. Newer implementations of TCP often have a large default TCP send window, since this generally improves performance a great deal. This algorithm is, in contrast to its name, quite aggressive. This is especially true if there are several sources slow starting simultaneously.

2.2 Congestion avoidance

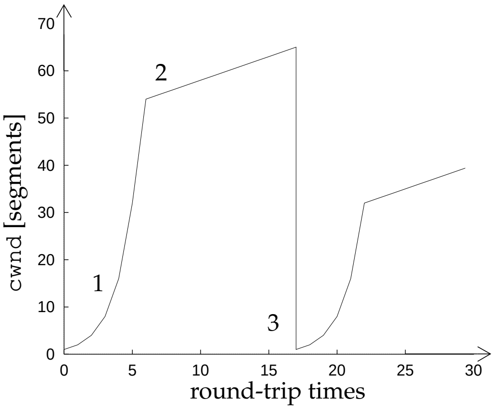
A variable called `ssthresh` (slow start threshold) is maintained within TCP. When the congestion window (`cwnd`) has grown to be equal to `ssthresh`, TCP enters the congestion avoidance state. The purpose of this state is to make the increase of `cwnd` less aggressive and in this way avoid to overload the network completely. For each segment acknowledged, `cwnd` is increased by $\frac{1}{cwnd}$ of a segment. This means that the increase of `cwnd` will be one segment each round-trip time and is linear, shown in step 2 of figure 2. Eventually, the network will become congested and one or more segments are lost.

2.3 Retransmit timeout

For one of the segments sent each round-trip, a timer called the retransmit timer will be set. If an acknowledgement for a segment with a sequence number less or equal to the timed segment is not received before this timer expires, that segment will be retransmitted.

TCP has to be able to distinguish lost segments, who's acknowledgements will not ever arrive, from delivered segments where the acknowledgements are just delayed, in order to avoid unnecessary retransmissions. For this purpose, the round-trip time (rtt) is measured continuously. This is done by counting the number of ticks of a 500 ms timer until the acknowledgement is received. The retransmit timeout (RTO), which is the value the retransmit timer is set to, is calculated using both a smoothed rtt and a smoothed mean deviation of the rtt.

When the retransmit timer expires, indicating a segment loss, `ssthresh` is set to half of `cwnd`, `cwnd` is set to one segment, the next segment to send is set to the one to be retransmitted and finally slow start is initiated. If the retransmitted segment is timed out too, the next RTO is doubled to make sure that the acknowledgement has time enough to get back if delay is increased due to the congestion. This action is called exponential backoff.



1: Slow start state. Growth of `cwnd` is exponential.
2: Congestion avoidance state is entered when `cwnd` = `ssthresh`. Growth of `cwnd` is linear.
3: A segment has been lost and there is a retransmit timeout. `ssthresh` is set to `cwnd`/2=32, `cwnd` is set to one and slow start is initiated retransmitting the lost segment. Congestion avoidance is entered when `cwnd`=32.

Figure 2: Example of how the congestion window changes in various states.

2.3.1 Timer granularity

When the rtt in an average LAN is in the order of milliseconds, one might wonder why TCP counts the number of ticks of a 500 ms timer to measure the rtt. The accuracy in this measurement is obviously not very good, but the calculated RTO will surely be long enough to avoid unnecessary retransmissions. On the other hand, if an acknowledgement was expected several rtt's ago, why wait for more than half a second before the segment is retransmitted? Some newer implementations of TCP use a 100 ms timer to reduce the time spent waiting.

2.4 Fast retransmit

When TCP receives an out-of-order segment at the destination, a duplicate acknowledgement is sent. That is, an exact copy of the most recent acknowledgement is sent, indicating which segment that really was expected. To the source, this has one of two meanings: either the segment was lost and have to be retransmitted, or there was just a reordering of two segments along the way through the network. If several consecutive duplicate acknowledgements are received, it becomes more likely that what caused this is indeed a lost segment.

What the fast retransmit algorithm does is to count the number of consecutive duplicate acknowledgements received. If the counter exceeds a threshold, that is usually set to three, the presumed lost segment is retransmitted without waiting for the retransmit timer to expire. Just as for a retransmit timeout, `ssthresh` is set to half of `cwnd`, `cwnd` is set to one and slow start is initiated. The performance of TCP is dramatically improved with fast retransmit, since most retransmissions can be handled without waiting for a timeout.

2.5 Fast recovery

TCP connections with large windows that suffer from segment loss spend a long time recovering from the loss when slow starting with `cwnd` set to one. If the network is not badly congested, it might be enough to reduce `cwnd` somewhat and try to maintain a fairly high throughput. The fast recovery algorithm, which is only implemented in combination with fast retransmit, try to accomplish this by changing the events following the reception of three duplicate acknowledgements and the retransmission.

Each time another duplicate acknowledgement is received, `cwnd` is increased by one allowing one new segment to be sent. This is since receiving acknowledgements is a sign that the network is not completely congested. If a segment has left the network at the other end there must be room for another one. When the acknowledgement for the retransmitted segment is received, `sstresh` is set to half of `cwnd` and `cwnd` is set to `sstresh` instead of one. This means that the slow start procedure is skipped and congestion avoidance is initiated instead.

Performance of connections that suffer from a single segment loss is improved compared with implementations with the fast retransmit algorithm only. If there is heavy congestion in the network and several segments are lost, the fast recovery algorithm may have difficulties handling the situation, resulting in one or more timeouts [7].

3 ATM

The Asynchronous Transfer Mode (ATM) supports both real-time and non-real-time services with varying quality of service requirements. Upon setup of a connection, there is a request for network resources made to the network. Both desired and acceptable quality of service parameters as well as the type of service intended for use are included in the request. The network may either accept or reject the request depending on the resources available at the moment. If the connection is accepted, both parties have to conform to the traffic contract agreed upon.

3.1 ATM service categories

The ATM service category chosen for a connection depends on the quality of service requirements. Five categories are available.

CBR	Constant Bit Rate
rt-VBR	Real-Time Variable Bit Rate
nrt-VBR	Non-Real-Time Variable Bit Rate
UBR	Unspecified Bit Rate
ABR	Available Bit Rate

The CBR service allocates a static amount of bandwidth available at all times throughout the duration of the connection. Maximum cell delay and delay variation are specified in the traffic contract. It is intended for real-time applications such as video and voice.

The VBR services are intended for traffic where the bandwidth required varies in time. Peak cell rate and maximum burst size are specified. Those sources conforming to the contract will experience low cell loss. For the rt-VBR service, the contract include constraints on cell delay and delay variation.

To be able to fully utilize a link where all resources have not been allocated or are not presently used, the UBR or ABR services could be used. These two services have lower priority than CBR and VBR, and often have a separate queue in the switches. Therefore, the resources available for UBR and ABR will vary in time if used in combination with higher priority services.

For the UBR service, no commitments are made with respect to cell loss or delay. Peak cell rate could be enforced or just informational. This is a best-effort service where flow control is left to higher level protocols.

The ABR service uses flow control to let the sources adjust their cell rates. Those sources that follow the directives of the flow control mechanism will have low cell loss and fair allocations of the bandwidth available. A minimum cell rate is specified, which may be zero.

3.2 ABR flow control

Connections using the ABR service send Resource Management cells (RM-cells) in addition to their ordinary data cells. Usually every 32:nd cell is a RM-cell. When in transit to the destination, they are called forward RM-cells, and when they have reached the destination they are returned to the source, now called backward RM-cells. This establishes a closed control loop. Some of the flags and fields contained in the RM-cells are:

DIR	Direction. If set, this flag indicates that it is a backward RM-cell.
CI	Congestion Indication. If this flag is set, the source must reduce its cell rate.
NI	No Increase. If this flag is set, the source may not increase its cell rate.
ER	Explicit Rate. This is the rate the network limits the source to.
CCR	Current Cell Rate. This is the current cell rate of the source.
MCR	Minimum Cell Rate. This is the minimum cell rate the source is entitled to according to the traffic contract.

A set of rules for the behaviour of switches, destinations and sources apply for ABR connections. They are defined in [1] and the most important ones can be summarized as follows:

3.2.1 Switch behaviour

A switch in the network must use at least one of the following methods to handle congestion:

a) EFCI marking. The EFCI (Explicit Forward Congestion Indication) bit in the header of ordinary data cells may be set.
b) Relative rate marking. The CI or NI bit may be set in RM-cells.
c) Explicit rate marking. The ER field may be reduced in RM-cells.
d) VS/VD control. The ABR control loop may be divided into two or more Virtual Sources and Virtual Destinations.

3.2.2 Destination behaviour

The destination saves the EFCI bit of incoming data cells as the EFCI state of the connection. Before returning an incoming RM-cell to the source, the DIR bit is set and, if the EFCI state is set, the CI bit is also set. The destination may also reduce the ER field and/or set the CI or NI bits if suffering from internal congestion.

3.2.3 Source behaviour

When the source receives a backward RM-cell, the CI and NI bits are checked first. If the CI bit is set, the rate must be reduced by a factor called RDF (Rate Decrease Factor), but if none of the CI or NI bits are set, the rate may be increased by a factor called RIF (Rate Increase Factor) instead. Just after this, the source verifies that the rate is within the interval [MCR,PCR] from the traffic contract, but not greater than the ER field in the RM-cell.

3.2.4 ERICA+

The ERICA+ switch algorithm [2] is an Explicit Rate marking algorithm designed by the Ohio State University. It is a modified version of the ERICA algorithm first presented in February 1995. The algorithm is applied to each output port of the switch. The goal of ERICA+ is to give fair and efficient bandwidth allocations to all ABR connections using the port. The allocations are made using the measurement of two metrics, where the input rate of each connection is the primary and the queue length of the port the secondary.

The measurements are continuously made during averaging intervals. The length of each interval is the shortest of a specified period of time and a specified number of cells passing through the port. When few cells pass through the port, the time limit ends the averaging interval giving new measurements regularly. When there are many cells passing through, the cell limit ends the interval giving accurate measurements frequently.

During each averaging interval, the number of cells received are counted on a per-VC basis for those connections using the ABR service. This also gives information about the number of active sources during the interval.

