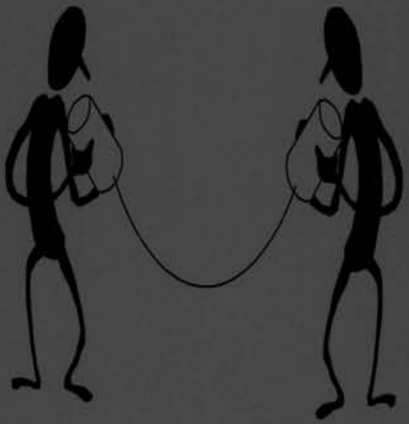
Internet Traffic over Broadband Networks

Transferring TCP/IP traffic over ATM

M.J. Bastiaansen



WBBM Report Series 43

Delft University Press

731240

Internet Traffic over Broadband Networks

Transferring TCP/IP traffic over ATM

WBBM Report Series 43

WBBM
Delft University of Technology
Faculty of Information Technology and Systems
Department of Mathematics and Computer Science
Room ET 05.040
Mekelweg 4
2628 CD Delft, The Netherlands
Phone +31 15 278 16 35
Fax +31 15 278 72 55

Internet Traffic over Broadband Networks

Transferring TCP/IP traffic over ATM

M.J. Bastiaansen

Delft University Press, 1999

The WBBM Report Series is published by:

Delft University Press
P.O. Box 98
2600 MG Delft, The Netherlands
Phone +31 15 278 32 54
Fax +31 15 278 16 61

Editors:

E. de Klerk
H. van Maaren

Delft University of Technology
Faculty of Information Technology and Systems
Department of Mathematics and Computer Science

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK DEN HAAG

Bastiaansen, M.J.

Internet Traffic over Broadband Networks / M.J. Bastiaansen - Delft : Delft University Press. - Ill. - (WBBM Report Series 43)
ISBN 90-407-1968-3
NUGI 841
Trefw.: TCP, IP, Internet, Broadband networks

Contents

Acknowledgements

This report was written as part of the two year, post graduate course offered by the Technical University (TU) of Delft under the name 'Mathematical Support and Decision Models'. The report was both commissioned by and funded by KPN Research, the research department of the Dutch telecom operator KPN Telecom.

To complete the course and merit the title of 'Master of Technological Design', this report was both presented and defended before a review committee consisting of the following people:

- Ir. J. G. Snip, KPN Research
- Ir. D. Sparreboom, TU Delft
- Prof. Dr. Ir. P. van Mieghem, TU Delft
- Prof. dr. R.M. Cooke, TU Delft
- Dr. H. van Maaren, TU Delft

There are a number of people whom I wish to thank for their support and cooperation over the past year, without which this report could not have been possible. First and foremost, I would like to thank both Jan Gerard Snip, at KPN Research, and Dirk Sparreboom, at the TU Delft, for their continual support and guidance throughout the making of this report.

At KPN Research, I would additionally like to thank the SEQUOIA project team members for accepting me into their team. I would especially like to thank Maurits de Graaf, for reviewing my work on CAC algorithms, and Bart Gijssen, for both reviewing this report and cooperating with me on the ABR analysis. Furthermore, I would also like to thank my room mate, Kees van der Wal, for always being willing to answer any and all of my questions.

At the TU Delft, I would also like to thank my fellow twAIOs, especially Ardo, Marco, Mark, Marjanke and Petri, for making my stay at the university an enjoyable one. Lastly, I would like to thank Hans van Maaren and Etienne de Klerk for their support in general during the past two years.

Chapter 1 Introduction

Nowadays, the Internet seems ubiquitous. Where barely nine years ago the Internet was mainly the domain of universities and a handful of internet pioneers, today there are more than an estimated[1] 179 million users world wide and businesses are rapidly discovering the potential of electronic commerce. At the basis of this success lies the internet protocol, IP for short. It is thanks to this protocol that computers everywhere, whether a personal computer or a powerful Cray, are able to interconnect and form the fast growing network of computers called the Internet.

IP is a relatively simple protocol. Data that has to be transferred over an IP network, be it an image, an audio fragment or a large document, is sent to the network in the form of packets. Each individual packet is then routed to the desired destination by the IP network itself, implying that subsequent packets may well follow different routes through the network. However, IP is not fail safe. No transfer guarantees are given and packets may be lost in the network if congestion occurs. For this reason IP is often referred to as a best effort transfer service, the network doing the 'best' it can.

To deal with this packet loss, IP is commonly paired with an additional protocol called the transmission control protocol, TCP for short. TCP is a more complex protocol designed to reliably transfer IP packets over an IP network. If packets are lost in the network then TCP ensures that they are resent. In addition, TCP regulates the flow of IP packets, decreasing the flow if the network is congested. However, the combination of TCP and IP still fails to provide transfer guarantees such as an upper bound on the number of packets lost by the IP network or the transfer delay any one given packet experiences.

Work is currently under way to provide such quality of service (QoS) guarantees for the IP protocol, but there is still a lot of work to be done. In contrast, broadband networks based on the asynchronous transfer mode, ATM for short, already provide the desired QoS guarantees. In contrast to the TCP/IP protocols, which provide for only one type of best effort transfer service, the ATM network caters for a variety of transfer services ranging from best effort to guaranteed delivery. These transfer services are also called ATM transfer capabilities, or ATCs for short.

Obviously, it would be ideal if TCP/IP traffic could be transferred over an ATM network and in this way benefit from the QoS guarantees provided by the ATCs. However, accomplishing this is far from simple and a lot of issues have to be

[1] Source www.nua.ie

resolved first. One of these issues is that an ATM network transfers data in small, fixed size chunks called cells. An IP packet is therefore chopped into numerous smaller cells prior to transfer, and the loss of any one of these cells implies the loss of the complete IP packet. Another issue is that TCP regulates the flow of IP packets based on the assumption of a best effort transfer service and is thus less suited for ATCs which provide QoS guarantees. As a result, TCP/IP traffic may have difficulty cashing in on the QoS guarantees provided by an ATM network.

In an effort to deal with some of these issues, a new ATC is currently being standardised under the name Guaranteed Frame Rate, or GFR for short. The GFR ATC is the only ATC that explicitly takes into account the packet based nature of IP traffic and thus promises to be better suited for the transfer of TCP/IP traffic than any of the other ATCs. This entire report is dedicated to investigating if this is indeed the case.

To determine this, the advantages and disadvantages of using the various ATCs for the transfer of TCP/IP traffic are discussed in Chapter 2. On the basis of these discussions two ATCs are deemed promising and Chapter 3 discusses the first of these, called the Available Bit Rate (ABR) ATC, in more detail. The other ATC is not surprisingly GFR and a simplified network model is used in Chapter 4 to analyse the expected performance of TCP/IP traffic over this ATC as well as the ABR ATC. Chapter 5 then delves deeper into how QoS guarantees are provided by the GFR ATC, comparing two different connection admission control algorithms.

Chapter 6 represents the cumulation of all these efforts and details the set up used to simulate TCP/IP traffic over both ATCs. Based on the results obtained, conclusions are drawn about the expected performance of TCP/IP traffic over a real, operational ATM network. In addition, a non standardised ATC is introduced under the name GFR lite and is also simulated. The reason for this is that GFR lite is simpler than GFR and promises to be better suited for transferring TCP/IP traffic.

Finally, in Chapter 7 conclusions are drawn and recommendations are stated as to how TCP/IP traffic can be best transferred over an ATM network.

Chapter 2 Transferring TCP/IP traffic over ATM

This discusses the advantages and disadvantages of using the various ATM transfer capabilities (ATCs) for the transfer of TCP/IP traffic. The main focus is on the guaranteed frame rate (GFR) ATC, discussed in section 2.3, as this ATC is relatively new and was designed specifically for the transfer of packet based traffic such as TCP/IP traffic. However, the 'older' ATCs are also discussed in section 2.2, allowing conclusions to be drawn in section 2.4 about which ATCs are best suited for the transfer of TCP/IP traffic. Those ATCs deemed most suitable are simulated in Chapter 6 in order to provide some insight into the expected performance of TCP/IP traffic over them. First, however, a general discussion about how TCP/IP traffic is transferred over an ATM network is given in section 2.1.

2.1 TCP/IP over ATM

This section discusses how TCP/IP traffic is transferred over an ATM network irrespective of which ATC is used. Here TCP/IP traffic is defined as the traffic generated between a source and destination between which a TCP session has been established. This traffic is transferred between the source and destination over an ATM virtual connection (VC). This report does not discuss how such a VC is setup but simply assumes this to be the case.

Figure 1 illustrates the TCP, IP and ATM protocol stacks which are required to transfer the data belonging to a TCP session over an ATM network[2]. A detailed description of each layer is not given here as numerous books have been written on the subject, see [23] and [27] for example. Instead a brief description is given of the how data is sent to and received from the network. First, the data that has to be transferred is chopped into segments by the TCP layer and a 20 byte header is added. The maximum segment size (MSS) of these segments is negotiable per TCP session. Then, the TCP segments are handed down one by one to the IP layer where an additional 20 byte header is added, resulting in an IP packet. Subsequently, the ATM adaptation layer (AAL) adds an 8 byte AAL5 trailer before handing the resulting data block down to the ATM layer where it is chopped into blocks of 48 bytes. To each of these blocks a 5 byte header is added, resulting in a 53 byte ATM cell. Finally, the resulting ATM cells are handed down to the physical layer for actual transfer over an ATM VC. At the receiving end the data is reconstructed by traversing the layers in the opposite direction, starting down at the physical layer and ending back up at TCP layer. The receiver can determine which cells belong to which IP

[2] LLC/SNAP encapsulation was not assumed, see [15]

packet thanks to the AAL5 protocol. This ensures that a bit in the ATM header is set in each last cell of any given IP packet.

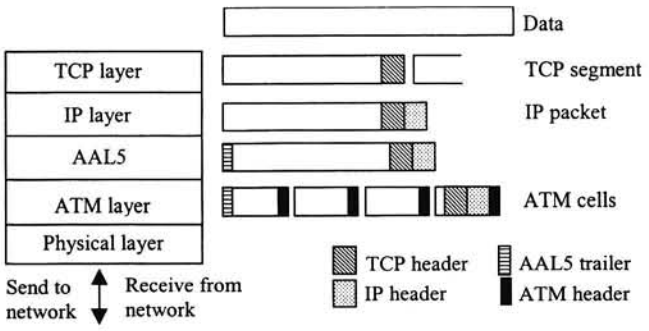


Figure 1 The layers and headers used in the transfer of TCP/IP traffic over ATM

An IP packet cannot be reconstructed, however, if even as little as one cell belonging to that IP packet has been lost or discarded. Such a packet is commonly referred to as a corrupt packet and is discarded at the receiving end. Therefore, if cell loss does occur in the ATM network then the transfer of the remaining cells belonging to the corrupt packet implies a waste of the cell rate available in the network. If these cells were discarded instead, the cell rate initially wasted in the transfer of the corrupt packet would now be available for the transfer of the other traffic in the network. Therefore, to increase the efficiency of the network a packet discard mechanism such as early packet discard (EPD), see for example [27], is desirable.

With respect to the mapping of TCP sessions to ATM VCs, two distinct mappings are of interest:
1. The **native** mapping, where the traffic belonging to *one* TCP session is transferred over one VC.
2. The **router** mapping, where the traffic belonging to *multiple* TCP sessions is transferred over one VC.

The latter case is common if two IP routers are interconnected via an ATM network, the prior if TCP/IP sources are directly connected to an ATM network.

The manner in which the traffic flow over the VC responds to cell loss is expected to be very different for either mapping. In the native mapping, cell loss triggers the TCP congestion control mechanisms, see [26], possibly resulting in a decrease in the volume of the traffic flow. In the router mapping, TCP sessions suffering cell loss also respond in the same way. However, if relatively few sessions suffer cell loss then the resulting decrease of their traffic should have only a minor impact on the total traffic flow of all the TCP sessions combined. In conclusion, it is expected that TCP will greatly influence the traffic flow over a VC for the native mapping, but that this influence will decrease as the number of TCP sessions over the VC increases.

2.2 Current ATCs

This section discusses the advantages and disadvantages of using the various ATCs already standardised and currently in use in operational ATM networks for the transfer of TCP/IP traffic. Each separate ATC is discussed in it's own subsection and no comparison is made between the different ATCs. Such a comparison is made in the last section, after the section discussing the GFR ATC.

2.2.1 Deterministic bit rate

The deterministic bit rate (DBR[3]) is standardised as an ATC by the telecommunication standardisation sector of the international telecommunication union (ITU-T), see [17], and has only two traffic parameters:

- the peak cell rate (PCR) and the associated cell delay variance (CDV) tolerance τ_{PCR}.

On the basis of these two parameters quality of service (QoS) guarantees are given by the network to those cells that pass the generic cell rate algorithm test: GCRA(PCR,τ_{PCR}), see [17]. Those cells that fail the test are discarded. Basically, all cells sent at the PCR or lower pass the GCRA test while all cells sent at a higher cell rate are discarded. As a result, DBR provides a QoS guarantee in terms of cells and this does not automatically imply any QoS guarantees in terms of packets. In the most extreme case the QoS guarantees could be satisfied without as much as one complete IP packet being delivered. For DBR this is very unlikely but for the ATCs discussed next, however, the chance of this happening is not so remote.

Anther disadvantage of DBR is that it does not match the traffic characteristics of TCP/IP traffic. DBR would be ideally suited if the volume of TCP/IP traffic were constant and always exactly equal to that which the PCR could handle. However, this is not the case as:

1. firstly, the slow start mechanism of TCP, see [26], produces a bursty traffic pattern which is far from constant, and
2. secondly, the same mechanism continually increases the volume of traffic being sent.

Chapter 4 discusses the traffic characteristics of TCP/IP traffic more fully.

As a result of this mismatch, network resources are not used efficiently. If the volume of TCP/IP traffic is larger than that which the PCR can handle but smaller than that which the ATM network can handle, then this traffic is still discarded. On the other hand, for a given DBR VC the network reserves a cell rate equal to PCR and this cell rate remains unavailable for the other traffic in the network, even during periods that no cells are being sent over the DBR VC.

[3] The constant bit rate (CBR) service category as specified by the ATM Forum, see [2], is identical to the DBR ATC.

2.2.2 Statistical bit rate

The statistical bit rate (SBR[4]) is standardised as an ATC by the ITU-T, see [17], and has five traffic parameters:

- the PCR and corresponding CDV tolerance τ_{PCR},
- the sustainable cell rate (SCR) and corresponding CDV tolerance τ_{SCR},
- the maximum burst size (MBS).

Furthermore, three variants of SBR are defined, namely SBR1, SBR2 and SBR3, each of which will be discussed next in a separate subsection. One thing all variants have in common, however, is that cells not passing the GCRA(PCR,τ_{PCR}) test are discarded. Also, like DBR, a disadvantage of SBR is that it provides a QoS guarantee in terms of cells and not in terms of packets.

2.2.2.1 SBR1

In this variant the cell loss priority (CLP) bit in the ATM header has no function. Cells passing the GCRA(SCR, (MBS-1)(1/SCR-1/PCR)+τ_{SCR}) test are given QoS guarantees, those cells failing the test are discarded. Basically, SBR1 allows bursts of not more than MBS cells to be sent at a rate of maximally the PCR, as long as the resulting average cell rate does not exceed the SCR.

Another disadvantage of SBR1 is that it does not match the traffic characteristics of TCP/IP traffic. This issue is discussed more fully in the previous section on DBR and the same arguments apply here. However, SBR1 is more suited to handle the bursty nature of TCP/IP traffic than DBR is due to the additional MBS traffic parameter.

2.2.2.2 SBR2

In this variant the CLP bit in the ATM header may be marked by the TCP/IP source, where marking implies changing the CLP bit value from the default value of zero to the value of one. Cells marked as CLP=1 do not receive any QoS guarantees and are transferred on a best effort basis. Cells marked as CLP=0 are subjected to the GCRA(SCR, (MBS-1)(1/SCR-1/PCR)+τ_{MCR}) test. Those cells that pass are given QoS guarantees, those that fail are discarded. Basically, SBR2 is identical to SBR1 with the added possibility of sending additional best effort traffic as CLP=1 cells. These cells are subject to discard if the network is congested. Quite often a packet discard mechanism is implemented in the network in combination with SBR in order to ensure that if one cell of any given IP packet is discarded, then so are the other cells of that packet.

An advantage of SBR2 is that it caters to the traffic characteristics TCP/IP traffic. Both the bursty nature of this traffic, as well as it's continuously increasing volume are catered for.

[4] The variable bit rate (VBR) service category specified by the ATM Forum, see [2], is identical to the SBR ATC.

2.2.2.3 SBR3

This variant is identical to SBR2 except that CLP=0 cells not passing the GCRA(SCR, (MBS-1)(1/SCR-1/PCR)+τ_{MCR}) test are now tagged[5] as CLP=1 instead of being discarded. A disadvantage of SBR3 in this respect is that the CLP tagging is not packet aware. That is, if a cell of a particular packet is marked then the other cells of that packet are not marked by default as well. As a result a certain portion of a packet may be transferred on a best effort basis, while the remaining portion receives QoS guarantees. If all packets are split in this way and the network is congested, then it is possible for only corrupt packets to arrive at the destination.

2.2.3 Available bit rate

The available bit rate (ABR[6]) is standardised as an ATC by the ITU-T, see [17], and has four traffic parameters:

- the PCR and corresponding CDV tolerance τ_{PCR},
- the minimum cell rate (MCR) and corresponding CDV tolerance τ_{MCR}.

ABR is unique as an ATC in that it has a congestion control mechanism. An ABR VC receives feedback from the network specifying the maximum allowed cell rate (ACR) at which cells may be sent. This ACR fluctuates dynamically based on the amount congestion in the network but is always larger than or equal to the MCR, and smaller than or equal to the PCR. As long as the ABR VC does not exceed the ACR then all of the cells receive QoS guarantees. If the ACR is violated then cells may be discarded by the network. Again, like for SBR and DBR, a disadvantage of ABR is that the QoS guarantee is specified in terms of cells and not in terms of IP packets.

Another disadvantage of ABR is that, for the ABR congestion control mechanism to work, the TCP/IP source must be able to regulate the flow of it's traffic in accordance with feedback received from the network. Current TCP/IP sources typically send traffic the moment it is generated. From the viewpoint of the source the latter behaviour is much simpler to implement.

TCP also has a congestion control mechanism by which it responds to packet loss in the network. Having two different and independent congestion control mechanisms operating at the same time might be seen as a disadvantage of ABR, especially if the one mechanism hinders the other. However, ABR has the potential advantage that virtually no cells are lost, in which case the TCP congestion control mechanism is not active at all.