At the end of each averaging interval, the load factor and each VC's fair share of the available capacity are calculated as follows:

$$\text{Load factor } z = \frac{\text{ABR Input Rate}}{\text{Target ABR Capacity}}$$

$$\text{Fair Share} = \frac{\text{Target ABR Capacity}}{\text{Number of active sources}}$$

where the `ABR Input Rate` is the number of cells received by the port from all VCs during the interval divided by the length of the interval. The `Target ABR Capacity` is defined as:

$$\text{Target ABR Capacity} = f(q) * \text{Total ABR Capacity}$$

The `Total ABR Capacity` is the link capacity subtracted by the capacity used by higher priority services, and `f(q)` is a function of the queueing length q. The objective of ERICA+ is to maintain a queue, with a fixed queueing delay `T0`, that makes it possible to utilize the link better, since having a non-zero queue implies full utilization. This results in a queue with varying target queue length `Q0` as there are changes in the available bandwidth, and it is defined as:

```
Q0 = Total ABR capacity * T0
```

The fixed queueing delay `T0` must be set long enough to avoid low utilization at transient underloads, but short enough not to cause unnecessary delay.

The function `f(q)` should regulate the `Target ABR Capacity`, and thus the queue length, so that the queue delay will be `T0`. The proposed function is:

$$f(q) = \frac{b * Q0}{(b-1) * q + Q0} \qquad 0 \leq q \leq Q0$$

$$f(q) = \max\left(QDLF, \frac{a * Q0}{(a-1) * q + Q0} \right) \qquad q > Q0$$

where q is the current queue length. `QDLF`, `a` and `b` are constants set to limit the function's reserved capacity for draining the queue and the aggressiveness of the capacity calculations. An example of how the function might look is seen in figure 3. Since the target queue length `Q0` changes with the available capacity, the function `f(q)` is stretched along the x-axis back and forth at each new averaging interval.

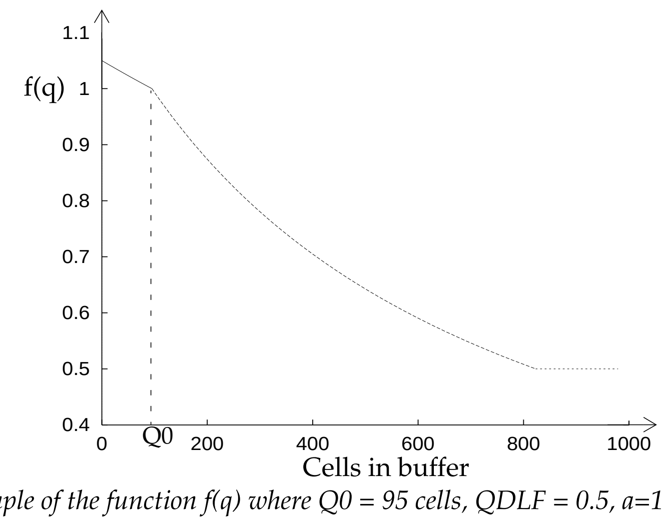


Figure 3: Example of the function f(q) where Q0 = 95 cells, QDLF = 0.5, a=1.15 and b=1.05.

When a backward RM-cell is received at the switch, the proper rate allocation for this VC is to be calculated and written into the ER field of the cell. If a connection can not use its fair share, the remaining capacity is fairly allocated to those sources that can. To be able to do this, the following is calculated for the VC:

$$\texttt{VCShare} = \frac{\texttt{Cell rate of VC}}{\texttt{Load factor z}}$$

If all connections were given an allocation of `VCShare` and they all used this capacity, the `Load factor` would become unity. To achieve max-min fairness, the value of the calculated ER during overload becomes:

`ER Calculated = max (FairShare, VCShare)`

and during underload

`ER Calculated = max (FairShare, VCShare, Max Alloc Prev)`

where the threshold for overload is were the `Load factor z` is greater than 1+δ, where δ is a small fraction. `Max Alloc Prev` is the greatest allocation made to any of the connections during the previous interval. In this way, the connections are given as fair and big allocations as possible during overload, and allocations as big as any other during underload. Finally, the algorithm verifies that the calculated ER is not bigger than the `Target ABR Capacity`;

`ER Calculated = min (ER Calculated, Target ABR Capacity)`

Since there might be a congestion elsewhere in the network, where the allocations for this VC is even smaller, the calculated ER is only written to the ER field of the RM-cell if it is smaller than the present value.

3.3 Packet discard policies

It is crucial for an ATM switch, experiencing cell loss due to buffer overflow, to use a packet discard policy for TCP/IP traffic. This is true since loss of just one cell from a packet makes the rest of the cells from that packet useless, but they are still transmitted through the network occupying precious buffer space. Furthermore, a connection using a packet size of 200 cells, having a cell loss probability of 0.005, will loose as good as every packet sent. If the useless cells were discarded, the buffers could be used by those connections that really need them.

ABR connections usually do not suffer from buffer overflow and are not affected by this as much as UBR connections are. Without a packet discard policy, TCP connections using the UBR service are extremely unfair and may experience very low throughput. The synchronization of the TCP sources is another problem. When the buffer overflows, one or more cells from almost every TCP connection will be lost since they are interleaved. All sources are interrupted and have to start retransmitting the lost packets. This leads to low utilization of the link.

A better scenario would be to let some sources have their cells accepted by the buffer while other source's cells are lost. This means that there will be traffic using the link at all times.

3.3.1 Early packet discard

The idea with early packet discard [5] is to discard all cells from new arriving packets when the number of cells in the buffer exceeds a threshold. All connections, where the first cell of a packet has already been received before the threshold is exceeded, have the rest of the cells from these packets accepted as long as there is buffer space enough. A reasonable setting for the threshold is 90% of the buffer size. The maximum buffer space required above the threshold is the packet size multiplied by the number of active sources.

3.3.2 Selective drop

The selective drop scheme [9], which is a simpler version of the Fair Buffer Allocation scheme proposed in [6], counts the number of cells from each VC occupying the buffer. The VCs having cells in the buffer are seen as active. The `Fair Allocation` and the `Load Ratio` of a VC are defined by:

$$\texttt{Fair Allocation} = \frac{\texttt{Total buffer occupancy}}{\texttt{Number of active sources}}$$

$$\texttt{Load Ratio of VC} = \frac{\texttt{Cells in buffer from VC}}{\texttt{Fair Allocation}}$$

As long as the total number of cells in the buffer is less than a threshold, all cells are accepted. But when the threshold is exceeded, the next new arriving packet from connections having a `Load Ratio` greater than the parameter `Z` is dropped. `Z` is chosen to a value of 0.8 to 1.0 for best performance.

The selective drop scheme ensures that the buffer is shared among the connections in a fair manner. This improves fairness of TCP connections using the UBR service a great deal. Selective drop is also successful in breaking synchronization of TCP sources, giving higher throughput and utilization.

4 The Model

The model is written in C++ for the event driven PLASMA simulation environment developed by Ericsson. A number of objects have been designed to represent the functionality of a real network consisting of hosts, links and switches. Each host is made up of a TCP, AAL5 and ATM layer. The switches have a switch core with any number of output buffered ports connected.

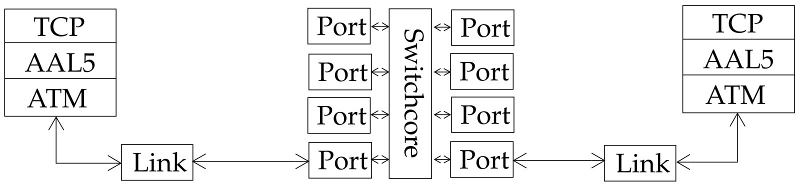


Figure 4: The elements used in the model.

The links delay cells passing through to simulate propagation delay. Traffic is bi-directional. That is, acknowledgements for TCP segments are sent as any other IP packet. Each host may act as a TCP source, destination or both. With this model, it is possible to simulate an environment with any number of TCP hosts and switches connected in an arbitrary manner.

4.1 TCP

Two different implementations of TCP are modelled. These two implementations, Tahoe and Reno, are common reference implementations of TCP from the Computer Systems Research Group at Berkeley. Tahoe was released in 1988 and include the Fast Retransmit algorithm. In 1990 Reno was released with the Fast Recovery algorithm.

The TCP sources are greedy in the sense that they always have data to send. Parameters that could be set for TCP are the size of the send window, the segment size, granularity of the retransmit timer and TCP implementation. It is also possible to use the proposed bugfix for successive fast retransmits [10], a known problem for TCP connections suffering from multiple non-consecutive segment loss within the same window.

4.2 ATM

The hosts are able to use either of the UBR or ABR services. For UBR connections, the PCR (Peak Cell Rate) is set at initialization and is used throughout the simulation. For ABR connections, the cell rate is first set to ICR (Initial Cell Rate) and is later controlled by the ER marking switches.

The switches use the ERICA+ algorithm for ABR connections. The averaging interval and target queueing delay for ERICA+ as well as the buffer size, discard policy and discard threshold are set at initialization.

The switch ports have a capacity of 155 Mbit/s. Traffic using the VBR service is modelled by reducing the cell rate of the output port by an amount representing its actual load given by seven sources sending data at 20 Mbit/s each. The bandwidth available to ABR and UBR connections varies between 15 Mbit/s and 155 Mbit/s.

The bandwidth used by VBR sources is well modelled by a Markov chain, as the one in figure 5, where the states represents the number of active VBR sources. The intensities by which each source starts and stops sending are denoted by β and μ. To make the simulations faster, a simplified version of this model is used. After specified intervals of time, the number of active sources is set at random, where the distribution of the number of active sources is binomial with a mean of 3.5 active sources, just as it is for the Markov chain in figure 5, if β is equal to μ. This model will give more drastic changes in available bandwidth than the reference model, and will therefore make it easier to analyse the behaviour of the ABR flow control mechanism. In figure 6 an example of the bandwidth used by the VBR traffic during one second of simulation is shown.

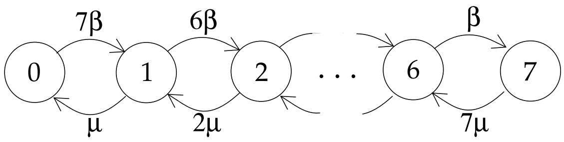


Figure 5: The Markov chain reference model with seven sources.

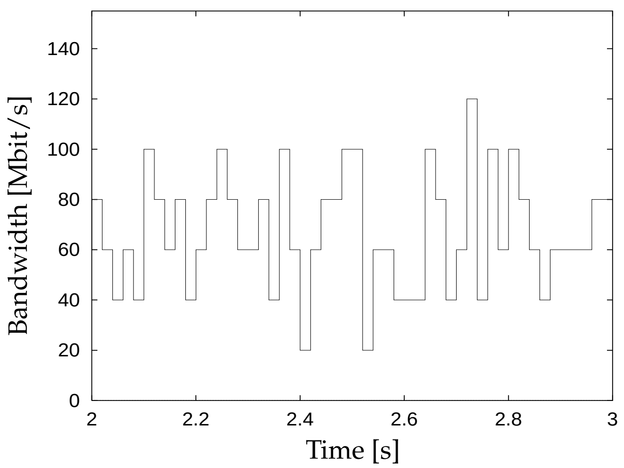


Figure 6: Example of the bandwidth used by VBR traffic at a switch port during one second of simulation, where the time interval between changes is 20 ms.

4.3 Network configurations

In the simulations presented in later chapters, some general parameter settings are used, unless others are specified. For TCP, these are:

- The TCP implementation used is Tahoe. Since Reno has problems with multiple packet loss, Tahoe is better for analysing other properties of the performance of TCP over ATM. The problems with Reno are presented in the next chapter.
- The TCP send window is set to 64 kBytes for LANs and 8*64 kBytes for WANs.
- The segment size is set to 512 Bytes, a standard value for many TCP implementations. This size is often used for TCP connections over WANs, since there might be different types of link layers, limiting the maximum segment size, along the path.
- The TCP retransmit timer granularity used is 0.1 s.
- The bugfix for successive fast retransmits proposed in [10] is used.
- The delayed acknowledgement timer is not set. Acknowledgements are sent as soon as the segments arrive at the destination.
- All TCP sources start sending simultaneously at start-up of the simulation. This makes it easier to test the ability to break synchronization of TCP sources at buffer overflow.

General settings for the switches and links are:

- The packet discard threshold is set to 90% of the buffer size.
- The cell and time limits for the averaging interval for ERICA+ is set to 100 cells and 1 ms. The parameters `QDLF`, `a` and `b` are set to 0.5, 1.15 and 1.05. These are recommended settings taken from [2]. The target queueing delay `T0` is set to 0.5 ms for LANs and 2 ms for WANs. For LANs this will result in a target queue length `Q0` of between 24 and 182 cells depending on the available capacity. For WANs `Q0` will be in interval of 95 to 730 cells.
- The access links between the sources and the switches have a propagation delay of 5 µs. The propagation delay of the bottleneck links, that is, the links between the switches, is set to 5 µs for LANs and 5 ms for WANs.
- Link capacity and the PCR of all ports are set to 155 Mbit/s.
- The time between changes in the bandwidth used by VBR is set to 20 ms.

For simulations using the UBR service, the selective drop scheme, with the parameter `Z`=0.9, is used. A comparison between the use of the early packet discard and selective drop schemes for the UBR service is presented in the next chapter. Since buffer overflow is not very common for ABR connections, early packet discard is used instead. The complexity of the selective drop scheme compared to early packet discard does not motivate the use for it under these conditions.

For LAN simulations, the N-source configuration, showed in figure 7, is used. The time simulated for LANs is 5 seconds. This is enough to let 15 sources send approximately three Mbytes of data each.

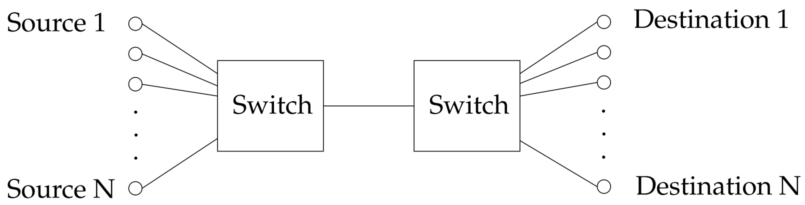


Figure 7: N-source configuration used for simulations of a LAN.

The N+M configuration in figure 8 is used for WAN simulations to study the effects of sources having different feedback delays and passing through more than one bottleneck link. The time simulated for WANs is 10 seconds.

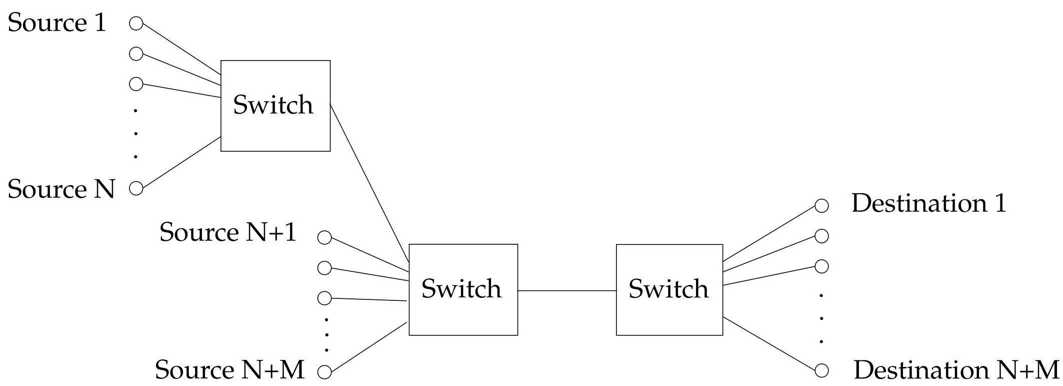


Figure 8: N+M source configuration used for simulations of a WAN.

5 Simulations

The results from the simulations have been divided into one section presenting properties of TCP and one section presenting the performance of TCP over both ABR and UBR. A summary of the parameter settings that are fixed and variable in these simulations is found in Appendix A.

5.1 Properties of TCP

The simulations done to analyse the properties of TCP include studies of the TCP implementations Tahoe and Reno, the granularity of the retransmit timer and problems with successive fast retransmits.

5.1.1 The Tahoe and Reno implementations

Since the Reno implementation include the fast recovery algorithm, Reno performs better than Tahoe when there is just a single segment lost within the same window. But, when there is a retransmission, Reno is able to retransmit just one segment each round-trip time. Therefore, there is a problem with Reno when there are several segments lost within one window. In most of these cases, Reno has to wait for a timeout to recover from the loss [7].

In figure 9, traces from the sources in a 15-source LAN configuration, using Reno TCP over UBR, are shown. No background VBR traffic is used in the simulation, meaning that the bandwidth available for the TCP sources to share is 155 Mbit/s. The buffer size is 3000 cells.

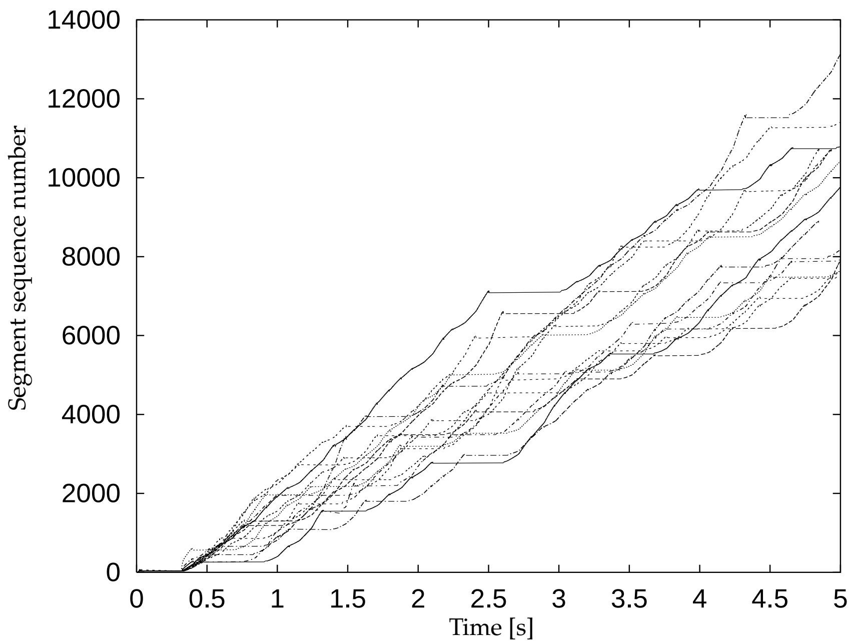


Figure 9: Sequence number of delivered segments for 15 sources using Reno TCP.

At a timeout, the source is stalled for a period of time, indicated by a horizontal line in the figure. All of the sources suffer from multiple segment losses at several occasions that they can not recover from without waiting for a timeout.

In figure 10, a simulation for Tahoe TCP over UBR, using the same configuration, is shown. There are no timeouts but for two connections once at the very start. Both utilization and fairness is better. Recall that Reno was released in 1990 as an enhanced version of Tahoe, released in 1988.

There is a problem with Tahoe too though. When there is a retransmission, Tahoe forgets about the segments sent earlier and starts transmitting as many segments as the window allows, starting with the segments reported missing in the acknowledgements. As a result of this, Tahoe could retransmit segments that already have been successfully received at the destination. Both of these problems will be solved with the proposed implementation of SACK (Selective acknowledgement) TCP [7][8].