[5] If the CLP bit is set by the source it is called marking, if the CLP bit is set by the network it is called tagging.

[6] ABR is also specified as service category by the ATM Forum, see [2].

2.2.4 UBR

The unspecified bit rate (UBR) is not standardised as an ATC by the ITU-T but is only specified as a service category by the ATM Forum, see [2]. Only two traffic parameters are defined:

- the PCR and associated CDV tolerance τ_{PCR}.

The UBR ATC provides no QoS guarantees whatsoever. All cells passing the GCRA(PCR,τ_{PCR}) test are transferred on a best effort basis, meaning subject to the availability of network resources. Cells failing the GCRA test are discarded. Basically, UBR allows a source to generate any traffic pattern as long as the cell rate is always lower than the PCR.

An advantage of UBR is that it matches the traffic characteristics of TCP/IP traffic perfectly. No restrictions are placed on how bursty the traffic is, and the traffic volume is only restricted by the PCR. As this value is often chosen equal to the maximum possible in the network, namely the link cell rate, the latter is often no restriction at all. If the network is congested then cells are discarded. Often a packet discard mechanism is implemented in the network in combination with UBR to ensure that if one cell of a particular IP packet is discarded then so are all the other cells.

UBR is a best effort service, exactly like IP, and no use is made of the QoS guarantees available in an ATM network. One would expect, therefore, the performance of TCP/IP over UBR to be identical to the performance of TCP/IP over an all IP network. However, this is not the case as in an all IP network all traffic is treated equal[7], whereas in an ATM network traffic is treated on the basis of priority. A disadvantage of UBR in this regard is that UBR has the lowest priority of all the ATCs. This implies that the amount of network resources available for UBR will in general be lower than it would have been if all the ATCs had equal priority.

2.3 The GFR ATC

This section discusses the advantages and disadvantages of using the GFR ATC for the transfer of TCP/IP traffic. First, the definition of the GFR ATC is given, after which it's suitability for TCP/IP traffic is discussed. Then, GFR is compared with SBR as both ATCs are similar in many respects. These discussions serve as input for the final section of this where conclusions are drawn about which ATCs are likely to be the best suited for the transfer of TCP/IP traffic.

2.3.1 Definition

In this subsection the definition of the GFR ATC is given as it is currently being standardised by the ITU-T, see [17]. The ATM-Forum has also recently specified the

[7] The IP header does have a priority field but this is ignored by most IP routers.

GFR as a service category, see [2], but there are minor differences between both standards. These differences will not be discussed in this report and the ITU-T standard is the default standard used unless explicitly stated otherwise. It should be noted that GFR was specifically designed for the transfer of frames in general, not only IP packets. Therefore, in line with the terminology used in the ITU-T standard, this section uses the term frames instead of the term packets. Within the context of this report, however, frames will be understood to refer to IP packets only.

The traffic characteristics of a GFR VC are captured in a connection traffic descriptor, which is specified by the following six traffic parameters:

- the PCR and associated CDV tolerance τ_{PCR},
- the MCR and associated CDV tolerance τ_{MCR},
- the maximum frame size (MFS),
- and the MBS.

Based on this connection traffic descriptor an ATM *cell* is defined as **conforming** if the following three conditions are met:

1. the cell passes the GCRA(1/PCR,τ_{PCR}) test[17]. Basically, this is the case if all cells are sent at a cell rate lower or equal to the PCR.
2. the cell is either the last cell of a frame or the number of cells in this frame up to and including this cell is less than MFS.
3. the CLP bit in the ATM header of the cell has the same value as the CLP bit in the header of the first cell of the frame to which the cell belongs.

If a cell violates one or more of the above three conditions then it is defined as **non-conforming** and may be discarded by the network.

A *frame* is defined as conforming if all of it's cells are conforming and non-conforming as soon as one or more if it's cells are non-conforming. Furthermore, a frame is defined as a CLP=0 frame if all of it's cells have a CLP value equal to zero, and is defined as a CLP=1 frame if all of it's cells have a CLP value equal to one.

Besides specifying the values of the GFR traffic parameters, a source wishing to establish a GFR VC must also specify a requested QoS class[8]. In such a class the performance objectives of the network are specified. For GFR the only objective specified is the maximum allowed CLR and this objective must be met by the network for all cells belonging to conforming CLP=0 frames which also pass the following frame based GCRA (F-GCRA) test[17]:

$$\text{F-GCRA}(1/MCR, (MBS-1)(1/MCR-1/PCR)+\tau_{MCR}).$$

Simply stated, GFR not only guarantees that the cell rate at which cells are transferred across the network is always at least equal to MCR, but also guarantees

[8] A frame based QoS class is currently being standardised by the ITU-T.

that no corrupt frames are delivered. Frames passing the F-GCRA test will henceforth be referred to as **QoS eligible** frames, those failing the test will be referred to as **non-QoS eligible** frames. Figure 2 illustrates the conditions a frame must satisfy in order to be considered QoS eligible.

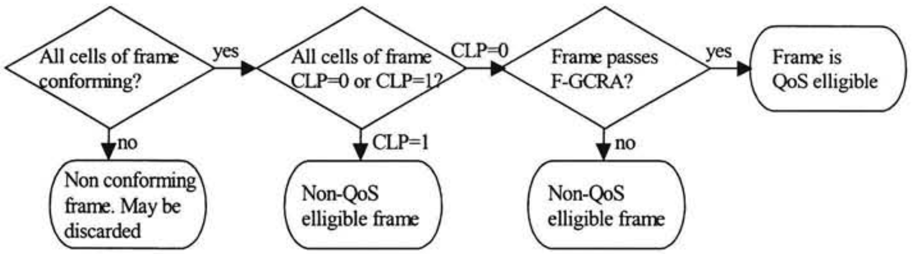


Figure 2 Flowchart determining whether a frame is QoS eligible or not.

The two following examples describe the QoS commitment that a source establishing a GFR VC can expect to receive:

1. If the source sends only CLP=0 frames, each consisting of fewer than MFS cells, and the source sends cells at a cell rate less than or equal to the MCR, then all of these frames should be delivered by the network with a minimum of cell loss.

2. If the source sends only CLP=0 frames, each consisting of fewer than MFS cells, and the source sends cells in bursts of maximally MBS cells at a cell rate not in excess of the PCR and follows each burst with an idle period, during which no cells are sent, long enough to ensure that the mean cell rate of the source is equal to or lower than the MCR, then all of these frames should be delivered across the network with a minimum of cell loss.

Figure 3 illustrates the traffic patterns which correspond with the above two examples.

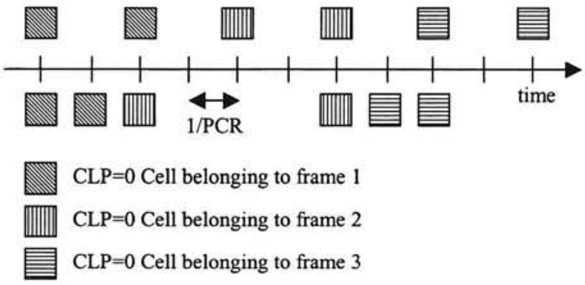

time
1/PCR
CLP=0 Cell belonging to frame 1
CLP=0 Cell belonging to frame 2
CLP=0 Cell belonging to frame 3

Figure 3 Two traffic patterns for which all frames should be delivered by a GFR VC with a PCR=2×MCR, a MFS=2 cells, and a MBS=3 cells.

In addition to the above QoS commitment, the GFR ATC is expected to deliver both conforming, non-QoS eligible CLP=0 frames and conforming CLP=1 frames on a best effort basis, meaning subject to the availability of network resources. If congestion occurs in the network then these frames may be discarded and it is expected that the CLP=1 frames belonging to a particular GFR VC are discarded prior to the CLP=0 frames belonging to the same VC. However, it is also expected that the best effort traffic belonging to a particular VC should not receive priority over the best effort traffic of another VC, even if for example the first VC sends only CLP=0 best effort frames while the second VC sends only CLP=1 best effort frames.

With respect to the F-GCRA test two variants of GFR exist:

1. GFR1, where the network does not perform CLP tagging,
2. and GFR2, where the network may tag frames that do not pass the F-GCRA test.

If tagging occurs by default for GFR2 then the CLP bit denotes whether a frame is QoS eligible or not. For GFR1 this is not the case as CLP=0 frames are not necessarily all QoS eligible.

2.3.2 Transferring TCP/IP traffic over GFR

This subsection discusses the advantages and disadvantages of using the GFR ATC for the transfer of TCP/IP traffic. GFR is unique as an ATC in that it is the only ATC that is specifically designed to deal with the transfer of frame based traffic such as IP over an ATM network. This is most clearly illustrated by the QoS commitment which states that the cells which are delivered by the network should belong to complete frames only. For all the other ATCs the QoS commitment is stated in terms of cells and this does not automatically imply any QoS commitments in terms of frames. Obviously, the frame based QoS commitment is an advantage of GFR.

Another advantage of GFR is that all cells in a frame are marked or tagged with the same CLP value. This allows a TCP/IP source to tell the network which packets it considers important and which it considers less important. This is especially true for the GFR1 variant of GFR as tagging is not allowed in the network and all packets are transferred with the CLP value left as marked by the source. For the GFR2 variant the source has less control as the CLP value may be changed if tagging occurs in the network.

For the native mapping, where only one TCP session is transferred over a VC, the CLP distinction is of little use as all the packets are equally important. For the router mapping, however, there are multiple TCP sessions over the same VC and the CLP bit can be used for a number of purposes. For example:

- All TCP acknowledgement (ACK) packets could be sent as CLP=0 while the remaining data packets are sent as CLP=1. This is also valid for the native mapping.
- Based on the priority fields in the IP header, high priority packets could be sent as CLP=0 while the remaining lower priority packets are sent as CLP=1.

- During a given time interval the packets of a particular VC are all sent as CLP=0 while the packets of the other VCs are all sent as CLP=1. After the time interval elapses the preferred VC is changed so that each VC in turn is given a chance to send it's packets as CLP=0.

An additional advantage of GFR is that it caters for a 'dumb' TCP/IP source. That is, GFR does not require a TCP/IP source to perform tagging or regulate it's traffic flow in any way. Instead a source may send traffic the moment it is generated, leave the CLP equal the default value of zero, and still receive QoS guarantees.

2.3.3 Comparing the SBR and GFR ATC

In this subsection the differences and similarities between the SBR and GFR ATCs are highlighted. The reason for this is that the definition of the SBR ATC is similar to the definition of the GFR ATC in many ways, raising the issue of whether GFR is indeed a different ATC from SBR. For a telecom operator this question translates to whether or not GFR should be implemented in the network if SBR already is.

The GFR ATC has the same set of traffic parameters as SBR, with an additional MFS parameter. Similarly, like SBR, GFR also has two variants that use the CLP bit in the ATM header. In this regard, the GFR1 variant of GFR most resembles the SBR2 variant of SBR as tagging does not occur in the network for either variant. Likewise, GFR2 most resembles SBR3 as both variants allow tagging to occur in the network. The SBR1 variant of SBR differs from GFR in that the CLP bit is not used and will therefore not be considered further.

However, despite these similarities, there are also important differences. The most important is the nature of the QoS commitment which for GFR provides a commitment in terms of entire frames, whereas for SBR the commitment is specified in terms of cells only. As a result, in extreme cases SBR can fulfill it's QoS commitment without delivering as much as one complete IP packet. Another difference, linked to the first one, is that the discarding and tagging of cells by the network does not occur in a packet aware manner for SBR.

To decrease these differences SBR could be augmented as follows:

- Require the network to implement a packet discard mechanism in order to ensure that no corrupt packets are delivered by the network.
- Require a TCP/IP source to mark it's CLP=0 cells in such a way that all cells marked belong to complete packets and all of them pass the GCRA test. For SBR2 this implies that no CLP=0 cells are discarded by the network, whereas for SBR3 this implies that no CLP=0 cells are tagged as CLP=1 cells. The net result of this is that no corrupt CLP=0 packets should be delivered by the network.

The latter requirement complicates the behaviour required of the TCP/IP source. An advantage of GFR in this respect is that GFR caters for a 'dumb' source, allowing the source to simply send all traffic as CLP=0. For GFR2, packets not passing the

FGCRA are then tagged as CLP=1, whereas GFR1 transfers all packets with the CLP value intact.

The main difference that now remains between GFR and SBR is the MFS parameter. The advantage of this extra parameter is that it protects the network from a source sending an infinitely long CLP=1 packet. Such a packet could wreak havoc in the network if sent at a sufficiently high cell rate. To see this note that if the network delivers the first cell of the packet, it is also expected to deliver the subsequent cells. Achieving this could violate the QoS commitments made to the other VCs also present in the network. Possibly, the MFS parameter can also aid the network in implementing a more efficient packet discard mechanism.

In addition to the MFS parameter, GFR1 and SBR2 differ further in that GFR1 allows the source to mark all those packets it considers important as CLP=0, whereas for SBR2 this marking must take into account the GCRA test to avoid cell discard. As a result, SBR2 restricts the freedom a TCP/IP source has in separating important traffic from less important traffic. As to the GFR2 and SBR3 variants, these are now virtually identical.

To summarise, at the cost of additional complexity in the TCP/IP source and the ATM network, SBR can be augmented to resemble GFR almost completely and only GFR1 still offers certain advantages over SBR2. However, if SBR is not augmented then GFR is clearly more suited for the transfer of TCP/IP traffic. In conclusion, GFR, especially the GFR1 variant, forms a worthy addition to the family of ATCs currently already standardised.

2.4 Suitable ATCs

In this section conclusions are drawn as to which of the ATCs discussed above are most suited for the transfer of TCP/IP traffic. The rest of this report will be based on these conclusions as only those ATCs considered suitable will be discussed further in the followings. Those ATCs considered less suitable will not be discussed again.

Of the ATCs discussed above the DBR ATC and the SBR1 variant of the SBR ATC are deemed least suitable for the transfer of TCP/IP traffic. The reason for this is the mismatch between these ATCs and the traffic characteristics of TCP/IP traffic. As a result, it can be virtually impossible for a TCP/IP source to fully utilise the reserved cell rate equal to MCR and benefit from the QoS guarantees offered by the ATM network.

Based on the discussion in the previous section, the SBR2 and SBR3 variants of the SBR ATC are also considered less suitable than the GFR1 and GFR2 variants of the GFR ATC. The main reason for this is that both SBR variants require additional complexity in the TCP/IP source in order for SBR to be comparable to GFR, and even then GFR1 is still preferable to SBR2. Besides, it is very likely that at least a large portion of the TCP/IP sources will be 'dumb' and thus unsuited for use with SBR.

This leaves the ABR, UBR and GFR ATCs, all of which seem suitable for the transfer of TCP/IP traffic. However, the UBR ATC is not considered further because it provides no QoS guarantees and comparing it with the ABR and GFR ATCs is thus difficult. Besides, in a congested ATM network UBR is much less suited for TCP/IP traffic than ABR and GFR. Instead, a non standardised ATC called GFR lite is introduced in section 6.2.3 and studied in it's place. Basically, GFR lite is a stripped down version of the GFR1 variant of GFR that can also be considered as a sort of UBR with QoS guarantees[9]. This non-standardised ATC holds promise of being more suited for the transfer of TCP/IP traffic than GFR1, especially for the native mapping.

In conclusion, the ABR and GFR ATCs seem likely to be most suited for the transfer of TCP/IP traffic over ATM. In addition, a non-standardised ATC called GFR lite is also deemed suitable and will be introduced later on in section 6.2.3.

In conclusion, the ABR and GFR ATCs seem likely to be most suited for the transfer of TCP/IP traffic over ATM. In addition, a non-standardised ATC called GFR lite is also deemed suitable and will be introduced later on in section 6.2.3.

[9] Originally, the GFR ATC also grew from the idea of adding QoS guarantees to the UBR ATC and was initially called UBR+, see [13].

Chapter 3 ABR congestion control

This analyses the ABR congestion control mechanism implemented in the Ascend CBX 500 ATM switch, a switch commonly used in currently operating ATM networks. This mechanism allows a congestion control loop to be established between two Ascend switches, one acting as the sender and the other as the receiver. By means of this loop the receiving switch can indicate to the sending switch whether or not it is experiencing congestion and the sending switch can adjust the rate at which it is sending accordingly. The analysis is used in Chapter 6 in order to set up the ABR simulations.

Firstly, a simplified model of ABR traffic is introduced in section 3.1 upon which the analysis in the subsequent sections is based. Section 3.2 then defines two performance measures, the cell loss ratio (CLR) and the efficiency, which are subsequently used to analyse the performance of the ABR congestion loop in a number of different scenarios. Sections 3.3 and 3.4 analyse the first two of these scenarios, the ramp-up and ramp-down scenarios, where the amount of non ABR background traffic in the network is assumed to suddenly fluctuate. The results of these two scenarios are then used in section 3.5 which analyses the steady state scenario, where the amount of background traffic is assumed to be constant. Section 3.6 presents the results for the final scenario where the ABR congestion loop is simulated in the presence of continuously fluctuating background traffic. Section 3.7 concludes the with an analysis of ABR fairness issues.

3.1 Aggregate fluid flow model

This section introduces the model upon which the subsequent analysis of the ABR congestion control loop is based. The model is a simplification of what can be expected in an operational ATM network as multiple traffic flows over several different ABR VCs are grouped together to form an aggregate traffic flow over one single hypothetical ABR VC. This allows the analysis to be simplified considerably from a discrete event analysis to a fluid flow analysis, without the analysis losing it's validity for situations likely to occur in operational networks. The ABR mechanism used is also a slight simplification of the proprietary ABR mechanism implemented in the Ascend CBX 500, see Appendix 2.

Figure 4 illustrates the ABR congestion control loop for the case of only one VC. The sending switch is assumed to be greedy in that it always has data ready to send. The VC used to transfer this data to the receiving switch is defined by it's MCR and PCR traffic parameters and is assumed to have a negligible transfer delay, implying that data cells arrive almost instantaneously at the receiver. Congestion information pertaining to this VC is fed back from the receiver to the sender by means of resource management (RM) cells[10]. These are generated by the receiver periodically, one every T_{RM} seconds, and are assumed to arrive at the sender after a fixed transfer delay of τ seconds. The cell rate available for the ABR data at the receiving switch is a function of the background traffic only. This traffic is defined to have a higher priority than the ABR traffic. No subsequent ABR congestion control loop controlling the send rate of the receiver is assumed.