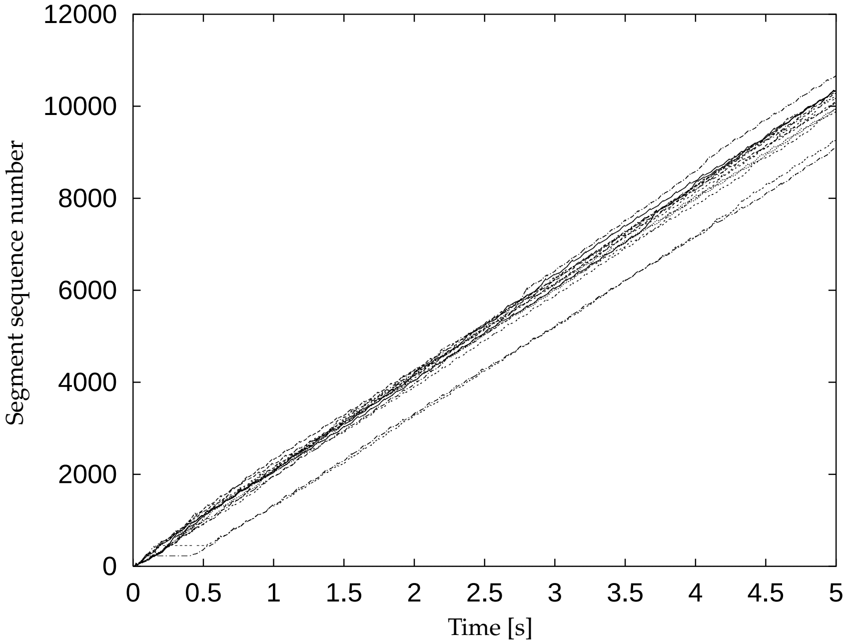


Figure 10: Sequence number of delivered segments for 15 sources using Tahoe TCP.

5.1.2 Timer granularity

As seen in figure 9, the time spent waiting at a timeout usually is 300 ms or 500 ms. The granularity of the retransmit timer in this simulation is 100 ms and the round-trip time is about 5 ms. If the granularity is set to 1 ms, the traces for Reno is similar to the ones for Tahoe with a 100 ms timer, show in figure 10, with slightly better throughput, even though there still are as many timeouts. The time spent waiting is reduced to 20 ms each timeout. Note that if there are no timeouts in the first place, a finer granularity will not improve performance at all, it will just remove the safety margin for avoiding unnecessary retransmissions.

When the granularity is as fine as 1 ms, TCP becomes very sensitive to delay variations. In a LAN environment, the delay may not vary much, but in a WAN it does. Since there is only one timed segment each round-trip, the variations in delay for timed segments are bigger for connections with long round-trip time as the level of congestion is more likely to change during a long period of time.

5.1.3 Successive fast retransmits

What first appeared to be a bug in the model of TCP, turned out to be a previously documented problem with the protocol. TCP connections that suffer from multiple non-consecutive packet loss within one window, will experience very poor performance [10]. This is due to TCP not being able to handle a situation where the fast retransmit algorithm is invoked several times for the same window of data. For Tahoe, this may occur if two non-consecutive runs of packets are lost within the same window. It is mainly connections with large windows that suffer from these packet losses.

The events following the successive fast retransmits are somewhat complex, but the result is that the source will decrease its slow start threshold (`sstresh`) repeatedly until it becomes two segments. At this point, the source will do congestion avoidance from the very start, having linear growth of the window instead of exponential. The result is an extremely slow recovery from the packet loss.

In figure 11, the traces of ten sources using Tahoe TCP over UBR are shown. At the time 0.6 seconds, there is a buffer overflow resulting in packet losses. Three sources are suffering from multiple non-consecutive packet losses and experience successive fast retransmits. At the time 4 seconds, these three sources have sent about 400 packets each during the time after the buffer overflow. During the same time interval, most of the other sources have sent 5300 packets each. Note that there are no timeouts at all.

There is a proposed fix [10] for this problem with TCP. The fix limits the amount of fast retransmits to one for each window of data. In figure 12, traces of the same ten sources as before are shown, using the fix for successive fast retransmits. They are performing much better than before. The disadvantage with using the fix is that there are situations where the fast retransmit algorithm is disabled and there is a need for it. A better way to solve this problem would be to use SACK TCP [8]. With SACK TCP, segments already received correctly would not be retransmitted. Therefore, duplicate acknowledgements causing fast retransmit to be invoked, would not be sent in the first place.

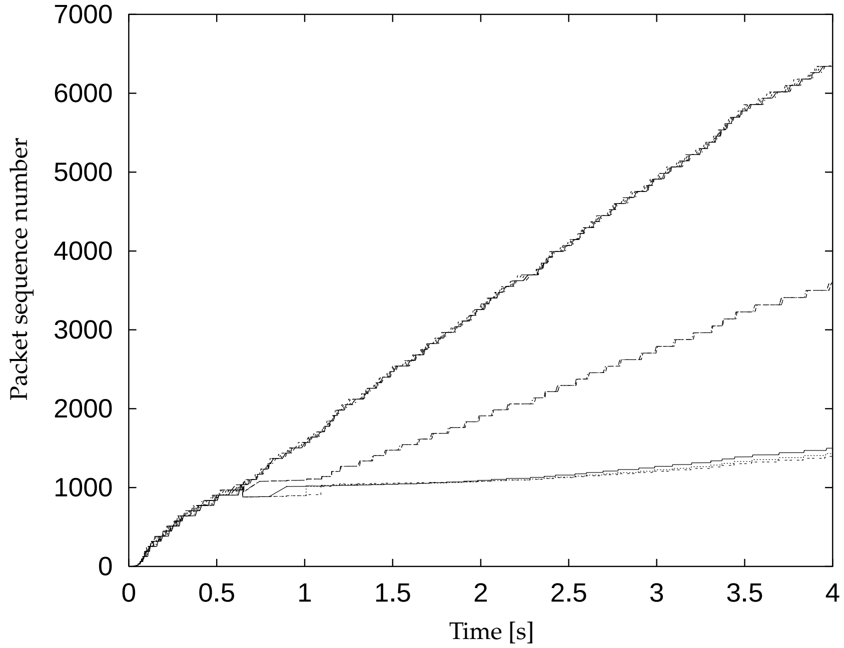


Figure 11: Example of sources suffering from successive fast retransmits.

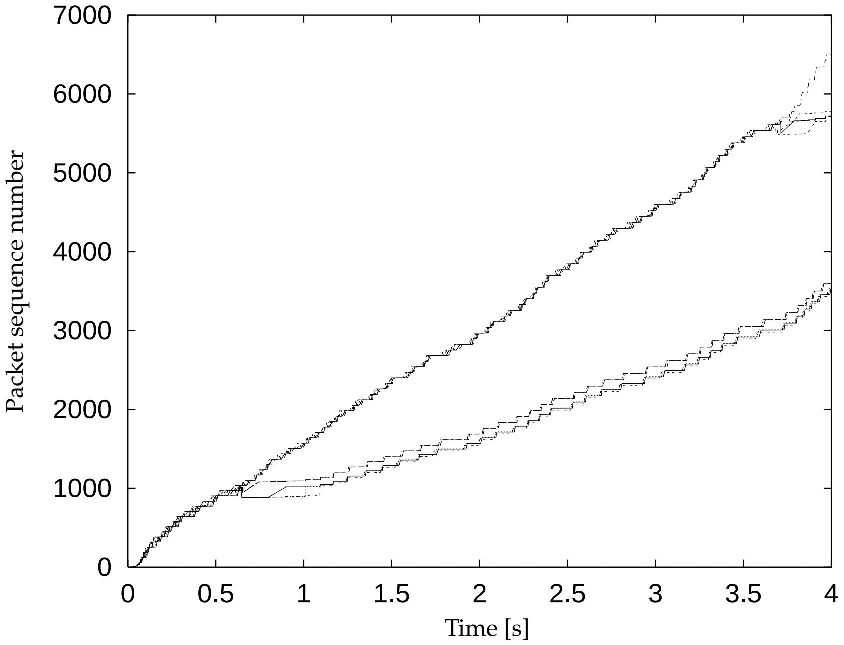


Figure 12: The same sources using TCP with bugfix for successive fast retransmits.

5.2 ATM

The results presented below have been attained from simulations of TCP traffic using the UBR and ABR services. The buffer size required to get high performance, dependent on the number of sources, the environment and TCP settings used, have been studied. The importance of the use of a discard policy is demonstrated, along with a comparison of the early packet discard and selective drop schemes.

The results include measurements of the fairness index [9] and link utilization. These measurements are defined as:

$$\text{Fairness index} = \frac{(\Sigma x_i)^2}{n * \Sigma x_i^2}$$

where x_i is the number of delivered segments of the i:th source and n is the number of sources. If all sources deliver the same amount of segments, the fairness index will become unity. A fairness index below 0.95 is not very fair, and an index below 0.9 should be avoided.

$$\text{Utilization} = \frac{\text{Sum of delivered segments for all sources}}{\text{Maximum number of delivered segments possible}}$$

Segments received more than once by the destination are counted as one segment only. The maximum number of delivered segments possible is the amount attained if ATM cells, holding user data, are sent at the maximum possible cell rate of the link at all times, where a 512 byte segment with its 40 byte TCP/IP header and 8 byte AAL5 trailer use 12 ATM cells. A 9140 byte segment use 192 ATM cells.

5.2.1 TCP over UBR

The fairness index and utilization for the N-source LAN configuration are shown in figure 13 and figure 14 as a function of the buffer size of the bottleneck switch port. The figures show the results obtained from simulations with 5, 15, 30 and 60 Tahoe sources.

In all of these cases there is a very clear threshold where the fairness among the sources collapses if the buffer size is not big enough. If the buffer size is slightly bigger than the threshold, both the fairness index and utilization are high. If buffer size is further increased, the fairness drops. This is because some sources do not suffer any packet loss when the buffer overflows, and the buffer does not overflow as often as for a configuration with a smaller buffer. If there are many buffer overflows, however, the number of times each source will suffer from packet loss will tend to even out.

To get highest possible performance, the main issue is not to minimize the number of packet losses. TCP has to suffer from packet loss to adjust its rate properly. What is important is to make sure that all sources adjust their rates, otherwise the bandwidth will not be shared in a fair manner.

When the buffer size is as large as the sum of the TCP send windows of all sources, there is no packet loss at all. At this point, the utilization is as high as it can be and the bandwidth is shared completely fair. If 64 kbytes TCP send windows and 512 byte segment sizes are used, this should occur when the buffer size is 7680 cells with 5 sources, 23040 cells with 15 sources. Figure 13 and figure 14 show that this is correct. The fairness is equal to one and the utilization has reached its maximum. The reason why the utilization is not equal to one is that the buffers and links are empty at start-up, and take some time to fill up.

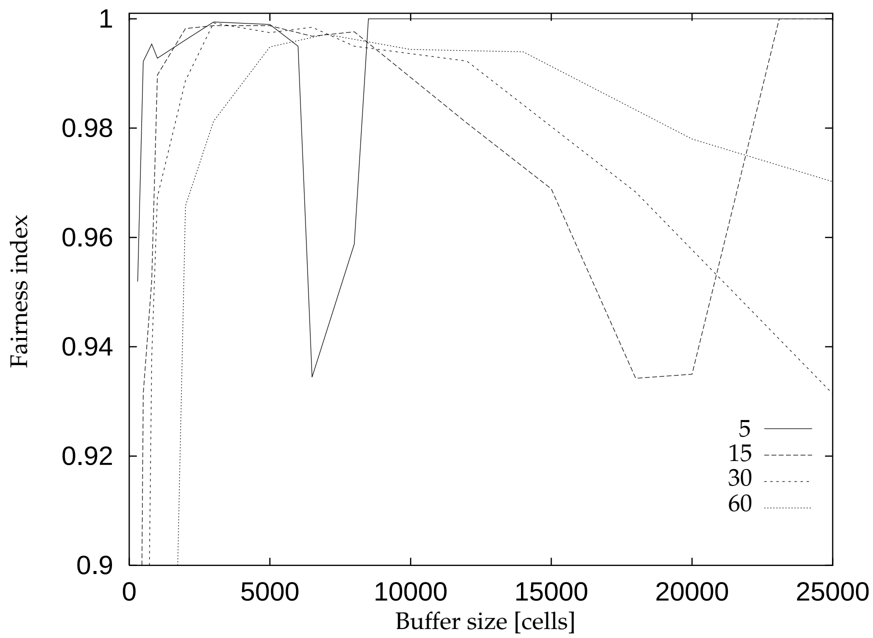


Figure 13: Fairness index for LAN configuration with 5, 15, 30 and 60 Tahoe sources using UBR.

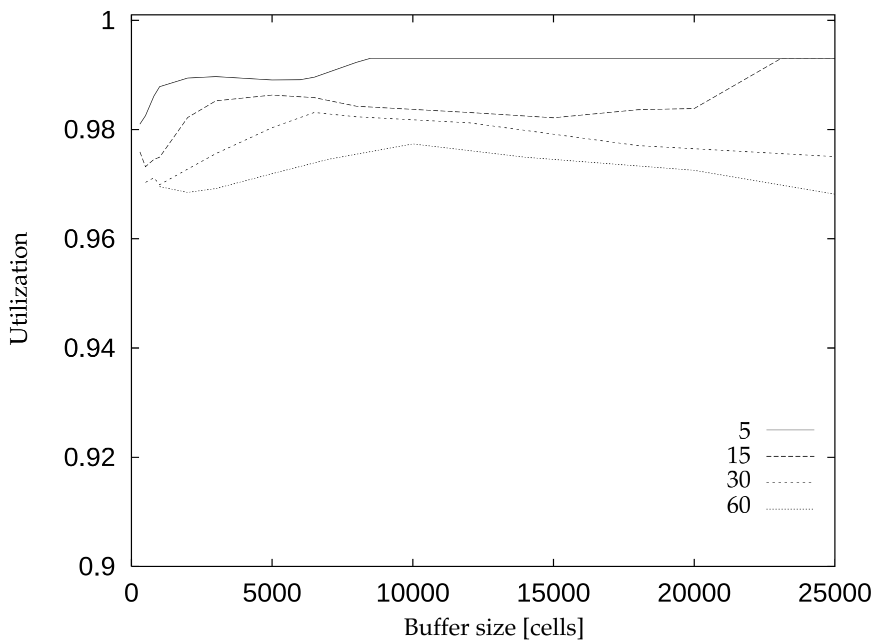


Figure 14: Utilization for LAN configuration with 5, 15, 30 and 60 Tahoe sources using UBR.

To get the best performance, the buffer size should obviously be somewhat bigger than the threshold. Having even larger buffers might reduce performance. To have buffers equal to the sum of the TCP windows, requires the knowledge of the number of sources and will imply very large buffers introducing a long delay.

The use of a larger TCP segment size will increase the required buffer size. The simulations show that if a segment size of 9140 bytes is used, the threshold is moved to a buffer size approximately 16 times larger than for the 512 byte segment size. For each 9140-byte segment, 192 cells are sent. 12 cells are sent for the 512-byte segment. The ratio between the number of cells sent for each segment for these two sizes is, not surprisingly, 16. This is also true for the 1460 byte segment size, where the ratio of the 32 cells for 1460-byte segments and the 12 cells for 512-byte segments is 2.7.

It seems that the buffer has to fit at least two hole segments per source that is using the port in order for them to have a high throughput. This is probably since new segments are sent two at a time during slow start. If the buffer is not able to fit both of these, the source will not be able to increase its window at all.

For the N+M source WAN configuration, the round-trip time is longer than for the LAN configuration. The buffers need to be bigger in order to let the sources to have large windows. If they do not have large windows, they will not be able to fill the pipe. The result is that there will not be a continuous flow of cells through the switch port, and therefore, utilization will be low.

If sources have different round-trip times, another problem is introduced. There will be sources near the destination, having short round-trip time, and sources far from the destination, having long round-trip time. When buffers are small they overflow quite often, and, as a result of this, most sources are forced to do slow start. In the slow start phase, the near sources, are able to increase their windows faster. When the buffer overflow, the near sources will have a larger `sstresh` (slow start threshold) than the far ones for the next run. That is, the near sources will slow start their windows to a larger size than the far sources.

To summarize the discussion above: if the buffers are too small, the near sources will benefit from this, causing unfairness among the near and the far sources. In figure 15 the fairness index and utilization for the N+M WAN configuration are shown, using 10, 20 and 30 sources. The fairness among the near sources alone in this simulation is very good, just as for the fairness among the far sources alone. The round-trip time is 10 ms for the near sources, and 20 ms for the far sources.

The fairness index in figure 15 shows how large the buffers must be in order to avoid this problem for this configuration. Where the fairness index in figure 15 flattens out, the bandwidth allocations for the near and the far sources are as fair as they ever will be. This property is also true for TCP connections using ABR.

In figure 16, the near sources have been removed. The descriptions of the problems with longer round-trip time and the far/near sources seem to be correct. The utilization is lower and the fairness index is high. Note that the number of sources used in these simulations does not affect the results very much.

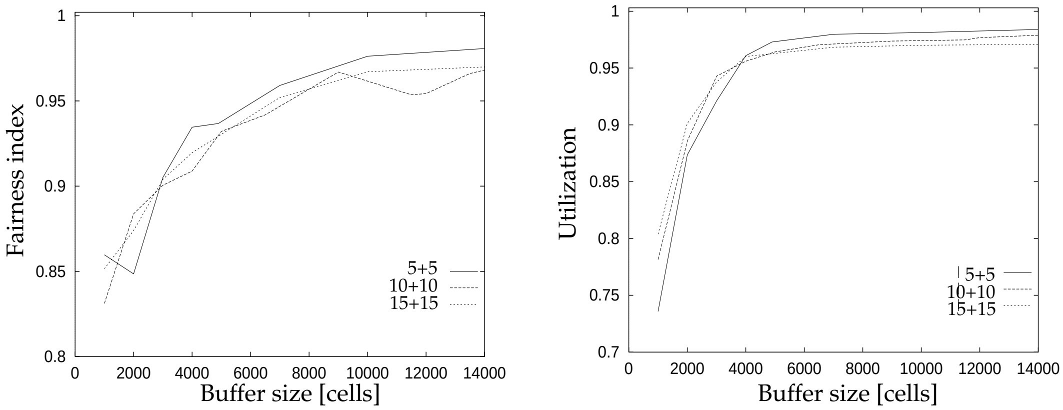


Figure 15: Fairness index and utilization for N+M configuration with 10, 20 and 30 sources

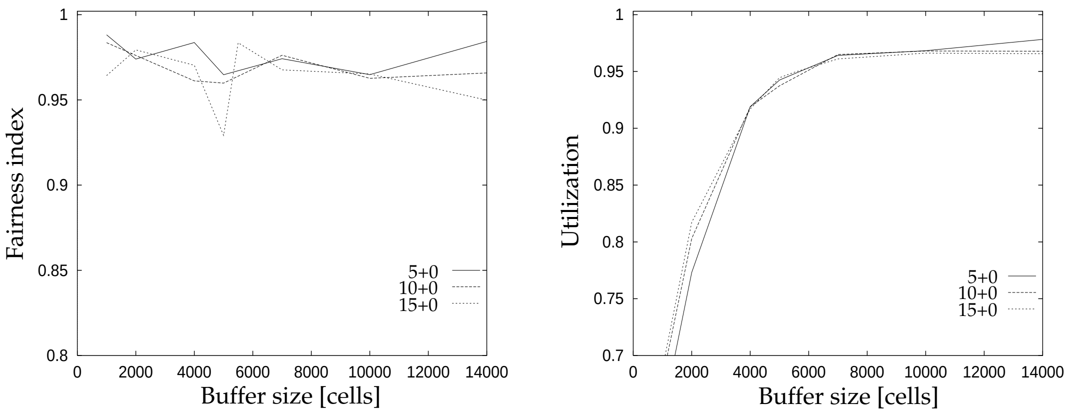


Figure 16: Fairness index and utilization for N+M UBR configuration without the near sources

5.2.2 TCP over ABR

When using the ABR service, a key issue is the feedback delay experienced by the source. The feedback delay is the time for a cell to be sent from a congested switch to the source and back to the switch. This is how long it takes for the flow of cells to change when there is a congestion or when a switch is no longer congested.

If the average time between drastic changes in available bandwidth at the switches is much longer than the feedback delay, ABR will function very well. If, on the other hand, the time between these changes is shorter than the feedback delay, the source will not ever deliver cells at the rate it is supposed to in order to avoid congestion or to make use of new available bandwidth. This situation is more likely to occur in a WAN environment, since delays are generally much longer than in a LAN.