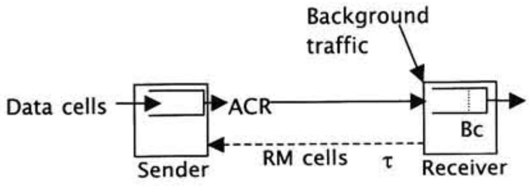


Figure 4 ABR congestion control loop for the aggregate fluid flow model

Congestion is detected at the receiver by means of a global congestion threshold B_c. If the number of cells in the receiving buffer is larger than the value of this threshold then the switch is congested and the congestion indication (CI) bit in the RM cell is marked (CI=1). Otherwise, the CI bit is left unmarked (CI=0).

The sender adapts the allowed cell rate (ACR) of the VC on the basis of the RM cell binary feedback. If an RM cell is received with it's CI bit marked then the switch lowers the ACR according to the following rule:

$$\mathrm{ACR}_n = \mathrm{ACR}_{n-1} \times (1 - \mathrm{RDF}), \qquad (1)$$

where RDF stands for the rate decrease factor and the index *n* denotes the nth ACR update. If an unmarked RM cell is received then the ACR is increased according to the following rule:

$$\mathrm{ACR}_n = \mathrm{ACR}_{n-1} + \mathrm{RIF} \times \mathrm{PCR} \qquad (2)$$

where RIF stands for the rate increase factor. However, the ACR can never be increased above the PCR nor lowered below the MCR. These parameters form the upper and lower boundaries between which the ACR is allowed to fluctuate.

In an operational network there will in general exist multiple VCs between any given two switches, each VC with it's own MCR, PCR, RIF and RDF values, and each VC

[10] Actually the Ascend CBX 500 uses either CCRM or BCM cells for feedback purposes, both proprietary RM cells. This assumes CCRM cells, referring to them simply as RM cells. Actually, the difference between the CCRM and BCM congestion control mechanisms is minor and the analysis presented in this is valid for both.

receiving it's own RM cells, one every T_{RM} seconds. This time interval of T_{RM} seconds between subsequent RM cells is fixed by the receiving switch and is equal for all VCs. However, the intervals need not be synchronised relatively, some VCs receiving their RM cells earlier than others.

Instead of analysing all of these VCs separately, the aggregate fluid flow model models the total traffic flow of all these individual VCs by only one aggregate VC. In order to do this the parameters of the individual VCs must first be aggregated to form the parameters of the aggregate VC. The MCR and PCR values of this aggregate VC are defined as the sum of the individual MCR and PCR values respectively:

$$MCR = \sum_i MCR_i, \quad PCR = \sum_i PCR_i \;,$$

where the summation over the index *i* denotes summation over the individual VCs. The RDF and RIF values of the aggregate VC are defined in terms of the parameters of the individual VCs by

$$(1 - RDF) = \frac{\sum_i MCR_i(1 - RDF_i)}{\sum_i MCR_i}, \quad RIF = \frac{\sum_i RIF_i \times PCR_i}{\sum_i PCR_i},$$

where, for the RDF formula, one additional assumption was made. Namely, the ABR congestion control loop was assumed to be fair, where fair is defined in this context as each VC having it's MCR proportional share of the total allowed cell rate. In formula this is equivalent with

$$\frac{ACR_i}{\sum_i ACR_i} = \frac{MCR_i}{\sum_i MCR_i}.$$

The issue of fairness is discussed more fully in section 3.7.

The T_{RM} interval of the aggregate VC is redefined as the original fixed T_{RM} interval divided by N, the total number of VCs being modelled:

$$T_{RM} \rightarrow \frac{T_{RM}}{N} \;.$$

The underlying idea here is that the individual T_{RM} intervals of the various VCs are assumed to be completely out-of-phase, and the starting times of the intervals are assumed to be spread evenly over one T_{RM} interval.

The ACR update rules for the aggregate VC are also altered in line with redefined T_{RM} interval:

$$ACR_n = ACR_{n-1}\,(1 - RDF)^{\frac{1}{N}}, \tag{3}$$

$$ACR_n = ACR_{n-1} + RIF \times \frac{PCR}{N}. \tag{4}$$

In the limit of an infinite number of VCs the redefined T_{RM} interval becomes infinitesimally small implying that the ACR is continuously updated instead of discretely. For the remainder of the analysis this is assumed to be the case. A continuous analysis of a basically discrete process is often referred to as a fluid flow analysis, hence the name aggregate fluid flow model.

3.2 Performance measures

This section defines two performance measures which are used in the subsequent sections to quantify the performance of the ABR congestion control mechanism. The first of these measures is the CLR and it is defined as the number of cells lost in the receiving switch, due to buffer overflow, divided by the total number of cells sent by the sending switch during any given time period. In formula this corresponds to

$$\mathrm{CLR} = \frac{\#\,\text{Cells lost}}{\#\,\text{Cells sent}},$$

where # denotes 'the number of'. The second performance measure is the efficiency which is defined as the fraction of the time the buffer at the receiving switch was not empty during any given time interval:

$$\text{efficiency} = \frac{\text{Total time - Time buffer empty}}{\text{Total time}}.$$

Note that as long as the buffer is full the receiving switch has cells to send implying that all of the available cell rate is used. It is only when the buffer is empty that the available cell rate is left unused, implying that the network resources are not being used as efficiently as they could.

3.3 Ramp-up

This section analyses both the behaviour and the performance of the ABR congestion control loop in the case that the amount of high priority background traffic suddenly decreases. This implies that the cell rate available at the receiver for ABR traffic suddenly increases. As a result the receiver will indicate back to the sender that it is not congested by means of periodic RM cells. In response, the sender will increase the ACR of the VC to match the extra cell rate available at the receiver.

Suppose that the cell rate of the high priority background traffic is initially fixed at a value of R_{hi} cells/s and at time $t=-\tau$ suddenly decreases to R_{low} cell/s. The cell rate available for the ABR traffic then increases from a value of AR_{low} cell/s, equal to the LCR of the receiver minus R_{hi}, to AR_{hi} cells/s, equal to the LCR of the receiver minus R_{low}. Suppose also that at time $t=-\tau$ the buffer occupancy has just decreased to the value of the congestion threshold B_c. The subsequent behaviour of the ACR at the sender and the buffer occupancy B at the receiver over time are then illustrated in Figure 5.

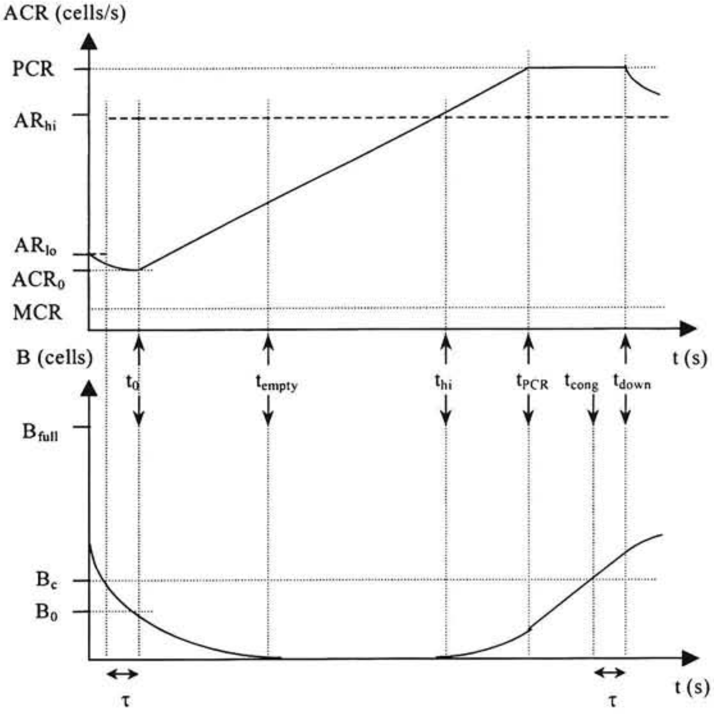


Figure 5 ACR and buffer occupancy as function of time for ramp-up scenario

By assumption, the receiver is not congested at time $t=-\tau$ and will indicate this to the sender by sending an unmarked RM cell at time $t=-\tau$. This RM cell will arrive at the sender at time $t_0=0$ due to the assumed transfer delay of τ seconds. The sender, who has meanwhile exponentially lowered the ACR to ACR_0, will then start linearly increasing the ACR and will continue to do this until receiving a marked RM cell from the receiver indicating congestion. This will occur at time t_{down}, exactly τ seconds after the buffer occupancy at the receiver exceeds B_c at time t_{cong}. After time t_{down} the sender will again start lowering the ACR.

Actually, Figure 5 only illustrates a typical example of the dynamic buffer and ACR behaviour. A number of implicit assumptions are hidden:

1. Firstly, it is assumed that the PCR is larger than the AR_{hi}. If this is not the case then the ACR simply increases to the PCR and the buffer empties itself, resulting in a stable situation as both the ACR and the buffer occupancy now remain constant over time. The times t_{hi}, t_{cong} and t_{down} are then undefined. As the resulting behaviour is somewhat trivial, the remaining analysis assumes that PCR > AR_{hi}.

2. Secondly, the ACR is assumed to reach the PCR at a time t_{PCR} which is earlier than time t_{down}. If this is not the case then the ACR will continue to increase until time t_{down} instead of being limited by the PCR. The time t_{PCR} is then undefined.
3. Thirdly, it is assumed that time t_{PCR} is smaller than time t_{cong}. If t_{PCR} is larger than t_{cong} but still smaller than t_{down} then this influences the buffer occupancy slightly.
4. Fourthly, it is assumed that the buffer empties itself completely before the buffer occupancy starts increasing again. This increase in occupancy will begin as soon as the ACR exceeds the AR_{hi} at time t_{hi}. It is possible however, that the buffer does not empty itself completely before this time. In this case t_{empty} is not defined.
5. Lastly, it is assumed that the buffer occupancy does not exceed the maximum buffer size of B_{fill}. If this occurs before time t_{down}, buffer overflow will occur resulting in cell loss.

Using the ACR update rules (3) and (4) , the CLR and efficiency over the time interval t_0 to t_{down} can be expressed as function of B_0, B_c, ACR_0, AR_{hi}, PCR, RIF, T_{RM} and τ. Both expressions are listed in Table 1 along with the expressions for the relevant parameters determining the behaviour illustrated in Figure 5. Exceptions 2, 3, 4 and 5, mentioned in the previous paragraph, are taken into account in this table. The notations ACR_{hi}, B_{hi}, ACR_{down} and B_{down} are introduced for the ACR and buffer occupancy B at times t_{hi} and t_{down} respectively.

Ramp-up
Given B_0, B_c, ACR_0, AR_{hi}, PCR, RIF, T_{RM}, τ then define:
$t_{hi} = T_{RM} \cdot \frac{AR_{hi} - ACR_0}{PCR \times RIF}$, $B_{hi} = \max\left(B_0 - \frac{T_{RM}}{2} \frac{(AR_{hi} - ACR_0)^2}{PCR \times RIF}, 0 \right)$, $t_{cong} = t_{hi} + \sqrt{\frac{2T_{RM} \times (B_c - B_{hi})}{PCR \times RIF}}$, $t_{PCR} = T_{RM} \frac{PCR - ACR_0}{PCR \times RIF}$.
If $t_{cong} + \tau \le t_{PCR}$ then define
$t_{down} = t_{cong} + \tau$, $B_{down} = B_c + \tau\left(ACR_0 - AR_{hi} + (\tau + 2t_{cong}) \frac{PCR \times RIF}{2T_{RM}} \right)$, $ACR_{down} = ACR_0 + (t_{cong} + \tau) \frac{PCR \times RIF}{T_{RM}}$,
Else if $t_{cong} < t_{PCR} < t_{cong} + \tau$ then define
$t_{down} = t_{cong} + \tau$, $B_{down} = \begin{array}{l} (ACR_0 - AR_{hi})(t_{PCR} - t_{cong}) + (t_{PCR}{}^2 - t_{cong}{}^2) \frac{PCR \times RIF}{2T_{RM}} \\ + B_c + (t_{cong} + \tau - t_{PCR})(PCR - AR_{hi}) \end{array}$, $ACR_{down} = PCR$,
Else define
$t_{down} = t_{PCR} + \frac{1}{PCR - AR_{hi}} \left(B_c - B_{hi} - \frac{PCR \times RIF (t_{PCR} - t_{hi})^2}{2T_{RM}} \right) + \tau$, $B_{down} = B_c + \tau(PCR - AR_{hi})$, $ACR_{down} = PCR$,
If $B_{hi} = 0$ then
$t_{empty} = t_{hi} - \sqrt{t_{hi}^2 - T_{RM} \frac{2B_0}{PCR \times RIF}}$, Efficiency $= \frac{t_{down} - (t_{hi} - t_{empty})}{t_{down}}$, # cells sent = $AR_{hi} \times t_{down} + B_{down} - \frac{T_{RM}}{2} \frac{(AR_{hi} - ACR_0)^2}{PCR \times RIF}$,
Else
Efficiency = 1, # cells sent = $AR_{hi} \times t_{down} + B_{down} - B_0$.
If $B_{down} > B_{fill}$ then
CLR = $\frac{B_{down} - B_{fill}}{\text{\# Cells sent}}$,
Else
CLR = 0.

Table 1 Performance of the ABR congestion control loop in the ramp-up scenario.

3.4 Ramp-down

This section analyses the exact opposite scenario to the ramp-up scenario analysed in the previous section. Now, instead of suddenly decreasing, the amount of high priority background traffic suddenly increases. As a result, the amount of cell rate available at the receiver for the ABR traffic suddenly decreases, leading to congestion. The receiver will indicate this congestion back to the sender by means of the periodic RM cells. In response, the sender will lower it's ACR to match the reduced cell rate available at the receiver. Again, both the behaviour and the performance of the ABR congestion loop are analysed.

Suppose that the cell rate of the high priority background traffic is initially fixed at a value of R_{lo} cells/s and at time $t=-\tau$ suddenly increases to R_{hi} cell/s. The cell rate available for the ABR traffic then decreases from a value of AR_{hi} cell/s, equal to the LCR of the receiver minus R_{lo}, to AR_{lo} cells/s, equal to the LCR of the receiver minus R_{hi}. Suppose also that at time $t=-\tau$ the buffer occupancy has just increased to the value of the congestion threshold B_c. The subsequent behaviour of the ACR at the sender and the buffer occupancy B at the receiver over time are then illustrated in Figure 6.

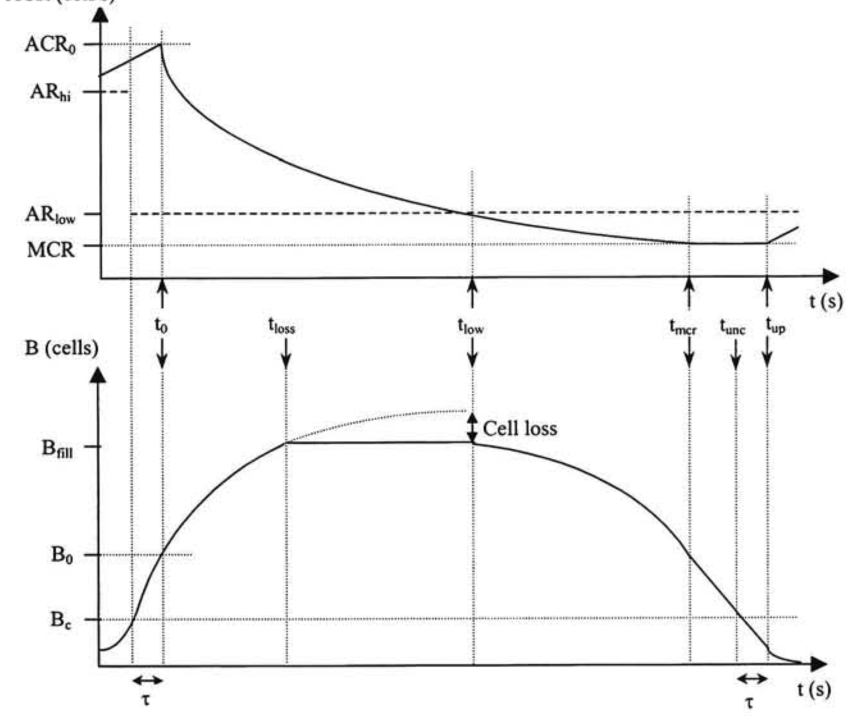


Figure 6 ACR and buffer occupancy as function of time for ramp-down scenario

By assumption, the receiver is congested at time $t=-\tau$ and will indicate this to the sender by sending a marked RM cell at time $t=-\tau$. This RM cell will arrive at the sender at time $t_0=0$ due to the assumed transfer delay of τ seconds. The sender, who has meanwhile increased the ACR to ACR_0, will then start lowering the ACR and will continue to do this until receiving an unmarked RM cell from the receiver. This will occur at time t_{up}, exactly τ seconds after the buffer occupancy at the receiver drops below B_c at time t_{unc}. After time t_{up} the sender will again start increasing the ACR.

Again, like in the ramp-up scenario, Figure 6 only illustrates a typical example of the dynamic buffer and ACR behaviour. A number of implicit assumptions are hidden:

1. Firstly, note that AR_{low} is higher than the MCR. This should always be the case as the MCR is the minimum cell rate guaranteed to any ABR VC.
2. Secondly, the ACR is assumed to reach the MCR at a time t_{MCR} prior to t_{up}. If this is not the case then the ACR will continue decreasing instead of being limited by the MCR. The time t_{MCR} is then undefined.
3. Thirdly, the time t_{MCR} is assumed to be prior to t_{unc}. If t_{MCR} is larger than t_{unc} but smaller than t_{up} then this influences the buffer occupancy slightly.
4. Fourthly, the buffer occupancy is assumed to reach the maximum buffer size of B_{fill} prior to t_{up}. If this is not the case then no cell loss occurs.
5. Finally, the buffer is not assumed to empty itself completely in the τ seconds between t_{unc} and t_{up}. If this does happen then this implies a loss of efficiency.

The CLR and efficiency over the time interval t_0 to t_{up} can be expressed as a function of B_0, B_c, ACR_0, AR_{low}, MCR, RDF, T_{RM} and τ. Table 2 lists both expressions, along with the expressions for the relevant parameters determining both the behaviour illustrated in Figure 6. This table also takes into account the exceptions 2, 3, 4 and 5 mentioned in the previous paragraph. The parameters ACR_{low}, B_{low}, ACR_{up} and B_{up} are introduced for the ACR and buffer occupancy at times t_{low} and t_{up} respectively.