First, results from simulations where the time between changes in available bandwidth is longer than the feedback delay are presented. In figure 17 and figure 18 the fairness and utilization for the N-source LAN configuration using ABR is shown.

As for TCP over UBR, there is a clear threshold for the fairness index. For the LAN configuration, the target queue length `Q0` will vary in the interval 24 to 182 cells. It is quite obvious that the threshold should be at a buffer size bigger than 200 cells, since ERICA+ tries to maintain a queue as big as `Q0`. Above the threshold, the fairness index is very close to unity.

There are two reasons for the utilization being lower for ABR than for UBR. First, the RM-cells represent 3% of the cells sent and are reducing the utilization by the same amount, and second, the small buffers can not hold enough cells to avoid getting empty at all times when there is a temporary underload.

The segment size does not affect the buffer requirements of the switches at all. When one or more segments are to be sent, they are buffered internally at the ATM interface of the source, waiting for transmission. The cells are then sent at the rate allocated by the network.

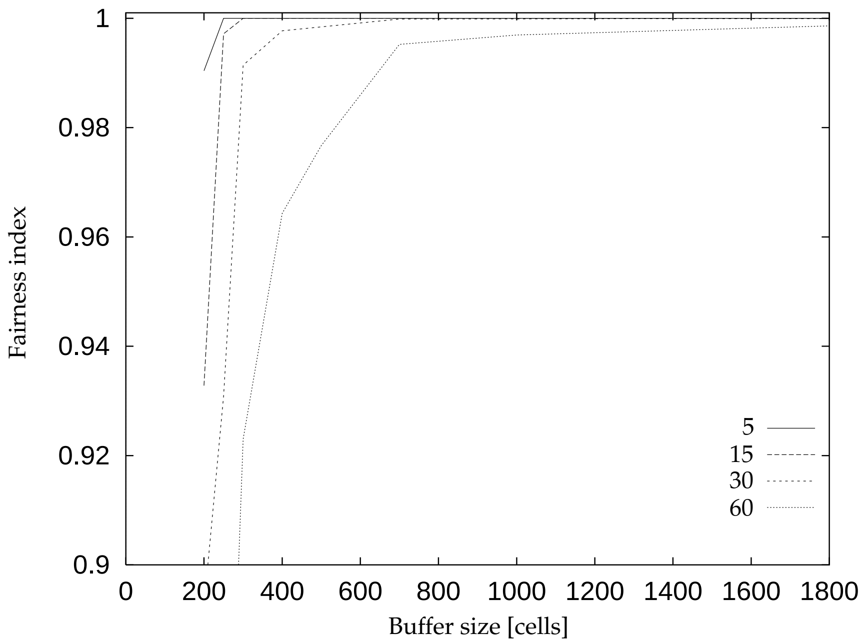


Figure 17: Fairness index for LAN configuration with 5, 15, 30 and 60 sources using ABR.

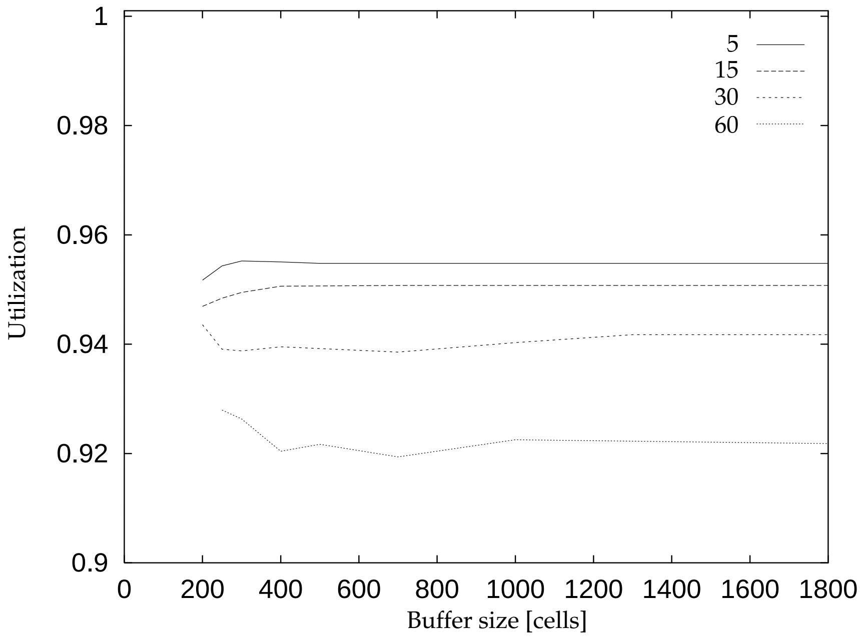


Figure 18: Utilization for LAN configuration with 5, 15, 30 and 60 sources using ABR.

If there is no loss, the TCP window will grow to its limit. If the window is large enough, this will enable the source to fill the pipe. That is, there will be no situation where TCP is idle waiting for acknowledgements. There will be a continuous flow of cells at the allocated rate. This means that TCP is not aware of the level of congestion in the network, and could temporarily deliver far more data to the ATM layer than it would otherwise. As mentioned before, these cells will be buffered internally at the ATM interface, waiting for transmission. This will cause segments to be sent in the eyes of TCP, but not yet transmitted to the network. The round-trip delay measurements will be affected and a false retransmission could occur if the granularity of the retransmit timer is not coarse enough.

For the N+M source WAN configuration, the sources near the bottleneck link will have a feedback delay of 10 µs, where as the far sources will have a feedback delay of 10 ms. ERICA+ handles this situation very well. The near sources respond very quickly to the feedback and are able to make use of new available bandwidth when the far sources have not yet been informed of the new situation. When they have, bandwidth is shared fairly.

The target queue length `Q0` will vary in the interval 95 to 730 cells for the WAN configuration. The performance threshold should be at a buffer size somewhat bigger than this. As seen in figure 19, the fairness index drops fast at a buffer size almost three times the maximum value of `Q0`. As described in the previous section, having too small buffers when there are both near and far sources will make bandwidth allocations unfair, even

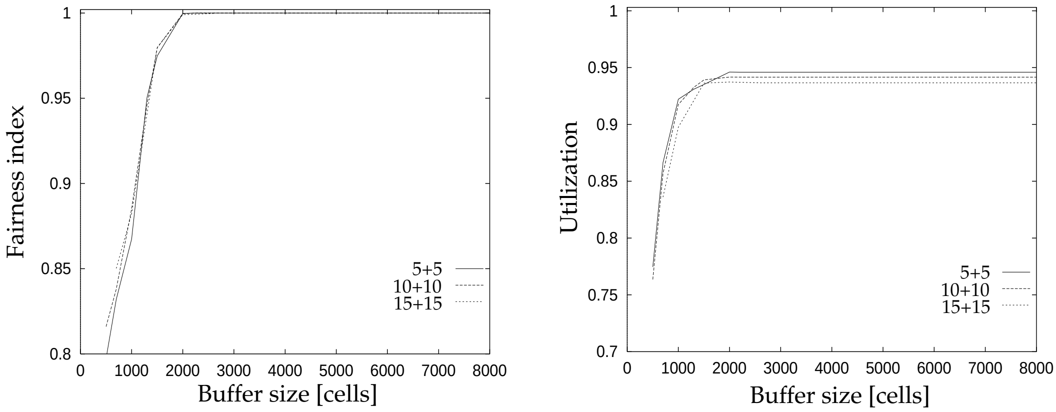


Figure 19: Fairness index and utilization for N+M configuration with 10, 20 and 30 sources.

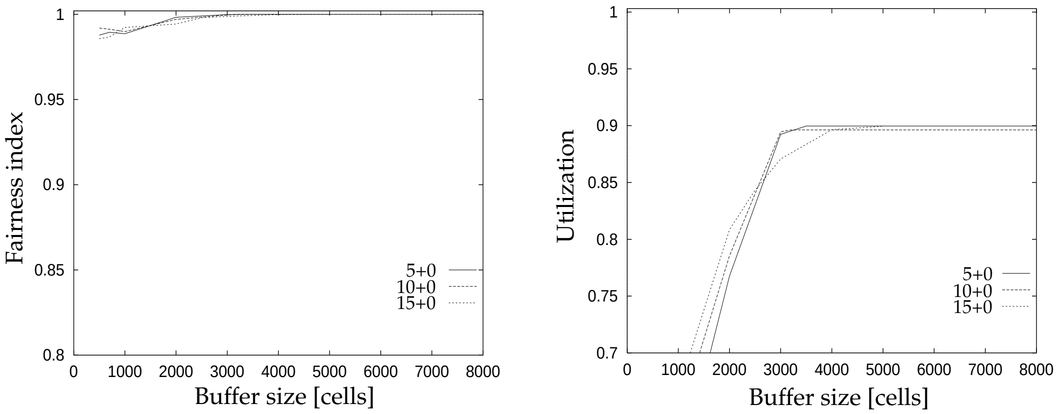


Figure 20: Fairness index and utilization for N+M ABR configuration without the near sources.

though the near or far sources among themselves are fair.

In figure 20, the near sources have been removed. The fairness is high, but now it takes a long time before any source will know about any new bandwidth available, and there are no big queues to use as a backup during these underloads as for UBR. This will bring the utilization down. A comparison of the utilization in figure 16 and figure 20 show that the threshold for the utilization using ABR is now closing in on the threshold for UBR. The feedback delay is 10 ms in these simulations and is still shorter than the time between changes in available bandwidth, which is 20 ms.

When the feedback delay is longer than the mean time between drastic changes in the available bandwidth, the information in the RM-cells will always be inaccurate. As a result of this, the ABR flow control does not function as it is supposed to. The amount of cells sent from the sources will not represent the current level of congestion. The behaviour of the sources will be more like that of TCP over UBR, with the exception that the mean peak cell rate will be lower. Compared with a TCP source using UBR, this will result in a lower utilization obtained with the same buffer size, but the fairness index will still be higher.

In figure 21 the propagation delay of the links between the switches have been changed to 20 ms instead of 5 ms. Apart from this, the same configuration as above, with 10 far sources, is used. The feedback delay will now be 40 ms, and the round-trip time 80 ms. The time between changes in the available bandwidth is still 20 ms, and therefore, the information in the RM-cells will always be inaccurate. As seen in the figure, ABR still have a better fairness index, but is outperformed by UBR with respect to the utilization.

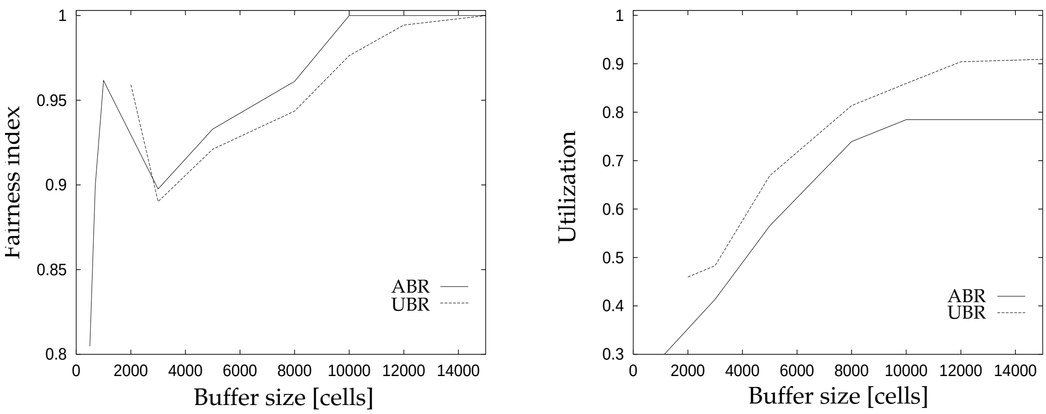


Figure 21: Comparison of fairness index and utilization for ABR and UBR when feedback always is inaccurate.

5.2.3 Packet discard policies

For TCP sources using the UBR service, buffer overflows play an important role for the flow control mechanism. It is therefore important to consider the actions taken by the switch at a buffer overflow. Since the ABR service tries to avoid buffer overflows, they are not that frequent for ABR as for UBR, and not as important. The studies below are done using the UBR service only.

In figure 22, the fairness index and utilization for the 15-source LAN configuration, using different discard policies, are shown. The figure shows the performance of the selective drop and early packet discard schemes compared with not using any discard policy at all.

Without any discard policy, the fairness index is generally below 0.9, giving very unfair behaviour. The early packet discard scheme is quite fair and have high utilization. The selective drop scheme is even more fair, and the utilization is nearly as high as for early packet discard. To get high overall performance with UBR, the most important issue is to improve the fairness.

Figure 23 shows traces from simulations using either no discard policy at all, or one of early packet discard or selective drop. The 15-source LAN configuration is used, with a buffer size of 2000 cells. For the configuration without a discard policy, most of the sources experience long timeouts. After five seconds, there is a huge difference in the number of packets delivered among the sources. Still, at this buffer size, this configuration has a relatively high fairness index compared with other buffer sizes, as seen in figure 22.

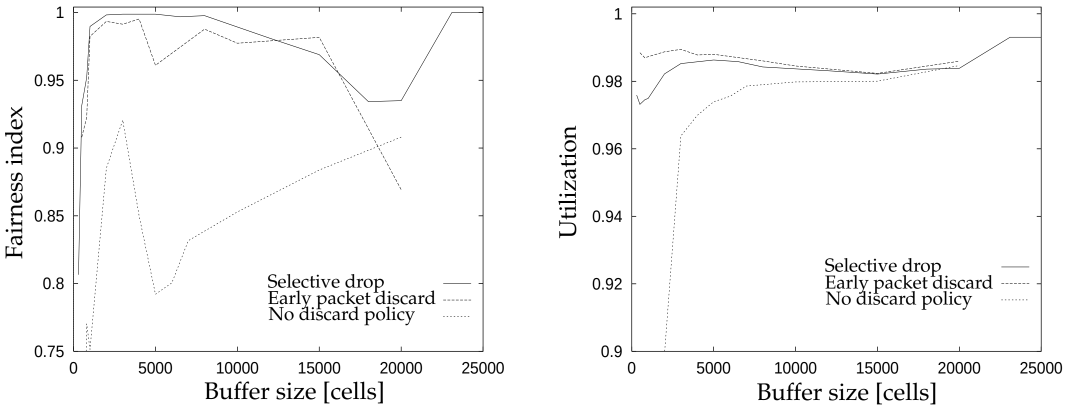


Figure 22: Fairness and utilization for the 15-source LAN configuration using the UBR service with different discard policies.

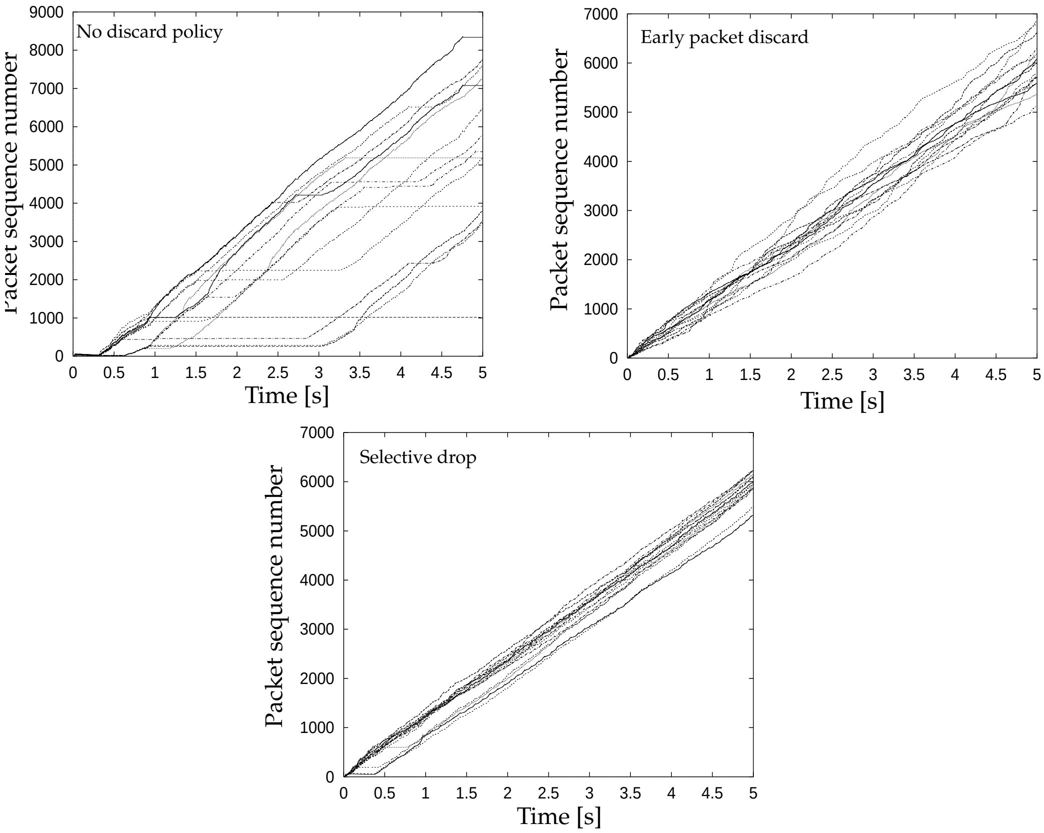


Figure 23: Traces of the sent packets for the 15-source LAN configuration with a buffer size of 2000 cells.The fairness index for these configurations were 0.885 without any discard policy, 0.993 with early packet discard and 0.998 with selective drop.

In figure 24 the buffer occupancy for the three configurations during 1 second of simulation is shown. If no discard policy is used, all sources but an occasional one suffer from packet loss at each buffer overflow. This results in all sources being synchronized, meaning that all sources stop sending and start retransmitting a lost segment at the same time.

If the round-trip delay is long, this synchronization might result in that the dips in buffer occupancy will empty the buffer for some time. The links will first be congested and then left unused, over and over again. The early packet discard scheme is slightly better, but is not able to break the synchronization. Selective drop, on the other hand, is much better. At each overflow, about half of the sources suffer from packet loss at a time. Because of this, there is always a flow of packets through the link. A similar study of the performance of various discard policies for TCP over UBR is presented in [11]. As seen in the cases for the early packet discard and selective drop, the discard threshold could be set higher than the default 90% used for this configuration.

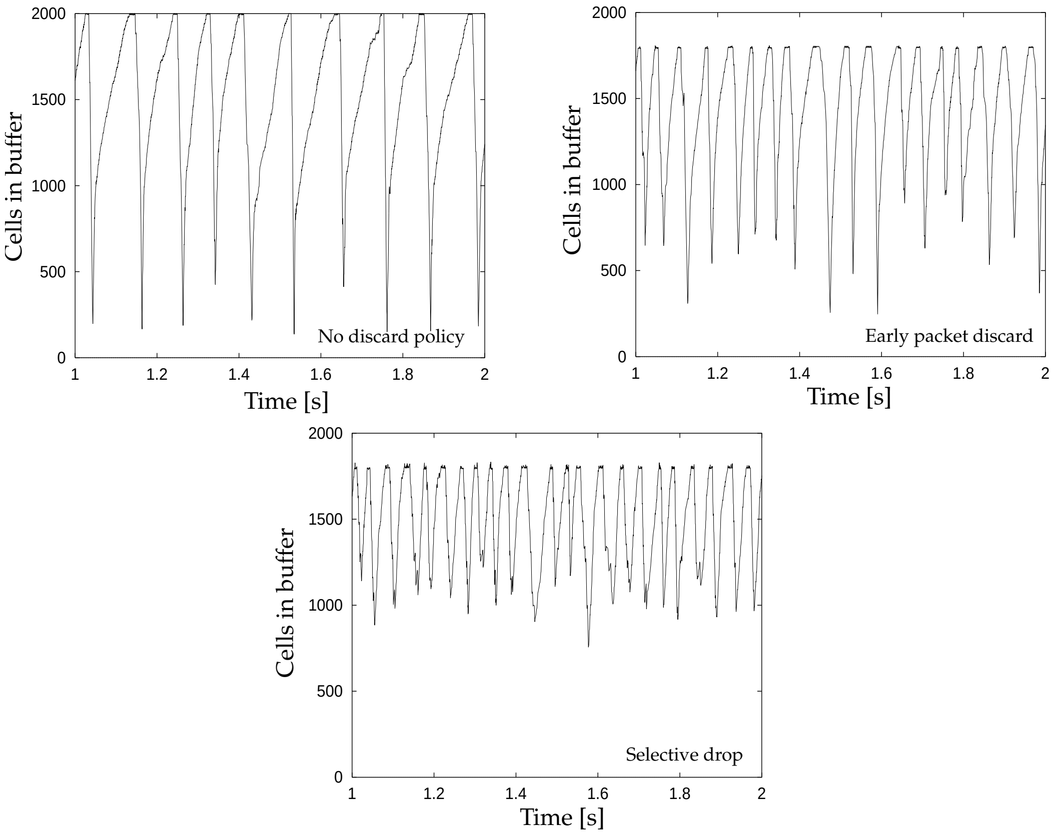


Figure 24: The buffer occupancy for the 15-source LAN configuration with a buffer size of 2000 cells, with different discard policies during 1 second of simulation.