Ramp-down
Given B_0, B_c, ACR_0, AR_{low}, MCR, RDF, T_{RM}, τ then define:
$t_{low} = \frac{T_{RM}}{\ln(1-RDF)} \ln(\frac{AR_{low}}{ACR_0})$, $t_{MCR} = \frac{T_{RM}}{\ln(1-RDF)} \ln(\frac{MCR}{ACR_0})$, $B_{low} = \min(B_0 + \frac{T_{RM}(AR_{low} - ACR_0)}{\ln(1-RDF)} - t_{low} AR_{low}, B_{fill})$, $t_{unc} = t_{low} + t_{num}$, where t$_{num}$ satisfies[11]: $\frac{B_{low} - B_c}{AR_{low}} = t_{num} + \frac{AR_{low}}{ACR_0} \frac{1-(1-RDF)^{\frac{t_{num}}{T_{RM}}}}{\ln(1-RDF)}$
If $t_{unc} + \tau \le t_{MCR}$ then define
$t_{up} = t_{unc} + \tau$, $ACR_{up} = ACR_0 (1-RDF)^{\frac{t_{up}}{T_{RM}}}$, $B_{up} = B_c - \tau \times AR_{low} - \frac{T_{RM}}{\ln(1-RDF)} ACR_0 (1-RDF)^{\frac{t_{unc}}{T_{RM}}} (1-(1-RDF)^{\frac{\tau}{T_{RM}}})$,
Else if $t_{unc} < t_{MCR} < t_{unc} + \tau$
$t_{up} = t_{unc} + \tau$, $ACR_{up} = MCR$, $B_{up} = B_c - \frac{T_{RM}}{\ln(1-RDF)} ACR_0 (1-RDF)^{\frac{t_{unc}}{T_{RM}}} (1-(1-RDF)^{\frac{t_{MCR}-t_{unc}}{T_{RM}}}) - (t_{MCR} - t_{unc}) AR_{low} - (t_{unc} + \tau - t_{MCR})(AR_{low} - MCR)$,
Else if $t_{unc} \ge t_{MCR}$
$t_{up} = t_{MCR} + \frac{B_{low} - B_c - AR_{low} \cdot (t_{MCR} - t_{low})}{AR_{low} - MCR} - \frac{T_{RM}}{\ln(1-RDF)} + \tau$, $ACR_{up} = MCR$, $B_{up} = B_c - \tau(AR_{low} - MCR)$.
If $B_{up} < 0$ then
Efficiency = $1 + \frac{B_{up}}{t_{up} AR_{low}}$, # cells sent[12] = $AR_{low} \times t_{up} + B_{up} - B_0$,
Else
Efficiency = 1, # cells sent[12] = $AR_{low} \times t_{up} + B_{up} - B_0$.
If $B_{low} = B_{fill}$ then
#cells lost = $B_0 + \frac{T_{RM}(AR_{low} - ACR_0)}{\ln(1-RDF)} - t_{low} AR_{low} - B_{fill}$, CLR = $\frac{\#\text{cells lost}}{\#\text{cells sent} + \#\text{cells lost}}$
Else
CLR = 0.

Table 2 Performance of the ABR congestion control loop in the ramp-down scenario.

[11] The time t$_{num}$ cannot be expressed as function of the other parameters and numerical methods are required to determine it's value.

[12] Ignoring any cells lost due to buffer overflow at the receiving switch

3.5 Steady state

This section analyses the behaviour and performance of the ABR congestion loop in the case that the amount of background traffic remains constant at a value of AR cells/s. In principle, the sender should adjust it's ACR to match this value after which a stable situation has been reached. However, the ACR update rules, see equations (3) and (4), only allow to the ACR to either increase or decrease and not to remain stable. Hence, the ACR value will fluctuate about the AR value except in the trivial cases that the AR is smaller or equal to the MCR or larger or equal to the PCR. For these two cases a stable situation is reached as soon as the ACR is equal to the MCR or PCR. Both of these trivial cases will not be considered here.

Figure 7 illustrates the behaviour of the ABR congestion control loop. These figures were made using the ABR congestion control loop simulator detailed in the next section, section 3.6. Table 3 lists the values of the parameters used. These values where chosen arbitrarily.

AR	265000 cells/s	N (# VCs)	23
B_{fill}	16000 cells	T_{RM}	30/N ms
B_c	2662 cells	PCR	N*300000 cells/s
RIF	1/32	MCR	N*4000 cells/s
RDF	1/8	τ	0 s

Table 3 Values used in ABR simulation tool to obtain Figure 7.

The top right figure in Figure 7 illustrates the value of the ACR at the sender as a function of time. Clearly, the ACR oscillates about the AR value which is indicated by the horizontal line. The bottom left figure illustrates the buffer occupancy B at the receiver as a function of time, the vertical line indicating the value of the congestion threshold B_c. The top left figure combines these two figures, displaying the ACR value at the sender as function of the buffer occupancy at the receiver. This figure illustrates most clearly the cyclic nature of the congestion loop, the ACR and B values always circling around the 'stable' state defined by an ACR equal to the AR and a B equal to B_c. The dot in the figure represents the stable state.

Figure 7 is suggestive in that it seems to indicate that the oscillation cycles become smaller and smaller as time progresses. This is not necessarily the case however and it is quite possible for the oscillation cycles to increase again after a period of decreasing.

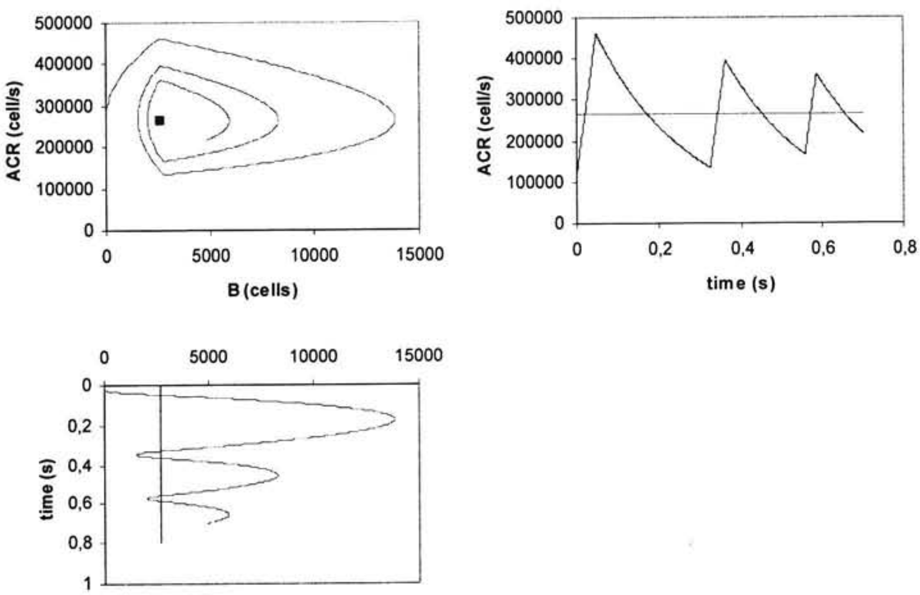


Figure 7 Example of the behaviour of the ACR congestion control loop in the steady state scenario

For the behaviour and performance of the ABR congestion control loop in the steady-state scenario no new analysis is needed. Instead, the analyses of the ramp-up and ramp-down scenarios suffice! This is seen most clearly in the top right figure in Figure 7 where each ACR oscillation consists of a ramp-up part followed by a ramp-down part. Setting the ACR_0 and B_0 of the ramp-up analysis equal to the resulting ACR_{up} and B_{up} of the previous ramp-down analysis, see Table 2, ties both these analyses together at one end. At the other end, both analyses can likewise be tied together by setting the ACR_0 and B_0 of the subsequent ramp-down equal to the resulting ACR_{down} and B_{down} of the current ramp-up analysis, see Table 1. This leads to an iterative, cyclic analysis that completely characterises the steady state behaviour.

In principle, a stable cycle, with the ACR and B values ending up at the same values to which they started, might be possible. Such a stable cycle would form a closed ellipse in the top left figure of Figure 7. To achieve this requires ACR_{down} to be equal to ACR_{up} as well as requiring B_{down} to be equal to $B_{up.}$ It can be shown that, unless the ACR is equal to at least either the MCR or the PCR at some time, or the buffer is at least either empty or full at some time then, this double set of equations has no solution and a stable cycle is not possible.

3.6 Fluctuating

This section illustrates the behaviour of the ABR congestion control loop in the presence of fluctuating background traffic. This traffic is characterised by two periods, both of fixed length, and two corresponding fixed cell rates. During each period the amount of background traffic that arrives is determined by the cell rate corresponding to that interval. The interaction between the ABR congestion loop and this background traffic is already so complex that an analytic study of the resulting behaviour and performance, such as the preceding ramp-up and ramp-down analyses, is no longer feasible. Instead, a simulator was constructed in Microsoft Excel. This allows the behaviour and performance of the control loop to be simulated, an example of which was already seen in the previous section for the case of traffic with equal cell rates in both periods, see Figure 7.

Figure 8 illustrates the behaviour of the ACR and the buffer occupancy for the same setup that was used in the previous section, see Table 3. The only difference is that the background traffic is not constant but varies with an active period of 1/10 seconds, during which cells are sent at a fixed rate of 200000 cells per second, and an less active period of 2/10 seconds, during which cells are sent at a fixed rate of 50000 cell/s. The resulting average cell rate available for the ABR traffic is again equal to 265000 cell/s, as in the setup used in the previous section. The horizontal lines in the top right figure in Figure 8 illustrate the behaviour of the background traffic as a function of time.

Clearly, the resulting behaviour is more complex than the behaviour in the steady state scenario. A closed cycle is shown in the top left figure of Figure 8, the buffer being equal to both B_{fill} and zero at some time in line with discussion on stable cycles in the previous section. However, such a cycle is no longer stable due to the fluctuating background traffic, as the figure illustrates. Buffer overflow also occurs, unlike in the steady state scenario, resulting in a CLR larger than zero. This indicates that the ABR congestion control loop is not functioning correctly as, in principle, no cell loss is allowed for ABR. To overcome this either the RDF can be increased or the RIF can be decreased, or both can be adapted simultaneously. Alternatively, the value of the T_{RM} can also be altered, changing the period between subsequent ACR updates. In general such alterations lead to a further decrease in the efficiency of the control loop which is already less than optimal given that the buffer is empty during a short time interval, see the bottom left figure in Figure 8. The tuning of the RIF, RDF and T_{RM} parameters is therefore a far from simple task in and is it difficult to both guarantee no cell loss and use the network efficiently.

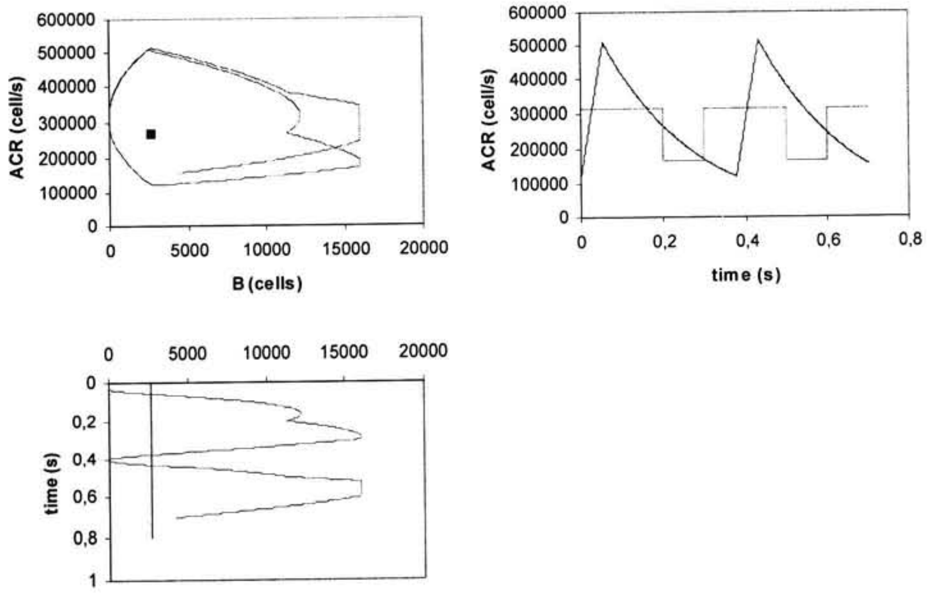


Figure 8 Example of the behaviour of the ACR congestion control loop in the deterministic on/off scenario

3.7 Fairness

This section discusses the issue of fairness amongst competing ABR VCs. Basically, each VC should get a fair share of the available cell rate in the network, with a minimum guaranteed cell rate equal to the MCR. What exactly constitutes a fair share is discussed, along with an analysis of how the RDF and RIF parameters of the ABR congestion control loop can be chosen in order to optimise fairness. This analysis is similar to that presented in [5].

The aggregate fluid flow model is no longer assumed in this section. Instead, VCs are treated on an individual basis and the ACR update rules are again given by equations (1) and (2). Sources generating ABR traffic are assumed to be greedy, implying that they always have data ready to send.

Two definitions of what constitutes a fair share are common, resulting in the following two definitions of fairness:

1. **Equal** fairness, where a fair share is defined as the total unreserved cell rate available for ABR traffic divided by the total number of VCs. The cell rate each VC gets in surplus of it's MCR is thus equal for all VCs, or in formula

$$ACR_n^i - MCR^i = ACR_n^j - MCR^j,$$

where the indices *i* and *j* refer to VC_i and VC_j respectively, and the index *n* refers to the nth ACR update, see equations (1) and (2).

2. **Weighted** fairness, where a fair share is defined as an MCR proportional portion of the total unreserved cell rate available for ABR traffic. The cell rate each VC gets in surplus of it's MCR is thus proportional to it's MCR, or in formula

$$\frac{ACR_n^i - MCR^i}{MCR^i} = \frac{ACR_n^j - MCR^j}{MCR^j}.$$

In the remainder of this section only weighted fairness is considered. One reason for this is that network operators will typically charge more for a VC with a higher MCR, implying that weighted fairness is preferable. Another reason is that the Ascend CBX 500, which is used as the default model of an ATM switch throughout this report, allows for the setting of MCR proportional per VC congestion thresholds on the ABR buffer. If the value of these thresholds is set proportional to the MCR then they can be used to stimulate weighted fairness, congestion being detected earlier for VCs with smaller MCR values. In the remainder of this section this is assumed to be the case.

Defining

$$\theta_n^i = \frac{ACR_n^i - MCR^i}{MCR^i},$$

allows for the introduction of

$$\nabla_n^{ij} = \theta_n^i - \theta_n^j$$

as a measure of the divergence from fairness between VC_i and VC_j. The smaller the divergence, the greater the fairness between the two VCs. The ACR update rules, see equations (1) and (2), imply that

$$\nabla_n^{ij} = \nabla_{n-1}^{ij} - RDF^i(1+\theta_{n-1}^i) + RDF^j(1+\theta_{n-1}^j)$$

if the ACR of both VC_i and VC_j is being decreased, and similarly imply that

$$\nabla_n^{ij} = \nabla_{n-1}^{ij} + \frac{RIF^i \times PCR^i}{MCR^i} - \frac{RIF^j \times PCR^j}{MCR^j}$$

if the ACR of both VCs is being increased.

It is also possible for one VC to increase it's ACR while the other VC decreases it. This is the case if one VC has exceeded it's per VC threshold while the other has not. As these thresholds are assumed to be set proportional to the MCR this case automatically entails a decrease in the divergence[13].

Choosing the RDF equal for all VCs,

[13] This is only true under the assumption that ABR traffic sources are greedy.

$$RDF^i = RDF^j \equiv RDF\ ,$$

and the RIF as follows

$$\frac{RIF^i \times PCR^i}{MCR^i} = \frac{RIF^j \times PCR^j}{MCR^j},$$

results in the following update rule for the divergence in case of ACR decrease

$$\nabla_n^{ij} = (1 - RDF) \times \nabla_{n-1}^{ij},$$

and

$$\nabla_n^{ij} - \nabla_{n-1}^{ij} = 0\,,$$

in the case of ACR increase.

Clearly, these update rules imply that successive ACR updates can only decrease or maintain the divergence, but never increase it. This is exactly what is desired.

Chapter 4 Performance of TC P/IP over a steady state ATM network

This analyses the expected performance of TCP/IP traffic over the ABR and GFR ATCs for the case of a steady state network. Here the term steady state denotes the assumption that the amount of background traffic in the ATM network is constant over time. More precisely, the cell rate available for an ABR or GFR VC is assumed to be constant at each ATM switch through which the VC passes. The switch with the minimum available constant cell rate (CCR) therefore determines the cell rate available for the VC and thus, ultimately, the goodput of the TCP/IP traffic. Here goodput is defined as the number of bytes of data received at the desired destination in complete IP packets per second and is the only performance measure considered in this. As each ATM cell has a payload of 48 bytes, the maximum goodput possible is given by μ_{max}=CCR*48*MSS/(MSS+48), where the CCR is assumed to be in cells/s and the last factor equal to MSS/(MSS+48) takes into account the TCP, IP and AAL5 headers added to a TCP segment of MSS bytes, see section 2.1. These headers do not count as data and therefore do not contribute to the goodput.

First, a brief summary of the TCP congestion control mechanisms are given. Then, the ABR ATC is discussed, specifically the issues relating to the two, independently operating ABR and TCP congestion control mechanisms. Finally, the expected TCP goodput is analysed for the GFR ATC. In this analysis a greedy TCP/IP source is assumed, implying that the source always has data ready to send. A source transferring a large file would be a typical example. The results of this analysis are used to set up the GFR simulations detailed in Chapter 6.

4.1 TCP congestion control

This subsection briefly summarizes the TCP congestion control mechanisms as they are described in [26]. First, the TCP sliding window mechanism is explained briefly, as this mechanism lies at the basis of how TCP regulates it's flow of segments. Then, the slow start algorithm is reviewed, followed by the congestion avoidance algorithm. Finally, the fast recovery and retransmit (FRR) algorithm is detailed briefly. In the remainder of this it is assumed that TCP implements the FRR algorithm although it should be noted that certain TCP versions do not do this. It should also be noted that work is currently underway on a new TCP version called TCP Vegas, see [6]. This version promises to be more suitable for use over ATM networks than most current TCP versions. However, as TCP Vegas is not widely applied as yet it will not be considered in this report.

4.1.1 Sliding window

This section reviews the sliding window mechanism which TCP uses to regulate the flow of segments. By means of this mechanism the destination informs the source of the maximum number of bytes it is willing to receive implying that the source can never send segments faster than the destination can handle.

When segments arrive at the desired destination in the order in which they were sent, they are acknowledged by the destination who sends small segments called ACKs back to the source. In these ACKs the destination not only indicates the cumulative number of bytes it has received in segments up to then, but also the maximum number of bytes it is willing to receive. The latter is also referred to as the *offered* window.

Back at the source, this value of the offered window size limits the number of bytes and thus segments the source can maximally send. This value can either be higher or lower than the number of segments a source can send at that moment. If it is zero then the source may not send any new segments for the moment. In this way the destination can regulate the speed at which the source transmits data.

In principle, each segment triggers it's own ACK, but it is also possible for multiple segments to be acknowledged by one and the same ACK. Such an ACK is referred to as a delayed ACK and is commonly sent if segments arrive at the destination out of order. If this happens the out of order segments are stored by the destination until the arrival of the missing segments which restore the correct ordering. These missing segments are then acknowledged along with the stored segments in one ACK.

4.1.2 Slow start

This section reviews the slow start algorithm which each TCP version is required to support. By means of this algorithm TCP increases the number of TCP segments a source may send exponentially. For a full discussion on this algorithm see [26] section 20.6.

The slow start algorithm uses a *congestion* window, not to be confused with the *offered* window discussed in the previous section, to limit the number of segments a source may send into the network. Initially, the congestion window is equal to one segment and a source may consequently send only one segment. Once this segment is acknowledged by an ACK, the source is in principle free to send a new segment. In addition, the congestion window is also increased by one segment implying that the source may, in principle, send a second segment as well. Whether or not the source may actually send these two new segments is subject to the size of the offered window, see the previous section. For each subsequent segment acknowledged, the congestion window is similarly increased by one segment and the source is again, in principle, allowed to send two new segments.

The traffic flow generated by the slow start algorithm is both bursty, new segments being sent in pairs interspersed by idle periods during which the source waits for subsequent ACKs, and continuously increasing in volume, due to the exponential growth of the congestion window. However, the congestion window cannot increase indefinitely as firstly, a maximum window size is defined, the value of which can be negotiated during the setup of the TCP session[14], and secondly, segment loss may occur in the network due to congestion. The congestion avoidance and FRR algorithms detailed in the next sections discuss how segment loss is handled.

4.1.3 Congestion avoidance

This subsection discusses the congestion avoidance algorithm which is designed to temper the exponential growth of the slow start algorithm, described in the previous section, and deal with the loss of segments in the network. The congestion avoidance and slow start algorithms are two independent algorithms, however, they are always implemented together and will be discussed as such. For a full discussion of the algorithm see [26] section 21.6.

The congestion avoidance algorithm places a threshold on the slow start congestion window. As long as the size of the congestion window is smaller than the value of this threshold, the size is increased exponentially by the slow start algorithm. However, as soon as the size exceeds the threshold, the size is increased linearly instead of exponentially. More specifically, if the size of the congestion window is equal to *cwnd* then the size is increased by 1/*cwnd* segments for each ACK received, instead of by one segment for each ACK received as in the slow start algorithm. Thus, each ACK received now triggers the sending of only one new segment[15], as opposed to two in the slow start algorithm. Once *cwnd* subsequent ACKs have been received the congestion window size has been increased by a total of one segment and the source may send two new segments[15]. The resulting traffic flow is therefore much less bursty than it is for the slow start algorithm, and the volume of traffic increases at a much slower rate.

In addition to tempering the growth of the congestion window, the congestion avoidance algorithm also deals with the retransmission of segments lost in the network. The loss of a segment is detected by the source if:

- one or more duplicate ACKs are received. These ACKs are triggered by the reception of out of order segments at the destination, for example segments sent after a particular segment which is lost in the network. These duplicate ACKs acknowledge the last segment[16] received in correct order and so indicate to the source which segment was lost. This case is dealt with by the FRR algorithm discussed in the next subsection.
- the TCP retransmission timer expires indicating that an ACK for a particular segment has not been received in time. The value of this timer is continually

14 Assuming that the TCP window scale option is implemented, see [26] section 24.4

15 If the offered window allows it, see 4.1.1.

16 Actually, an ACK indicates the number of the next byte of information it expects to receive, see [26] section 17.3 for further details.

modified in response to the time it takes for a segment to be acknowledged, the so called round trip time.

The latter case is dealt with by the congestion avoidance algorithm which retransmits the unacknowledged segment and initiates a new slow start. In addition, the value of the congestion window threshold is set equal to half the value of the congestion window at the moment the segment loss was detected. This results in an abrupt drop in the traffic flow generated by the source.

4.1.4 Fast recovery and retransmit

This subsection discusses the FRR algorithm which is designed to deal with the loss of segments in a network which is not overly congested. The algorithm is triggered by the reception of duplicate ACKs, which are taken to indicate segment loss, and results in the retransmission of the lost segment. However, unlike in the congestion avoidance algorithm, a new slow start is not initiated as, given the arrival of the duplicate ACKs, segments are apparently still arriving at the destination implying that the network is not overly congested. A new slow start would therefore restrict the flow of traffic to severely. However, the FRR algorithm fails if multiple segments are lost simultaneously in the network, or if the retransmitted segment is also lost. Both cases result in the timeout of the TCP retransmission timer and a subsequent new slow start. For a full discussion of the algorithm see [26] section 21.7.

The FRR algorithm works as follows:

- Upon reception of the third duplicate ACK (thus a total of four ACKs for the same segment), set the congestion window threshold to half of the current congestion window size.
- Retransmit the lost segment.
- Set the congestion window size equal to the window threshold plus three segments (for the three duplicate ACKs).
- For each subsequent duplicate ACK increase the window size by one segment and, if the window is sufficiently large, send a new segment not previously transmitted.
- Upon reception of a non-duplicate ACK (triggered by the arrival of the retransmitted segment at the destination), lower the congestion window to the threshold value plus one segment (due to the received ACK) and send two new segments.
- Proceed with the congestion avoidance algorithm.

Of course, the transmitting of segments during the FRR algorithm remains subject to the restrictions placed on the source by the sliding window mechanism, see section 4.1.1.

As an example of the FRR algorithm, suppose that the size of the congestion window was equal to W_{max} at the time the first duplicate ACK arrived. After reception of three duplicate ACKs, the window size is lowered to (W_{min}+3) and the lost segment is retransmitted. Here W_{min}=floor(W_{max}/2) and floor denotes rounding down to the nearest integer. At this point the source has sent a total of W_{max} segments into the network for which it has as yet not received an acknowledgement. Thus, before the source is allowed to send new segments into the network, an additional (W_{max}-W_{min}-3) duplicate ACKs are required in order to increase the window size to W_{max} again. The source can then send an additional (W_{min}-1) new segments into the network prior to the arrival of the first non-duplicate ACK triggered by the arrival of the retransmitted segment at the destination. This ACK acknowledges not only the retransmitted segment but also all the subsequent (W_{max}-1) segments sent prior to the retransmission of the lost segment. The size of the congestion window is now set equal to (W_{min}+1) allowing two new segments to be sent. The congestion phase is now reentered, ending the FRR algorithm. In conclusion, the FRR algorithm results in the window size being decreased from W_{max} to (W_{min}+1), and (W_{min}+1) new segments being sent into the network.

4.2 ABR

In this section the expected goodput of TCP/IP traffic over an ABR VC is discussed for a steady state network. If the ABR congestion control mechanism, see Chapter 3, functions correctly then the expected goodput should be equal to μ_{max}, the maximum value possible. The reason for this is that the ABR congestion control mechanism will eventually stabilise the allowed cell rate (ACR) of the TCP/IP source at the CCR, the minimum cell rate available in the steady state network. All the cells sent by the source in compliance with the ACR should arrive safely at the destination, resulting a goodput equal to the maximum possible.

However, the TCP congestion control mechanisms can have a detrimental effect on the goodput. For example, the TCP retransmission timer can expire due to the delayed arrival of a segment, triggering a new slow start. As a result the traffic volume will drop drastically and the source will not be able to fully utilise the available ACR, implying a drop in goodput. This is hard to avoid as ABR provides no guarantees on the maximum transfer delay a cell, and thus segment, experiences. A similar drop in goodput can occur if the ABR congestion loop malfunctions and cell loss occurs, possibly also resulting in a TCP slow start.

4.3 GFR

This section analyses the expected goodput of TCP/IP traffic over a GFR VC for the case of a steady state network. Unlike for ABR, cell loss is allowed to occur for GFR implying that the TCP congestion control mechanisms are of much greater influence on the goodput achieved by a TCP/IP source. First, the steady state network model upon which the subsequent analysis is based is discussed more fully. Then, the TCP traffic cycle resulting from the FRR algorithm and the subsequent congestion avoidance algorithm is analysed. The slow start algorithm does not affect this TCP traffic cycle as it will be shown that in the steady state network model the FRR algorithm never fails. Finally, an analysis of the expected goodput is given.

4.3.1 Steady state model

This subsection discusses the steady state network model upon which the subsequent analysis is based. The model is a simplification of a real network as it models an entire network by only one switch with a constant cell rate equal to CCR and a constant buffer equal to B, see Figure 9. In a real network, a GFR VC will pass through multiple switches and the amount of cell rate and buffer space available at each switch will fluctuate. The model used and the subsequent analysis performed are similar to those detailed in [19].

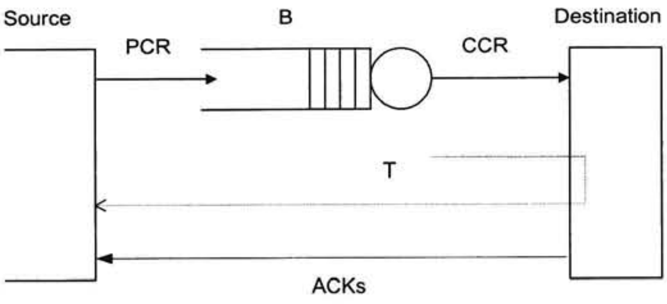


Figure 9 Steady state model of network

A TCP/IP source is directly attached to the switch by a GFR VC, see Figure 9, and sends segments to the switch at a cell rate equal to the PCR, with PCR>CCR. All segments are assumed to be of equal size and result in MFS cells. This implies a TCP MSS equal to 48*(MFS-1) due to the 48 byte TCP/IP header overhead and the 48 byte payload of the ATM cell, see section 2.1. The time that elapses between the sending of the segment by the switch and the arrival of the corresponding ACK back at the source, is assumed to be constant and equal to T seconds. Here $T=\tau+1/\mu$, where τ is the propagation delay from source to destination and back, and $1/\mu$ = MFS/CCR and denotes the time required by the switch to send one entire segment. In a real network the value of T will fluctuate due to the queueing delay experienced by both the segment and it's corresponding ACK as they are transferred between source and destination. It is implicitly assumed that the destination acknowledges each segment separately[17]. Often, TCP versions acknowledge segments in pairs, a reduction in ACK traffic which is especially handy during the slow start phase.

[17] It is also assumed that each segment is acknowledged immediately. Commonly, the operating system on which TCP runs generates new ACKs using a timer granularity in the order of 100 ms. Such details are ignored in this report.

However, such a delayed ACK triggers the sending of more than two new segments at the source and results in a more bursty traffic pattern than in the case of non-delayed ACKs.

4.3.2 TCP traffic cycle

This subsection analyses the cyclic TCP traffic pattern resulting from the repeated FRR and congestion avoidance algorithms. Initially, the source will use the slow start algorithm to increase the congestion window, eventually switching over to the congestion avoidance algorithm once the congestion window threshold is exceeded. This initial slow start will be ignored in the subsequent analysis which assumes that the source starts off in the congestion avoidance phase. It will be shown that a new slow start is never triggered, as in the steady state network only one segment is ever lost at one time and the FRR algorithm can handle it's retransmission successfully. This analysis also assumes that CCR < 2×PCR and deals with entire segments only, ignoring the fact that each segment is chopped into cells by the ATM network. Furthermore, the size of the offered window, see section 4.1.1, is assumed to always be larger than the size of the congestion window.

Table 4 illustrates an example of a possible TCP traffic cycle and this example will be used in the remainder of this subsection to explain the various aspects of such a cycle in general. The source is assumed to start off in the congestion avoidance phase with a congestion window equal to five segments. All five segments are assumed to have already been sent and the ACK acknowledging the first of these is assumed to arrive at time t=0, see the first and second columns of row one in Table 4. In response, the source sends out a new, sixth segment to the buffer, see the third column of row one in Table 4. As the source sends the segment at the PCR, which is larger than the CCR with which the buffer is emptied, the buffer temporarily fills up with this segment, see column four of row one in Table 4.

The ACK for the second segment sent arrives exactly 1/μ seconds later, see row two of Table 4. The reason for this is that the buffer empties segments at a rate of one per 1/μ seconds, implying that the interval between the first and second segments arriving at the destination is also equal to 1/μ seconds. As a result, the interval between both ACKs being sent and arriving back at the source is thus also equal to 1/μ. The arrival of the second ACK triggers the sending of a new, seventh segment. The buffer, in the meantime, has emptied itself of the sixth segment and is thus only filled by the seventh segment.

This process continues for the third and fourth ACKs, triggering the sending of two new segments, numbered eight and nine. Upon reception of fifth ACK, however, the congestion window increases by one segment, from five to six segments, see the fifth column of row 5 in Table 4. As a result, instead of sending only one new segment, the source now sends two new segments back to back, numbered ten and eleven. This results in a temporary buffer filling of these two segments, ignoring ATM cell aspects. Note that the source has now sent six segments into the network, numbered six through eleven, and the congestion window is also six segments, as it should be. The sixth segment can thus be considered as the first segment of the new congestion window, the seventh as the second etc.

Exactly 1/μ seconds later, the ACK for segment six arrives and triggers the sending of a new segment numbered twelve. The reason for this is that the round trip time in this example was set equal to μT=5. As a result, there is no break between the window increase, triggered by the fifth ACK, and the arrival of the ACK belonging to the sixth segment, the first segment of the new congestion window. Therefore, when segment twelve arrives at the buffer, segment eleven has not yet been emptied. As a result, the buffer occupancy rises to two segments again.

As the subsequent ACKs arrive one every 1/μ seconds, the buffer is never emptied again. Instead, the buffer is always filled by at least one segment and at most by two segments. In fact, the buffer occupancy continues to rise upon each subsequent window increase. For example, upon arrival of the eleventh ACK the congestion window is again increased by one segment and two new segments are again sent back to back. As a result, the buffer occupancy increases, being always filled by at least two segments and at most by three segments.

In principle, the buffer occupancy could continue to increase in this way after each window increase. However, in this example the maximum buffer size was set equal to four segments implying that as soon the window increase from seven to eight segments buffer overflow occurs and segment 26 is discarded. This is detected by the source after the reception of the first duplicate ACK at time t=5T. No new segment is now sent, nor in response to the arrival of the second duplicate ACK, and as a result the buffer occupancy drops by two segments. However, upon reception of the third duplicate ACK, triggered by the arrival of segment 30 at the destination, segment 26 is retransmitted by the FRR algorithm.

The FRR algorithm also halves the congestion window and subsequently increases it by three segments for the three duplicate ACKs, resulting in seven segments. As the source has now sent eight segments into the network unacknowledged, numbered 26 through 33, it may not send any additional new segments until the congestion window increases to nine. In the meantime the buffer will continue to empty, in this example emptying out completely. The FRR algorithm increases the congestion window by one for every duplicate ACK received, implying that the window will increase to nine upon reception of the fifth duplicate ACK. The new segment sent in response will be numbered one in order to illustrate the cyclic nature of the traffic pattern. The sixth and seventh duplicate ACKs also trigger new segments, numbered two and three.

There is no eighth duplicate ACK however. Instead a delayed ACK arrives, triggered by the arrival of the retransmitted segment 26 at the destination. This ACK acknowledges segments 26 through 33. The FRR algorithm now sets the congestion window back to four segments plus one for the delayed ACK, resulting in a window of five segments. As only three segments have been sent so far, numbered 1 to 3, the source sends two new additional segments, numbered 4 and 5. As a result, the buffer fills temporarily with these two segments.

There is now a break equal to 1/µ seconds during which no ACKs are received by the source. The reason for this is the round trip time, see the previous discussion, which is equal to µT=5 in this example. As a result, the ACK for segment one does not arrive immediately after the sending of segment five but 1/µ seconds later. During this break the buffer can empty itself of segment four, leaving only one segment in the buffer. The cycle is now complete, the end situation being identical to the begin situation.

Time of reception ACK	ACK received for segment	Send segment(s)	Segments in buffer	Window size
0	1	6	6	5
$1/\mu$	2	7	7	.
$2/\mu$	3	8	8	.
$3/\mu$	4	9	9	.
$4/\mu$	5	10,11	10,11	6
T	6	12	11,12	.
$T+1/\mu$	7	13	12,13	.
$T+2/\mu$	8	14	13,14	.
$T+3/\mu$	9	15	14,15	.
$T+4/\mu$	10	16	15,16	.
2T	11	17,18	16,17,18	7
$2T+1/\mu$	12	19	17,18,19	.
$2T+2/\mu$	13	20	18,19,20	.
$2T+3/\mu$	14	21	19,20,21	.
$2T+4/\mu$	15	22	20,21,22	.
3T	16	23	21,22,23	.
$3T+1/\mu$	17	24	22,23,24	.
$3T+2/\mu$	18	25, 26	23,24,25	8 (discard 26)
$3T+3/\mu$	19	27	24,25,27	.
$3T+4/\mu$	20	28	25,27,28	.
4T	21	29	27,28,29	.
$4T+1/\mu$	22	30	28,29,30	.
$4T+2/\mu$	23	31	29,30,31	.
$4T+3/\mu$	24	32	30,31,32	.
$4T+4/\mu$	25	33	31,32,33	.
5T	25 (triggered by 27)	-	32,33	.
$5T+1/\mu$	25 (28)	-	33	.
$5T+2/\mu$ (start FRR)	25 (29)	26 (retransmit)	26	4+3=7
$5T+3/\mu$	25 (30)	-	-	8
$5T+4/\mu$	25 (31)	1	1	9
6T	25 (32)	2	2	10
$6T+1/\mu$	25 (33)	3	3	11
$6T+2/\mu$ (end FRR)	33 (26)	4,5	4,5	4+1=5
$6T+3/\mu$	-	-	5	.
Restart cycle				

Table 4 Example of TCP traffic cycle for W_{max}=8, μT=5 and B/MFS=3.

The above analysis focused on the example in Table 4. The gist of the analysis remains valid for the general case however. The source starts off in the congestion avoidance phase, gradually increasing the congestion window. As soon as this window increases beyond W_0=floor[μT] there is no longer a break in the flow of ACKs and the buffer occupancy will rise with each subsequent window increase. Here floor denotes rounding down to the nearest integer. As a result, buffer overflow will occur as soon the window size reaches W_{max}=(W_0+floor[B/MFS]). After the FRR algorithm has successfully retransmitted the lost segment the congestion avoidance algorithm is restarted with a window equal to W_{min}=floor[W_{max}/2]+1, completing the cycle. Note that it is possible that W_{min} is larger than W_0 and the source never experiences a break in the flow of ACKs. The above cyclic analyses remains valid however.