6 Conclusions

To get high performance for TCP over ATM, it is very important to have knowledge about the properties of TCP and ATM upon which the performance is dependent. The choice of several parameter settings of TCP and ATM, as well as the characteristics of the network, have great impact on the efficiency.

6.1 TCP

The Reno implementation, with the fast recovery algorithm, is not able to handle multiple segment loss. Therefore, connections that will have frequent packet loss, like UBR connections, should use an implementation without the fast recovery algorithm. SACK TCP will be able to handle this situation and still maintain a high throughput.

A retransmit timer with fine granularity will decrease the time spent waiting for a timeout. However, the timeout procedure should not have to be invoked very often, since the fast retransmit handles most retransmissions. Therefore, it is better to have a coarse timer, avoiding unnecessary retransmissions, than to have a fine one. This is especially true for WANs and ABR connections where the delay variations are bigger.

TCP sources that experience successive fast retransmits will have very poor performance. It is mainly sources with large windows that suffer from multiple non-consecutive packet loss within the same window. The proposed bugfix makes the recovery from the loss faster. The best solution to avoid this situation will probably be to use SACK TCP.

6.2 TCP over UBR

There is a clear threshold where the fairness index collapse if buffers are too small. Having a buffer size somewhat larger than the threshold will give high performance. Larger buffers than that will either decrease the fairness, since the number of packets lost per source will not get even, or introduce a delay that will be very long. If the TCP segment size is increased, the required buffer space will also be increased.

If there are sources with different round-trip delays, the buffers need to be large enough not to introduce unfair behaviour among the far and near sources.

The use of a discard policy is very important for UBR connections. The early packet discard scheme is relatively simple to implement and gives high performance. The selective drop scheme is more complex but gives slightly higher fairness, and is able to break the synchronisation of the TCP sources, which may otherwise result in low utilization of the links.

6.3 TCP over ABR

As with UBR, TCP over ABR has very clear thresholds where the fairness drops. The buffers often just have to be somewhat bigger than the maximum of the target queue length `Q0`. The fixed target queueing delay `T0`, upon which `Q0` is dependent, should be chosen so that the queue can be used to keep utilization high at most of the temporary underloads, but short enough to not cause unnecessary delay.

TCP over ABR will not benefit from a finer granularity of the retransmit timer. Losses are very unusual. If the retransmit timer granularity is made finer, TCP over ABR will be very sensitive to delay variations.

For ABR to function as it is supposed to, the feedback delay of a source must be shorter than the time between drastic changes in the level of congestion. This condition is very often true for a LAN, but not necessarily for a WAN. Generally, the use of the ABR service improves fairness and decreases the buffer sizes required compared with UBR. But as the feedback delay gets longer, the benefits from using the far more complex ABR service becomes smaller and is harder to motivate.

7 Further studies

7.1 SACK TCP

The Selective Acknowledgement (SACK) TCP [8] is a variant of the Reno implementation where the receiver is able to inform the sender about which segments were received and which were not. Up to three blocks of segments can be defined in each acknowledgement, if the connection has suffered from multiple non-consecutive segment loss. This will enable the source to retransmit only those segments that were really lost. It will also resolve most of the situations where the fast recovery would timeout or where the fast retransmit would be invoked several times for the same window.

This implementation will probably be the next standard TCP used worldwide, and will improve performance of TCP generally. Studies should be done to explore the interaction of SACK TCP and ATM. SACK TCP could further improve the fairness for UBR connections since there should not be any exceptional situations where TCP will experience poor performance.

7.2 Changes to the model

If further studies of TCP over ATM is to be done with this model, some enhancements could be made to improve the accuracy of the results. First, the VBR traffic should be implemented as sources sending data through the network according to a traffic model, behaving like real VBR sources. Second, the network configurations should represent existing environments that might be more complex, introducing far more interesting situations that would not appear in a symmetric imaginary network. Further, the TCP sources could be modelled to send limited amounts of data, behaving like real applications, instead of acting like infinite FTP file transfers.

8 References

[1] "The ATM Forum Traffic Management Specification Version 4.0", ATM Forum Traffic Management af-tm-0056.000, April 1996. Available as ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps

[2] Raj Jain, Shiv Kalyanaraman, Rohit Goyal, Sonia Fahmy, Ram Viswnathan, "ERICA Switch Algorithm: A Complete Description", ATM Forum 96-1172, August 1996. Available through http://www.cis.ohio-state.edu/~jain.

[3] Van Jacobson, "Congestion Avoidance and Control", Proceedings of the SIGCOMM '88 Symposium, pp 314-329, August 1988. An updated version is available as ftp://ftp.ee.lbl.gov/papers/congavoid.Z.ps.

[4] W. Richard Stevens, "TCP/IP Illustrated, Volume 1", Addison-Wesley, 1994.

[5] Allyn Romanow, Sally Floyd, "Dynamics of TCP Traffic over ATM Networks", IEEE Journal on Selected Areas in Communications, vol. 13, no. 4, pp. 633-641, May 1995. Available as ftp://ftp.ee.lbl.gov/papers/tcp_atm.ps.Z.

[6] Juha Heinanen, Kalevi Kilkki, "A Fair buffer allocation scheme", Unpublished Manuscript.

[7] Kevin Fall, Sally Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", Computer Communications Review, July 1996. Available as ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[8] Matt Mathis, Jamshid Mahdavi, Sally Floyd, Allyn Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996

[9] Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, "Performance of TCP over UBR+", ATM Forum 96-1269, October 1996. Available through http://www.cis.ohio-state.edu/~jain.

[10] Sally Floyd, "TCP and Successive Fast Retransmits", Technical report, ftp://ftp.ee.lbl.gov/papers/fastretrans.ps, May 1995

[11] Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, "Further Results on UBR+: Effect on Fast Retransmit and Recovery", ATM Forum 96-1761, December 1996. Available through http://www.cis.ohio-state.edu/~jain.

9 Abbreviations and Acronyms

AAL5	ATM Adaption Layer type 5
ABR	Available Bit Rate
ATM	Asynchronous Transfer Mode
CBR	Constant Bit Rate
CCR	Current Cell Rate
CI	Congestion Indication
EFCI	Explicit Forward Congestion Indication
ER	Explicit Rate
FTP	File Transfer Protocol
ICR	Initial Cell Rate
IP	Internet Protocol
LAN	Local Area Network
MCR	Minimum Cell Rate
MTU	Maximum Transmission Unit
NI	No Increase
PCR	Peak Cell Rate
QDLF	Queue Drain Limit Factor
RM	Resource Management
RDF	Rate Decrease Factor
RIF	Rate Increase Factor
RTO	Retransmit Timeout
TCP	Transmission Control Protocol
UBR	Unspecified Bit Rate
VBR	Variable Bit Rate
VC	Virtual Channel
VS/VD	Virtual Source / Virtual Destination
WAN	Wide Area Network

Appendix A: Fixed and variable parameter settings

A summary of the fixed and variable parameter settings used for the simulations presented in chapter 5 is listed below. These settings override the default parameter settings listed in chapter 4.

A.1 Properties of TCP

The simulations made to analyse the properties of TCP include a comparison between the Tahoe and the Reno implementations and the use of various granularities of the retransmit timer. The problem with successive Fast Retransmits is also demonstrated.

Table 1: Tahoe vs. Reno implementation.

Fixed	Variable
TCP over UBR LAN configuration 15 sources Buffer size 3000 cells No VBR traffic	TCP implementation Retransmit timer granularity

Table 2: Problem with successive Fast Retransmits.

Fixed	Variable
TCP over UBR WAN configuration 15 + 15 sources Buffer size 24000 cells No VBR traffic	Use of bugfix for successive Fast Retransmits

A.2 Properties of TCP over UBR

The buffer size required to get high performance for TCP over UBR, using various settings of the buffer size and number of sources, is presented. The measure of performance is made using the fairness index and the utilization for the LAN and WAN configurations.

Table 3: Buffer requirements for LAN.

Fixed	Variable
TCP over UBR LAN configuration	Buffer size Number of sources Segment size

Table 4: Buffer requirements for WAN.

Fixed	Variable
TCP over UBR WAN configuration	Buffer size Number of sources Link delay

A.3 Properties of TCP over ABR

The buffer size required to get high performance for TCP over ABR, using various settings of the buffer size and number of sources, is presented. The measure of performance is made using the fairness index and the utilization for the LAN and WAN configurations.

Table 5: Buffer requirements for LAN.

Fixed	Variable
TCP over ABR LAN configuration	Buffer size Number of sources Segment size

Table 6: Buffer requirements for WAN.

Fixed	Variable
TCP over ABR WAN configuration	Buffer size Number of sources Link delay

A.4 Discard policies

The importance of the use of a discard policy is demonstrated along with a comparison of the early packet discard scheme and the selective drop scheme. The efficiency of the schemes for various buffer sizes and the ability to break synchronization are studied.

Table 7: Efficiency of the discard policy schemes.

Fixed	**Variable**
TCP over UBR LAN configuration 15 sources	Discard policy Buffer size

Table 8: Ability to break synchronization.

Fixed	**Variable**
TCP over UBR LAN configuration 15 sources Buffer size 2000 cells	Discard policy