The above analysis implicitly assumes that CCR < 2×PCR. If this is not the case then the resulting traffic pattern will deviate from the one described above. However, typically the PCR will be equal to the link cell rate whereas the CCR will be in the order of the MCR and is generally much smaller. The assumption that CCR < 2×PCR is therefore not a restrictive one. An additional simplification in the above analysis is that it is based on entire TCP segments and ignores ATM cell aspects. The most important of these aspects is that sending 2×MFS cells (2 segments) to the buffer results in a maximal buffer filling of 2×χ×MFS cells instead of 2×MFS cells, where χ is equal to (1-CCR/PCR) and denotes the part of the segment already transferred by the switch before the arrival of the final cell of that segment. However, this does not change the fact that per congestion window increase the maximal buffer filling increases by MFS. The effect these aspects have on the analysis are marginal and will be ignored. Another assumption the analysis makes is that the value of the TCP retransmission timer is large enough to avoid any unnecessary retransmissions, especially during the FRR algorithm. If this is not the case then a new slow start may be initiated and the above cyclic analysis breaks down. This is not an unreasonable assumption as the value of the retransmission timer is updated continuously in accordance with the measured packet round trip time. This value increases slowly during the cycle due to the filling of the buffer and, as a result, should be large enough to avoid unnecessary retransmissions during the traffic cycle.

4.3.3 Goodput

This subsection calculates the goodput achieved by the traffic cycle analysed in the previous section. The results are used in Chapter 6 to determine the value of the MBS traffic parameter used in the GFR simulations. The relationships between the CCR and MCR, and buffer B and MBS are also briefly discussed.

The total number of segments sent during the congestion avoidance phase and the subsequent FRR algorithm is given by:

$$N = \sum_{n=0}^{W_{max}-W_{min}} (W_{min} + n) + (W_{max} - 1) = \frac{1}{2}(W_{max} - W_{min} + 1)(W_{max} + W_{min}) + W_{max} - 1 .$$

To see this, note that all the segments belonging to windows W_{min} through to W_{max}-1 are delivered without problem. As soon W=W_{max} only W_{max}-1 segments are delivered as the last segment sent is lost due to buffer overflow. However, this lost segment is retransmitted and the ACKs of these W_{max}-1 segments still result in W_{max}-1 additional new segments being sent. This leads to the above expression.

The total time required by the switch to send these segments is given by:

$$\frac{N}{\mu}+t_1+t_2 .$$

Here N/μ denotes the time required to sent N segments at rate μ and

$$t_1 = \max\left[\frac{1}{\mu}\left(W_{\max} - W_{\min} + 1 - floor\left[\frac{B}{MFS}\right]\right),0\right],$$

represents the amount of time during the FRR algorithm that the switch buffer is empty. To see how this occurs note that when the window size is equal to W_{max}, the switch buffer is continuously filled by floor(B/MFS) segments. Upon reception of the first two duplicate ACKs however, the source does not send any new segments, implying a decrease in switch buffer by two segments. After retransmission of the lost segment, upon reception of the third duplicate ACK, no new segments are sent until an additional (W_{max}-W_{min}-1) duplicate ACKs are received. This implies that the buffer can empty a further maximum of (W_{max}-W_{min}-1) segments, resulting in a total of (W_{max}-W_{min}+1) segments that can possibly be emptied from the buffer. If there are not that many segments in the buffer to begin with then this implies that cell rate available at the switch is left unused, implying a loss in goodput.

Similarly, if $W_{min} \leq W_0$ then

$$t_2 = \max\left[T\left(W_0 + 1 - W_{\min}\right) - \frac{1}{\mu}\sum_{n=0}^{W_0 - W_{\min}}(W_{\min} + n),0\right],$$

represents the time the output link of the switch is idle as a result of a congestion window being smaller than or equal to W_0 and the flow of ACKs not being continuous. Each break in the flow of ACKs implies that the switch buffer empties and cell rate available at the switch is left unused. Subtracting the time the switch spent in sending segments from the total time the source required to increase the window size to W_0 gives the above expression for t_2.

The goodput is now given by

$$\frac{N}{\frac{N}{\mu}+t_1+t_2}\frac{MFS-1}{MFS},$$

where the (MFS-1)/MFS factor takes into account the TCP/IP header overhead.

The idle periods t_1 and t_2 are either both unequal to zero, or both equal to zero. To see this suppose that W_{max} is even. Both t_1 and t_2 can then be shown to be equal to zero as soon as floor[B/MFS]/floor[μT] $\geq$ 1 and unequal to zero if this is not the case. Similarly, if W_{max} is uneven then both can be shown to be equal zero if floor[B/MFS]/(floor[μT]+1) $\geq$ 1 and unequal if this is not the case. As a rough estimate of whether the maximal goodput of μ_{max} is achieved, the normalised buffer β=B/(CCR T) can be used. If $\beta \geq 1$ then a maximal goodput of μ_{max} is achieved, otherwise the goodput achieved is lower.

Chapter 5 A Comparison of tw o CAC algorithms

This compares two different connection admission control (CAC) algorithms. Such an algorithm is responsible for limiting the number of VCs simultaneously admitted onto the network. If this number is not limited and all VCs are simply admitted, then the network may become overloaded, implying that it can no longer provide QoS guarantees in terms of a maximum CLR or transfer delay. Without such an algorithm a GFR VC, for example, would not be able to provide the required QoS guarantees, implying in turn a loss of guarantees for the TCP/IP traffic being transferred over that VC.

There are numerous different CAC algorithms, only two of which are compared in this. The reason for comparing these specific two is that both are currently implemented in the ATM network of KPN. One is implemented in the Ascend CBX 500 ATM switch, whereas the other is implemented in a network planning tool called the ATM end to end manager (AEEM). If, when and how discrepancies occur in the number of VCs admitted by either of these two algorithms is therefore of particular interest.

First, section 5.1 discusses the GAN CAC algorithm detailed in [14], followed by section 5.2 which discusses the EMW algorithm detailed in [9]. Then, section 5.3 compares both algorithms with respect to the number of VCs they admit onto the network, one algorithm being said to be more conservative than the other if it admits fewer VCs. Finally, in section 5.4 conclusions are drawn about which algorithm is preferable and will be used to set up the simulations in Chapter 6.

5.1 GAN

This section briefly discusses the CAC algorithm detailed more fully in [14]. The letters GAN refer to the first letters of the surnames of the three authors of this article. The algorithm is actually based on two approximations, the first of which shall be referred to as GAN fluid, the second as GAN Gaussian. Each approximation will be discussed in a separate subsection, after which the CAC algorithm resulting from these two approximations is discussed.

5.1.1 GAN fluid approximation

This subsection discusses the first of the two approximations on which the GAN CAC is based. This approximation is valid in the case of large switch buffers. The notion of an effective bandwidth is also introduced. The word bandwidth is part of the standard terminology whenever CAC algorithms are concerned. In the context of this report bandwidth is understood to be equivalent to cell rate.

Consider a source alternating between an 'on' state and an 'off' state. During the 'on' state the source sends cells continuously at a fixed cell rate *R*. During the 'off' state the source is idle. The length of the 'on' and 'off' periods are independently, but not necessarily identically, exponentially distributed. The mean length of the 'on' period will be denoted by *b* and the mean cell rate by *m*. The mean length of the 'off' period can be determined from these two.

Given an ATM switch with a bandwidth $C<R$ and a buffer with size *B* being fed by one such source, then the time-independent, equilibrium probability of buffer overflow (ε) can be shown to be equal to

$$\varepsilon = \rho\chi \exp\left(\frac{1-\phi}{\phi}\frac{1-\chi\rho}{(1-\rho)(1-\chi)}\right), \tag{5}$$

where the following dimensionless ratios have been introduced for later convenience

$$\rho = \frac{m}{R}, \quad \phi = \frac{1}{1+\dfrac{B}{bC}}, \quad \chi = \frac{R}{C}. \tag{6}$$

The ratios ρ and ϕ are restricted to the interval (0,1), that is assuming that $0<m<R$, $B>0$, $C>0$ and $b>0$, whereas χ lies somewhere in the interval $(0,\infty)$. However, for a switch with bandwidth $C\geq R$, the buffer is emptied at least as fast as it is filled and thus $\varepsilon=0$. This implies that equation(5) is only valid for $\chi>1$, ε being equal to zero for $\chi\leq 1$. Note that the probability of buffer overflow is simply equal to the cell loss ratio (CLR), or in formula

$$\varepsilon = CLR.$$

Equation(5) is furthermore subject to the stability restriction $m/C<1$, ensuring that the rate at which the buffer is emptied(*C*) is greater than the average rate at which it is filled(*m*). If this condition is violated then the system of switch and source is unstable in that, as time goes to infinity, the buffer will always overflow, implying an $\varepsilon=1$.

The purpose of a CAC method is to limit the CLR to a specified value, typically in the order of 10^{-6} or 10^{-7}. Equation(5) can be used in this respect to determine whether a certain switch can or cannot admit a particular source, depending on whether the resulting CLR is smaller or larger than the specified value. Inverting equation(5) would allow the minimum switch capacity *C* required to limit the CLR to the specified value to be determined for a given source. However, this inversion cannot be accomplished analytically and numerical techniques would therefore be required.

To avoid this, article [14] makes the approximation $\rho\chi=m/C=1$. As the stability condition requires that $m/C<1$, this approximation allows for an explicit upper bound on the minimum required bandwidth to be obtained,

$$c_f = \frac{\alpha b(1-\rho)R - B + \sqrt{(\alpha b(1-\rho)R - B)^2 + 4B\alpha b\rho(1-\rho)R}}{2\alpha b(1-\rho)}, \tag{7}$$

where

$$\alpha = \ln\left(\frac{1}{\varepsilon}\right).$$

Instead of specifying the minimum required switch bandwidth, equation(7) can also be seen as defining an 'effective' bandwidth for a given source, the value of which is larger or equal to the mean cell rate *m* of the source and smaller or equal to the maximum cell rate *R*:

$$m \le c_f \le R.$$

To see that this is true note that c_f satisfies the following limits,

$$\lim_{\varepsilon\to 0} c_f = R, \quad \lim_{\varepsilon\to 1} c_f = m, \tag{8}$$

and increases with increasing α (and thus decreasing ε). The last limit ($\varepsilon \to 1$) corresponds with the stability condition $m/c_f<1$. This condition is violated for $c_f=m$, resulting in instability and an $\varepsilon=1$. Note that, in the limit of continuous traffic ($b \to \infty$ and m=R), c_f becomes *R* as expected.

5.1.1.1 Multiple sources

The above analysis deals with the case of one source only. For the case of *N* sources the situation becomes more complex, especially if the sources are not all identical with respect to their *m*, *R* and *b* parameters. The expression for the buffer overflow probability, analogue to equation for ε above, can now only be given explicitly for the case of *N* identical sources, see, and even then cannot be explicitly inverted. In general therefore, numerical approximations are required to obtain an expression for the minimum required switch bandwidth for a specified CLR and a given set of sources.

To avoid this, article [14] argues that an upper bound on the total required bandwidth for any *N* VCs is given simply by the sum of their individual effective bandwidths c_f. This approximation, which will henceforth be referred to as the GAN fluid approximation, is obtained in the limit of an infinite buffer and by ignoring the effects of statistical multiplexing. It states that, as long as the switch reserves c_f of its total capacity for each source, the CLR for all sources combined will be maximally equal to the specified ε.

5.1.1.2 GAN fluid CAC

In conclusion, the GAN fluid CAC works as follows: for each new source determine its effective bandwidth and add this value to the sum of the effective bandwidths of the sources already admitted onto the network. If the resulting total bandwidth is greater than the switch bandwidth, the new source should be rejected, if it is less, the source may be admitted.

The maximum number of *identical* sources the switch can admit simultaneously is therefore equal to

$$n_f = \frac{C}{c_f}. \tag{9}$$

This expression will be useful later on when the GAN fluid CAC is compared to the EMW lossless CAC.

5.1.2 GAN Gaussian approximation

This subsection discusses the second of the two approximations on which the GAN CAC is based. This approximation is based on statistical multiplexing and the ignoring of the buffer space available at the switches, and uses the Gaussian distribution to limit the CLR. The notion of an effective bandwidth is now no longer valid and an equivalent capacity is defined instead.

A different approximation that can be made in the case of *N* exponentially distributed 'on-off' sources (see previous section) is based on ignoring the switch buffer ($B \approx 0$) and using the central limit theorem. In the limit of infinite *N*, the central limit theorem states that the distribution of the total cell rate of all *N* sources will approach a Gaussian distribution with mean cell rate *m* and standard deviation σ given by,

$$m = \sum_{i=1}^{N} m_i, \quad \sigma^2 = \sum_{i=1}^{N} \sigma_i^2,$$

where m_i and $\sigma_i^2 = m_i(R_i - m_i)$ are the mean cell rate and standard deviation of the cell rate belonging to source i.

The probability (ε) that the total cell rate will exceed the capacity of the switch (*C)* can now be calculated from the Gaussian distribution. As the buffer is ignored, this probability is equal to the cell loss ratio (CLR) and we again have ε=CLR. Inverting the Gaussian distribution would thus allow the minimum required total switch capacity (C_g) to be obtained as function of the allowed CLR. In general however, this is not possible analytically and numerical techniques are required. To avoid this, article [14] uses an approximation valid for small ε which results in,

$$C_g = m + \beta\sigma, \tag{10}$$

where

$$\beta = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}.$$

This approximation will henceforth be referred to as the GAN Gaussian approximation. Note that it restricts $\varepsilon \leq 1/\sqrt{(2\pi)}$; an artefact of the fact that the approximation is accurate for small ε only. As the CLR is typically of the order 10^{-6} or 10^{-7} this should not be a problem however and the approximation will be quite accurate.

Instead of regarding C_g as the minimum required switch capacity, equation(10) can also be seen as defining an 'equivalent' capacity for a given mix of *N* sources, the value of which satisfies $C_g \in [m,\infty)$. To see this, note that C_g satisfies the following limits

$$\lim_{\varepsilon \to 0} C_g = \infty, \quad \lim_{\varepsilon \to \frac{1}{\sqrt{2\pi}}} C_g = m, \tag{11}$$

and increases with increasing β (and thus decreasing ε). The first limit (ε→0) is due to the assumption of a Gaussian distributed total cell rate. This implies that a CLR=0 is never possible as there is always a chance that the total cell rate of the *N* sources will exceed any finite switch capacity C_g, irrespective of how large C_g is.

Unlike in the case of the fluid approximation, it is now no longer possible to define an effective bandwidth per individual source. The value of the equivalent capacity C_g depends intrinsically on the mix of sources present. To see this more clearly let us restrict ourselves to the case of two different types of sources; one with parameters m_1, b_1, and R_1, and the other with m_2, b_2, and R_2. For the case of only one source of type 1 the equivalent capacity is given by

$$C_{g1} = m_1 + \beta\sqrt{m_1(R_1 - m_1)},$$

while similarly, for the case of only one source of type 2,

$$C_{g2} = m_2 + \beta\sqrt{m_2(R_2 - m_2)}.$$

For the case of both sources being active simultaneously however, the equivalent capacity is given by

$$C_{g12} = m_1 + m_2 + \beta\sqrt{m_1(R_1 - m_1) + m_2(R_2 - m_2)},$$

from which it is easily seen that

$$C_{g12} < C_{g1} + C_{g2}.$$

Thus, the concept of an 'effective' bandwidth does not apply to the case of the GAN Gaussian approximation as it does to the case of GAN fluid.

5.1.2.1 GAN Gaussian CAC

The GAN Gaussian CAC method now works as follows: for each new source recalculate the equivalent capacity, taking into account all the sources already admitted onto the network. If the resulting new equivalent capacity is lower than the switch capacity then admit the source, if it is larger then reject the source.

The maximum number of *identical* sources a switch can admit simultaneously is therefore given by

$$n_g = \frac{1}{\chi\rho}\left(1 - \frac{\chi\beta^2(1-\rho)}{2}\left[\sqrt{1 + \frac{4}{\beta^2\chi(1-\rho)}} - 1\right]\right), \tag{12}$$

an expression which can be obtained by inverting equation(10) for the case of only one type of source. This expression will be useful later when comparing the GAN Gaussian CAC with the EMW lossless CAC.

5.1.3 CAC based on both GAN approximations

This subsection details how the GAN CAC algorithm uses the two approximations discussed in the previous sections to limit the number of VCs admitted onto the network. In section 5.3 the resulting CAC algorithm will be compared to the CAC algorithm which is discussed in the next section.

The GAN CAC algorithm, discussed in article [14], assigns a total bandwidth to an arbitrary mix of N sources, also called an equivalent capacity (C_{GAN}), equal to the minimum of the above two approximations,

$$C_{GAN} = \min\left(C_g , \sum_{i=1}^{N} (c_f)_i \right), \tag{13}$$

where $(c_f)_i$ denotes the GAN fluid effective bandwidth for the i-th source. A new source is now accepted by a particular switch as long as it does not increase C_{GAN} beyond C, the capacity of the switch. Note that the maximum number of *identical* sources a switch can therefore admit simultaneously is given by

$$n_{GAN} = \max(n_f , n_g)\, . \tag{14}$$

Whether the GAN fluid or the GAN Gaussian approximation determines the equivalent capacity C_{GAN} depends on the mix of sources and the types of sources present. In article [14] it is argued that GAN Gaussian is the determining approximation for sources with a relatively large b and small ρ. In this case the GAN fluid approximation will admit fewer sources onto the network and is therefore said to be more conservative than GAN Gaussian. This result is in correspondence with the assumption of a large (infinite) buffer B for the fluid approximation, and with the ignoring of the buffer for the Gaussian approximation. The idea is that for large bursts of cells(b) at a high cell rate(R), the buffer will be filled quickly and its effect can thus be safely ignored ($B \approx 0$). This is in line with the GAN Gaussian approximation. In the opposite case, namely sources with small bursts and a mean cell rate(m) close to the peak cell rate(R), sources are 'on' most of the time implying that the buffer is important and the effects of statistical multiplexing can be ignored. This is line with the GAN fluid approximation.

5.2 EMW lossless

This section briefly discusses the EMW lossless CAC algorithm detailed more fully in [9]. Like the letters GAN, the letters EMW refer to the first letters of the surnames of the three authors of this article. The term "lossless" is used to distinguish between the two CAC algorithms detailed in the article. Only the algorithm ensuring a CLR equal to zero is considered here, hence the name.

The article takes an alternative approach to calculating effective bandwidths. A static, 'worst case' traffic pattern is now assumed where the length of the on and off periods are fixed instead of exponentially distributed. A source is now assumed to send at PCR during the on period, the length of which is characterised by a fixed burst size of MBS cells. The fixed length of the off period is determined by the mean cell rate which is given by SCR (or MCR in the case of GFR). The most logical relation between these parameters and those used by GAN in the previous sections is given by

$$m = SCR, \quad b = \frac{MBS}{PCR}, \quad R = PCR. \tag{15}$$

Note that three nearly equivalent sets of parameters are now in use, namely *(m,b,R)*, (ρ,ϕ,χ) and (SCR, MBS, PCR). Actually, the parameters (ρ,ϕ,χ) incorporate the additional switch parameters *B* and *C* as well, highlighting the fact that there are actually only three important parameters instead of five (i.e. *m,b,R,B* and *C*). All three sets will be used throughout this article, depending on which is more convenient at the time.

For a given ratio of *B/C*, a so called effective bandwidth ensuring a CLR equal to zero can now be calculated for a given source,

$$e_0 = \begin{cases} R\phi, & \phi \geq \rho, \\ m, & \phi \leq \rho. \end{cases} \tag{16}$$

This effective bandwidth, which lies somewhere between *m* and *R* ($e_0 \in [m,R)$), will be henceforth referred to as EMW lossless. The equivalent capacity for *N* sources is now given by the sum of the individual effective bandwidths,

$$C_{EMW} = \sum_{n=1}^{N} (e_0)_i , \tag{17}$$

where $(e_0)_i$ refers to the effective bandwidth of the i-th source. This implies that if EMW lossless is used as a CAC algorithm then a new VC is only admitted as long as the current equivalent capacity plus the e_0 of the new VC remains under the capacity *C* of the switch. For a particular source, the maximum number of *identical* sources that a switch with capacity *C* can then handle and still ensure a CLR=0 is given by

$$n_0 = \frac{C}{e_0}. \tag{18}$$

5.3 Comparing GAN and EMW lossless

In this section the two different CAC algorithms described in the previous two sections are compared with regard to the number of VCs each admits, one algorithm being called more conservative than the other if it admits fewer VCs. On first glance, one might assume that as EMW lossless guarantees no cell loss (CLR=0), it should always be more conservative than GAN which allows a cell loss ratio larger than zero (CLR=ε>0). After all, the EWM lossless algorithm calculates a lower bound on the number of VCs that may be safely admitted onto the network. If GAN were to admit fewer VCs than this lower bound then it would only imply an inefficient use of network resources. This section shows that EMW lossless is often less conservative the GAN, furthermore determining for which types of VCs, and mixes of these VCs, this is the case.

To see that GAN can indeed be less conservative than EMW lossless it is sufficient to note the following:

- The exponential distribution underlying GAN fluid implies that as the CLR$\rightarrow$0, $c_f \rightarrow R$ (see equation(8)).
- The Gaussian distribution underlying GAN Gaussian implies that as the CLR$\rightarrow$0, $C_g \rightarrow \infty$ (see equation(11)).
- For EMW lossless we have that $e_0<R$ (unless SCR=PCR but this violates the assumption that $\rho<1$).

The first two points imply that as the CLR decreases $C_{GAN} \rightarrow \Sigma R_i$. This, combined with the third point, implies that for sufficiently small ε, $C_{GAN}>C_{EMW}$. When this happens, GAN requires a switch to reserve a larger amount of bandwidth than EMW lossless for the same set of sources. This is equivalent with GAN admitting fewer VCs than EMW onto the network and therefore, with GAN being more conservative.

In the case of only one type of source the critical value of ε below which GAN fluid admits fewer VCs than EMW lossless can be determined by inverting the inequality $n_0>n_f$. This results in

$$\varepsilon_f = \exp\left(-\frac{1}{\phi(1-\rho)\chi}\left(1-\frac{\rho}{\phi}\right)\right), \tag{19}$$

and a similar expression for GAN Gaussian can be obtained by inverting $n_0>n_g$:

$$\varepsilon_g = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\phi}{2\chi\rho(1-\rho)}\left(1-\frac{\rho}{\chi}\right)^2\right), \tag{20}$$

It should be noted that both equations were obtained under the assumption that $\phi \geq \rho$, implying that $e_0>m$. If this is not the case then the critical values of ε simply become $\varepsilon_f=1$ and $\varepsilon_g=1/\sqrt{(2\pi)}$, in accordance with the limits in equations(8) and (11), and GAN is always more conservative than EMW lossless for a typical value of ε (say 10^{-6}).

For a given source and a specified value of the CLR=ε it is now possible to determine whether or not EMW lossless is more conservative than GAN by comparing the ε_f and ε_g values with the specified ε. If $\varepsilon<\varepsilon_f$ then GAN fluid is more conservative than EMW lossless with respect to this source, whereas if $\varepsilon>\varepsilon_f$ the opposite is true. For GAN Gaussian the same holds with ε_f replaced by ε_g. Note that the minimum taken over the Gaussian and fluid approximations in equation(13) implies that GAN (both fluid and Gaussian) admits $n_{GAN}=\max(n_g, n_f)$ of these sources onto the network. Thus, GAN is more conservative than EMW lossless for a given source only if both ε_f and ε_g are smaller than ε. If either ε_f or ε_g is larger than ε then the opposite is true.

Instead of having to recalculate ε_f and ε_g for each new source and having to compare these with the specified ε, it is more useful to be able to specify beforehand which types of sources will satisfy $\varepsilon_f<\varepsilon$ and/or $\varepsilon_g<\varepsilon$ for a specified ε. Section 5.3.1 does just this and sections 5.3.2 and 5.3.3 extend the obtained results to the case of two or more source types. Finally, section 5.3.4 describes a crude but simple test that can be used to determine when EMW lossless is more or less conservative than GAN.

5.3.1 One type of source

This subsection formulates a theorem stating when the EMW lossless CAC algorithm is more conservative than the GAN algorithm. The theorem is applicable to the case of one source type only but is used as the basis of the comparisons in the next two sections which deal with the case of two or more source types.

Equations(19) and (20) can be inverted to obtain the maximum allowed ρ as a function of ϕ and χ for specified ε. This results in the following equation,

$$\rho_f = \phi \frac{1-\alpha\chi\phi}{1-\alpha\chi\phi^2}, \tag{21}$$

in the case of GAN fluid and,

$$\rho_g = \phi \frac{2+\chi\beta^2 - \beta\sqrt{\chi^2\beta^2 + 4\chi(1-\phi)}}{2(1+\phi\chi\beta^2)}, \tag{22}$$

in the case of GAN Gaussian. The following theorem is now valid

Theorem 1 For a specified value of the cell loss ratio, $\varepsilon \in (0, 1/\sqrt{(2\pi)})$, and for a given source with parameters (ρ, ϕ, χ), EMW lossless is more conservative than GAN (with respect to this source) if and only if $\rho < \max(\rho_f, \rho_g)$.

Proof Assume first that EMW lossless is more conservative than GAN and will thus admit fewer sources. This implies that $n_{GAN} = \max(n_g, n_f)$ is larger than n_0, which in turn implies that at least one of either c_f or C_g is smaller than e_0. The latter is only possible if $\phi > \rho$ as otherwise $e_0 = m$ and both c_f and C_g are larger than m for $\varepsilon \in (0, 1/\sqrt{(2\pi)})$, see equations(8) and(11). This means that equations(19) and (20) are valid and that $\varepsilon > \varepsilon_f$ and/or $\varepsilon > \varepsilon_g$.

- Assume that $\varepsilon > \varepsilon_f$. Inverting the inequality $\varepsilon > \varepsilon_f$ results in the constraints $\rho < \rho_f$ for $\phi^2 < 1/(\alpha\chi)$ and $\rho > \rho_f$ for $\phi^2 \geq 1/(\alpha\chi)$. For $1/(\alpha\chi) > 1$, only the first constraint is applicable as $\phi \in (0,1)$. For $1/(\alpha\chi) \leq 1$ again only the first constraint is applicable as for $\phi^2 \geq 1/(\alpha\chi)$ we have $\rho_f \geq 1$, and we already know that $\rho \in (0,1)$. Thus we are left with the constraint $\rho < \rho_f$, valid for $\phi^2 < 1/(\alpha\chi)$. Actually, in order to avoid $\rho_f \leq 0$, ϕ must also satisfy the sharper constraint $\phi < 1/(\alpha\chi)$. Note that as we assume that for the given source $\varepsilon > \varepsilon_f$, ϕ must automatically satisfy $\phi < 1/(\alpha\chi)$; otherwise we have a contradiction.
- Assume that $\varepsilon > \varepsilon_g$. Inverting the inequality $\varepsilon > \varepsilon_g$ results in a second order equation with two roots ρ_+ and $\rho_- = \rho_g$, where

$$\rho_\pm = \phi \frac{2+\chi\beta^2 \pm \beta\sqrt{\chi^2\beta^2 + 4\chi(1-\phi)}}{2(1+\phi\chi\beta^2)}.$$

  The constraints on ρ now become $\rho < \rho_g$ or $\rho > \rho_+$, but it can be shown that $\rho_+ \geq \phi$ implying that the constraint $\rho > \rho_+$ is inconsistent with the assumption that $\rho < \phi$. Therefore ρ must satisfy $\rho < \rho_g$.

Combining the above two cases, we now have that ρ must satisfy $\rho<\max(\rho_f, \rho_g)$, proving the theorem one way.

Assume, on the other hand, that $\rho<\max(\rho_f, \rho_g)$. It is easy to prove that both $\rho_f<\phi$ and $\rho_g<\phi$, thus implying that $\rho<\phi$ as well. Therefore, equations(19) and (20) are again valid. By construction, substituting ρ back into these equations will results in ε satisfying either one or both of the following constraints, $\varepsilon>\varepsilon_f$ or $\varepsilon>\varepsilon_g$. This implies that EMW lossless is more conservative than GAN, proving the above theorem the other way.

Figure 10 and Figure 11 illustrate the restrictions placed on ρ as a function of ϕ for given χ and α. The area under the GAN fluid curve denotes the values of ρ for which EMW lossless is more conservative than GAN fluid, the same holding for the area under the GAN Gaussian curve. Clearly two distinct cases can be distinguished which can be shown to correspond with the cases $\alpha\chi>1$ and $\alpha\chi<1$.

- For $\alpha\chi<1$ it can be proven that $\rho_f>\rho_g$, implying that GAN fluid is always less conservative than GAN Gaussian. The GAN fluid restriction therefore determines whether GAN is more or less conservative than EMW lossless; GAN being less conservative if $\rho<\rho_f$.
- For $\alpha\chi>1$ however, ρ_f becomes smaller than ρ_g as ϕ increases. Now GAN fluid is the determining restriction for small values of ϕ, whereas GAN Gaussian becomes the determining restriction as ϕ increases. Note that for $\phi>1/(\alpha\chi)$ GAN fluid becomes more conservative than EMW lossless regardless of the value of ρ (see also the above proof).

This behaviour is in line with the assumptions underlying the two GAN approximations; GAN Gaussian becoming the determining approximation for large b (and thus large ϕ) and small ρ (and thus large *R* which in turn implies large χ).

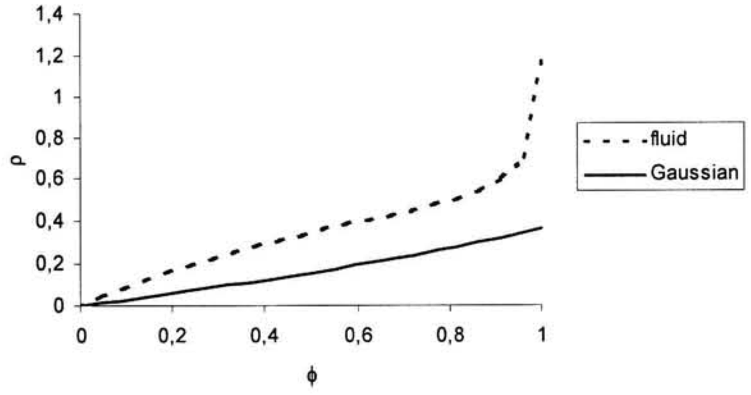


Figure 10 ρ_f and ρ_g as function of ϕ for $\alpha\chi=0,94$

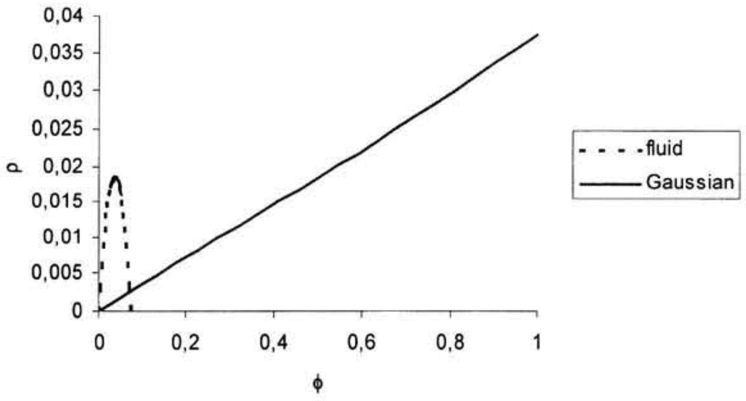


Figure 11 ρ_f and ρ_g as function of ϕ for $\alpha\chi$=13,8

5.3.2 Two types of sources

This section extends the results obtained for the case of one source type in the previous section to the case of an arbitrary number of sources of two different types mixed together. This is less straightforward than that it may seem due to the fact that the GAN Gaussian CAC depends intrinsically on the mix of the two types and not only on the two types individually (see section 5.1.2 for more details).

Defining source type 1 by the parameters (ρ_1, χ_1, ϕ_1), equations(18), (9), (14), (21) and (22) then determine n_{10}, n_{1f}, n_{1g}, ρ_{1f} and ρ_{1g} respectively. Similarly, defining source type 2 by the parameters (ρ_2, χ_2, ϕ_2) results in n_{20}, n_{2f}, n_{2g}, ρ_{2f} and ρ_{2g}. For a given number of sources of type 1 (n_1) it is now possible to determine the maximum number of sources a switch with finite capacity *C* and buffer *B* can admit of type 2 (n_2). The resulting boundary in two dimensional (n_1,n_2) space will be called the CAC boundary as it determines the boundary of admissible mixes. Only mixes lying within the CAC boundary can be safely admitted without fear of increasing the CLR beyond the specified value.

For the EMW lossless CAC the CAC boundary is linear and is given by

$$n_2 = -\frac{n_{20}}{n_{10}} n_1 + n_{20}, \qquad (23)$$

as in this case the CAC is based on the adding up of the individual effective bandwidths of both source types. For the GAN fluid CAC, which is also based on the summation of effective bandwidths, the resulting linear CAC boundary is

$$n_2 = -\frac{n_{2f}}{n_{1f}} n_1 + n_{2f}. \qquad (24)$$

For GAN Gaussian however, the CAC boundary is no longer linear but convex and can be obtained by inverting equation(10) to obtain n_2 as a function of n_1. The resulting CAC boundary is given by

$$n_2 = -\frac{\chi_1\rho_1}{\chi_2\rho_2}n_1 + \frac{1}{2\chi_2\rho_2}\left(2+\beta^2 d_2\right) - \frac{\beta}{2\chi_2\rho_2}\sqrt{n_1 4\chi_1\rho_1\left(d_1-d_2\right)+d_2(\beta^2 d_2+4)}, \quad (25)$$

where we have introduced

$$d_1 = \chi_1(1-\rho_1), \quad d_2 = \chi_2(1-\rho_2),$$

for convenience. Note that for GAN the CAC boundary is now defined by the maximum of the GAN fluid and GAN Gaussian CAC boundaries or, more precisely, for a given n_1 sources of type 1, the GAN CAC can admit the maximum of equation(24) and (25) n_2 sources of type 2.

Four distinct situations are now possible:

1. If $n_{10}>\max(n_{1f}, n_{1g})$ and $n_{20}>\max(n_{2f}, n_{2g})$ then EMW lossless is less conservative than GAN, irrespective of the mix of sources. This is because both the GAN fluid and GAN Gaussian CAC boundaries now lie wholly within the EMW lossless CAC boundary, see for example Figure 12. This implies that it is possible for EMW lossless to admit a particular mix of the two sources which GAN does not. This case is equivalent with the case $\rho_1>\max(\rho_f, \rho_g)$ and $\rho_2>\max(\rho_f, \rho_g)$.

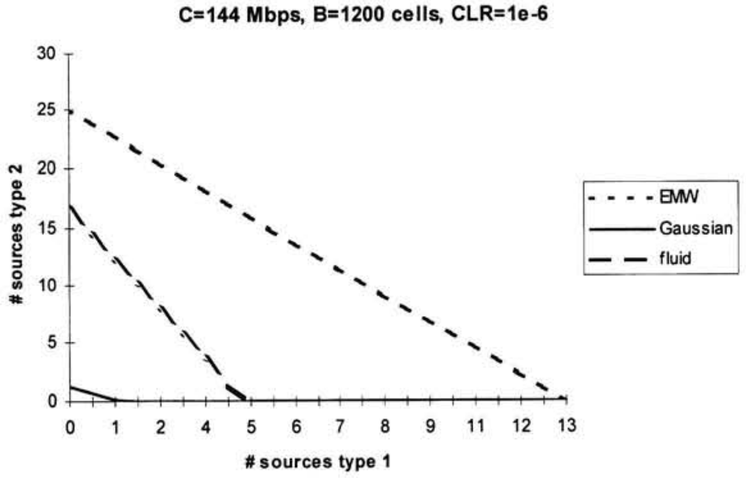


Figure 12 Example of CAC boundaries for $n_{10}>\max(n_{1f}, n_{1g})$ and $n_{20}>\max(n_{2f}, n_{2g})$.

2. If $n_{1f}>n_{10}$ and $n_{2f}>n_{20}$ then EMW lossless is more conservative than GAN, irrespective of the number of sources of type 1 or 2 mixed together. This is because the EMW lossless CAC boundary now lies entirely within the GAN fluid CAC boundary, implying that any mix of sources that EMW lossless admits, GAN

will certainly admit. Figure 13 illustrates this clearly. This case can be shown to be equivalent with the case $\rho_1<\rho_{1f}$ and $\rho_2<\rho_{2f.}$

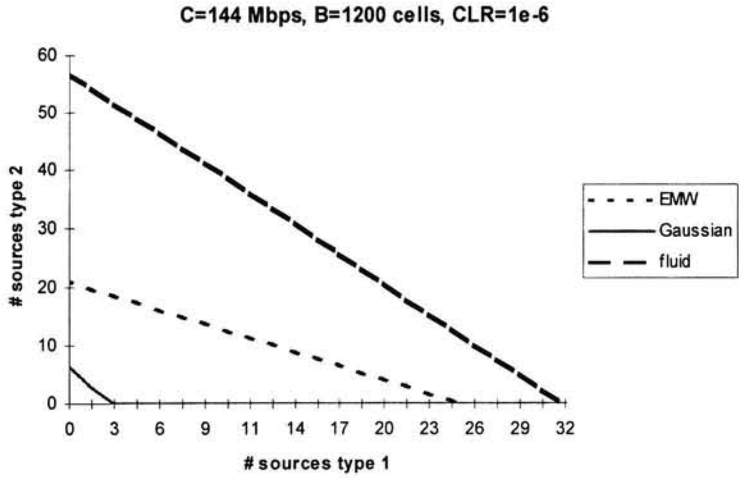


Figure 13 CAC boundaries for $n_{1f}>n_{10}$ and $n_{2f}>n_{20.}$

3. If $n_{1f}<n_{10}<n_{1g}$ and $n_{20}<n_{2g}$ (or vice versa) then EMW lossless can be either more or less conservative than GAN depending on the mix of sources. On first glance this might seem strange as one might expect the EMW lossless CAC boundary to lie wholly within the GAN Gaussian CAC boundary. However, this is not always the case due to the convexity of the GAN Gaussian CAC boundary, as Figure 14 clearly illustrates. This case is equivalent with $\rho_{1f}<\rho_1<\rho_{1g}$ and $\rho_2<\rho_{2g}$.

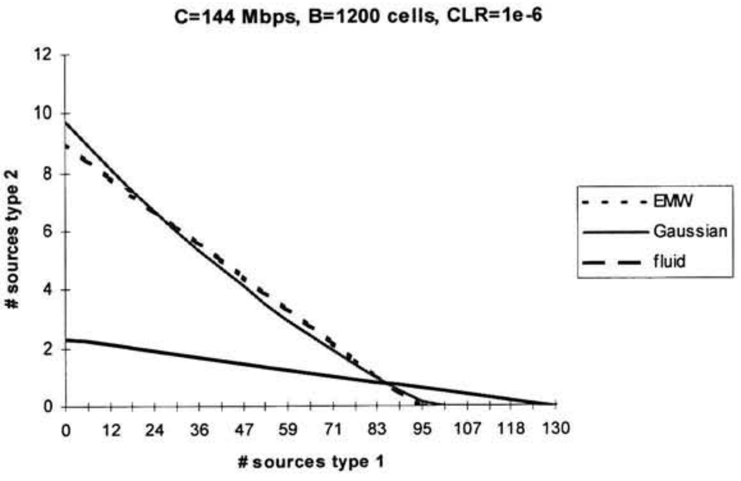


Figure 14 CAC boundaries for $n_{1f}<n_{10}<n_{1g}$ and $n_{20}<n_{2g}$.

4. Finally, if $n_{1g}<n_{10}<n_{1f}$ and $n_{2f}<n_{20}<n_{2g}$ (or vice versa) then EMW lossless can be either more or less conservative than GAN, depending on the mix of sources. Figure 15 illustrates this clearly. This case is equivalent with $\rho_{1g}<\rho_1<\rho_{1f}$ and $\rho_{2f}<\rho_2<\rho_{2g}$.

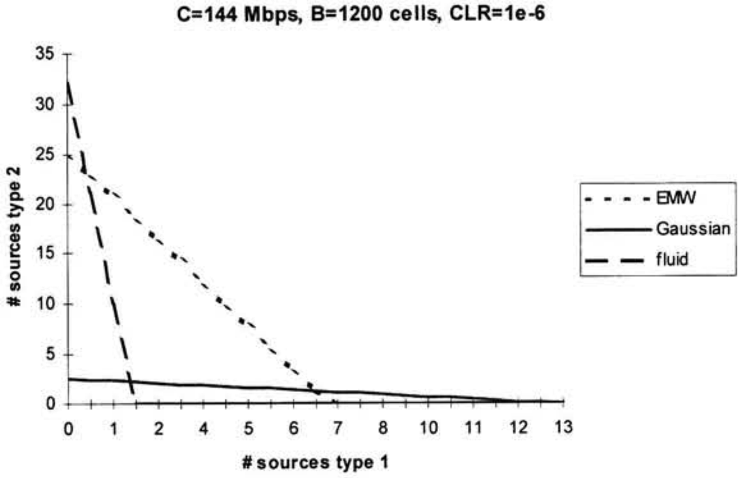


Figure 15 CAC boundaries for $n_{1f}<n_{10}<n_{1g}$ and $n_{2g}<n_{20}<n_{2f}$.

5.3.3 The general case

This section extends the results obtained for the case of two source types in the previous section to the case of an arbitrary number of source types. This extension is now relatively straightforward.

If each source type individually satisfies the GAN fluid restriction $\rho<\rho_f$ then GAN will be less conservative than EMW lossless for any mix of sources. If, on the other hand, each source satisfies $\rho>\max(\rho_f, \rho_g)$ then EMW lossless will always be less conservative GAN. If neither of these two cases hold then EMW lossless can be either more or less conservative than GAN depending on the mix of sources. There is now no easy way to determine beforehand which of the two CACs will be more conservative.

5.3.4 Simplified restrictions

In this section a crude but simple test is devised to determine whether GAN is more conservative than EMW lossless for a given set of sources. The test is based on a simplification of the GAN fluid and GAN Gaussian restrictions (ρ_f and ρ_g) and is only applicable for the case $\alpha\chi>1$.

From equations (6) and (15) we see that, for a given source, $\alpha\chi>1$ requires PCR to be larger than a critical value PCR_{crit} given by

$$PCR_{crit} = \frac{C}{\alpha}. \tag{26}$$

Figure 16 shows PCR_{crit} as a function of the CLR (and thus α) for a given switch capacity *C* chosen equal to 144 Mbps, slightly lower than the standard STM-1 rate of 155 Mbps as switches often reserve some capacity as backup. In this case, for a typical CLR=10^{-6}, $PCR_{crit}\approx 11$(Mbps) which is a relatively small value and a source with a PCR>PCR_{crit} should therefore not be an uncommon occurrence. From now on we will assume that all sources under consideration satisfy $\alpha\chi>1$.

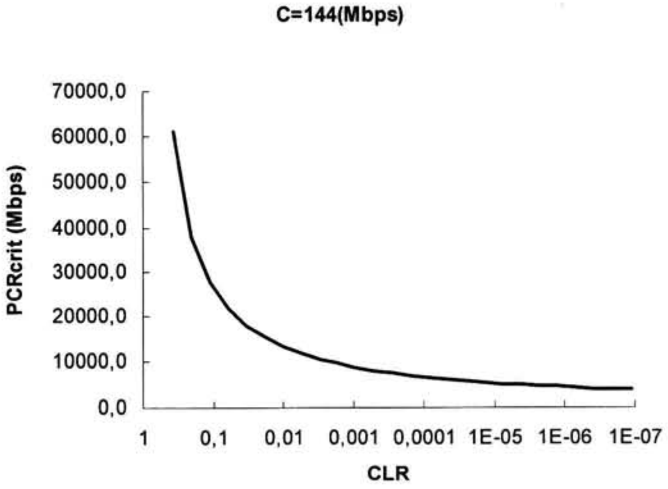


Figure 16 PCR_{crit} as function of the CLR.

As was already noted in section 5.3.1, if $\phi>1/(\alpha\chi)$ then GAN fluid is always more conservative than EMW lossless (see also Figure 11). Using equations (6) and (15) we see that this corresponds with a maximum burst size of MBS_{max} given by

$$MBS_{\max} = \frac{B}{C}\frac{PCR}{\alpha\chi - 1}. \tag{27}$$

If a source has a $MBS>MBS_{max}$ then this implies that for this source GAN fluid is more conservative than EMW lossless. Figure 17 illustrates the MBS_{max} as a function of the PCR for B=1200(cells), a value chosen somewhat arbitrarily, C=144(Mbps) and a CLR=10^{-6}. Clearly, MBS_{max} is not so large as to make it unlikely that a source will have a larger MBS.

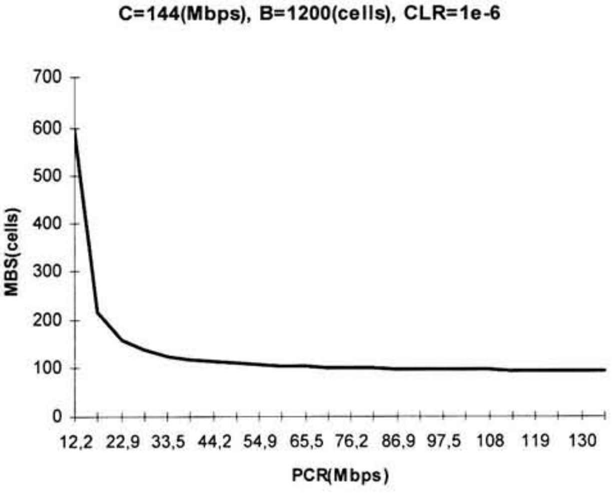


Figure 17 MBS_{max} as function of the PCR.

For $\alpha\chi>1$ the GAN fluid restriction ρ_f has a maximum value as can be clearly seen from Figure 11. Using equations (6) and (15) we see that this implies a maximum value of the SCR which can be shown to be given by

$$SCR_{\max,f} = \frac{PCR}{2}\left(1-\sqrt{1-\frac{1}{\alpha\chi}}\right). \qquad (28)$$

If a source has a $SCR>SCR_{max,f}$ then it is impossible for this source to satisfy $\rho<\rho_f$ for any value of ϕ. Therefore, GAN fluid is now more conservative than EMW lossless.

Similarly, the GAN Gaussian restriction ρ_g has a maximum value too, implying a maximum SCR which can be shown to be given by

$$SCR_{\max,g} = \frac{PCR}{1+\chi\beta^2}. \qquad (29)$$

For sources with an $SCR>SCR_{max,g}$, GAN Gaussian is always more conservative than EMW lossless, irrespective of the value of ϕ.

Figure 18 illustrates both $SCR_{max,f}$ and $SCR_{max,g}$ as function of the PCR for the same *B, C* and CLR as used in the previous figure. Clearly, both $SCR_{max,f}$ and $SCR_{max,g}$ are not so large as to make it unlikely that a source will have a larger SCR.

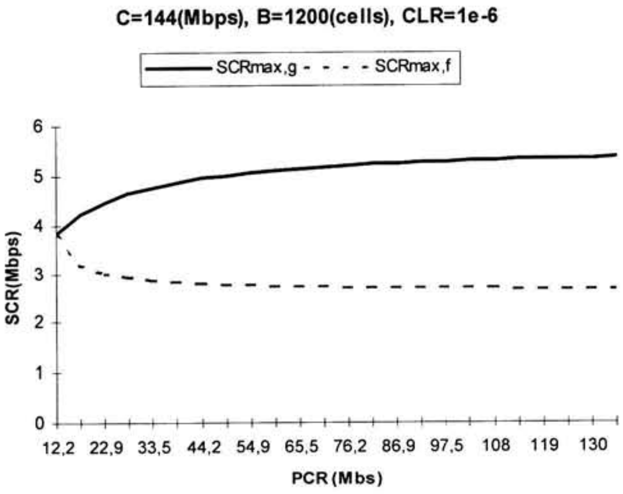


Figure 18 $SCR_{max,f}$ and $SCR_{max,g}$ as function of the PCR.

Equations (27), (28) and (29) now constitute a set of simplified restrictions which can be used to determine on a crude scale whether GAN is certainly more conservative than EMW lossless, or there is a chance that GAN may be more conservative. The idea is simply that if for a given source $MBS<MBS_{max}$ then $SCR_{max,f}$ determines a lower bound on the SCR above which GAN is certainly more conservative than EMW lossless. $SCR_{max,g}$ similarly forms a lower bound if the $MBS>MBS_{max}$. Figure 19 illustrates this concept. Thus, if, for a given mix of sources, all sources individually lie above the simplified lower SCR bounds, then GAN is more conservative than EMW lossless. Note that the simplified restrictions are crude in that it is still possible for GAN to be more conservative the EMW lossless if some or all of the sources lie below the simplified lower SCR bounds.

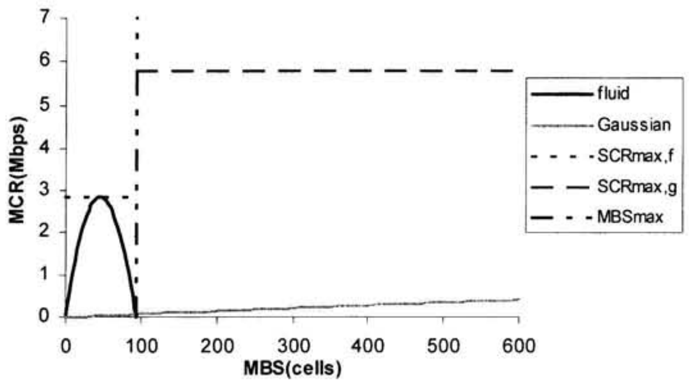


Figure 19 Simplified restrictions for determining whether GAN is more conservative than EMW lossless in (MBS,SCR) space, with C=PCR=144(Mbps) and CLR=10^{-6}.

5.4 Conclusions

This section summarises the results of the previous sections, stating in which cases the EWM lossless CAC algorithm is more conservative than the GAN CAC algorithm, and in which cases it is less conservative. Based on these conclusions, one of the two CAC algorithms is selected as preferable and used in Chapter 6 to set up the GFR simulations.

For a given mix of source types, a sufficient condition for ensuring that EMW lossless is more conservative than GAN is that each source type individually satisfies $\rho<\rho_f$, with ρ_f given by equation(21). A sufficient condition for ensuring that EMW lossless is less conservative than GAN is that each source type individually satisfies $\rho>\max(\rho_f, \rho_g)$, where ρ_g is given by equation(22). The latter condition can be crudely simplified for sources with a PCR>PCR_{crit}, where PCR_{crit} is given by equation(26). For such sources GAN is more conservative if:

- the MBS<MBS_{max} and the SCR>$SCR_{max,f}$, where MBS_{max} is given by equation(27) and $SCR_{max,f}$ by equation(28),
- or, the MBS>MBS_{max} and the SCR>$SCR_{max,g}$, where $SCR_{max,g}$ is given by equation(29).

If neither of these conditions are satisfied then EMW lossless can either be more or less conservative than GAN, depending on the mix of source types. No conclusions can now be drawn on the basis of the individual source types alone, it is the actual mix of source types which determines which of the two CAC methods is more conservative.

Based on these results, the EMW lossless CAC is considered preferable to the GAN CAC algorithm. The reason for this is that although the EMW lossless CAC algorithm represents the most conservative CAC algorithm one could possibly require, the GAN CAC algorithm is still more conservative for certain mixes of source types. This would not necessarily be bad if the mixes of source types for which this occurred were unlikely to occur in an operational network. However, there is no reason to assume this to be the case.

Chapter 6 Simulating TCP/IP traffic over ATM

The best way to quantify the performance of TCP/IP over ATM is by measuring actual traffic throughput in an operational ATM network. Measurement data of this kind has been collected for UBR and DBR, see [10] and [7] for example. However, for GFR and ABR no such data exists yet. In the case of GFR this is because current ATM switches do not yet support GFR. In the case of ABR the reason is that the first ABR networks have only recently become operational. Therefore, in order to gain some insight into the expected performance of TCP/IP over these ATCs, simulations are useful.

Various simulation studies of TCP/IP traffic over ATM have been performed and can be found in literature, see for example [21], [25], [3], [4], [24], [20], [11], [12], and [22]. However, most of these studies focus on one particular ATC only and do not compare various ATCs on the basis of their performance. Comparing the results of the various simulation studies for different ATCs is also not possible due to the differing simulation setup used. Furthermore, few studies simulate both the case of one TCP session over one ATM VC, as well as the case of multiple TCP sessions over one VC. Also, the standardisation of GFR by the ITU-T[18] is still under way and older GFR simulations are not always fully compliant with this evolving standard. Therefore, it was deemed useful to perform a new set of simulations.

In line with the conclusions reached in Chapter 2, only the ABR and GFR ATCs were simulated in this. In addition section 6.2.3 introduces a non-standardised ATC called GFR lite which was also simulated. First, sections 6.1, 6.2 and 6.3 detail the simulation setup used, after which the results obtained are discussed in section 6.4. Then, section 6.5 draws conclusions about the performance of the ATCs simulated based on these results.

6.1 General setup

This section details the setup and parameters common to all the simulations performed in this. Most importantly, the native and router setup are introduced, as well as the fast and slow background traffic scenarios. Furthermore, the switch architecture that was used is discussed along with various QoS and CAC issues.

The simulation packet used to perform the simulations was STCP, see Appendix 1. The advantage of this simulation packet is that it does not model the behaviour of TCP/IP but uses the original TCP/IP source code found in BSD 4.4 Lite. Furthermore,

[18] GFR has already been standardised as ATM service category by the ATM-Forum, see [2].

the source code of STCP can be freely downloaded from internet, allowing modifications to be made quite easily. The TCP fast and slow timers, see [26], were kept at their default values of 200 ms and 500 ms respectively for all simulations. The fast timer is used by the TCP delayed ACK algorithm whereas the slow timer is used by various timers, such as the retransmission timer, and denotes the timer granularity.

Two distinct simulation setup were considered:

1. the **native setup**, where TCP/IP sources are directly connected to the ATM network and each has it's own VC to it's desired destination. Only one TCP session is assumed per VC, see the native mapping in section 2.1.
2. and the **router setup**, where TCP/IP sources are connected indirectly to the ATM network via a router. TCP/IP traffic belonging to multiple sources is now transferred over one and the same VC between routers. There are thus multiple TCP sessions over the same VC, see the router mapping in section 2.1.

All TCP/IP sources were considered to be greedy in that they always have data ready to send. Destinations, on the other hand, have no data to transfer and only transmit ACKs. An example of such a TCP session would be an FTP session transferring a large file.

Besides the TCP/IP traffic generated by the sources, ATM background traffic was also generated and fed into the network. This traffic has a higher priority than the TCP/IP traffic and is intended to simulate traffic generated by other ATCs such as DBR. This background traffic consists of bursts of cells, the lengths of which are exponentially distributed, interspersed with exponentially distributed idle periods during which no cells are sent. Such traffic is often called on/off traffic. The following two scenarios were considered:

1. the **slow scenario**, where the background traffic consists of bursts of cells with a mean size of 24000 cells sent at a cell rate of 243333 cells/s and an average cell rate of 10000 cells/s.
2. the **fast scenario**, which is identical to the slow scenario except that the mean burst size is now 1000 cells, implying that the on and off fluctuations in the background traffic are much more rapid.

Note that as the average cell rate decreases in the slow scenario or the average cell rate increases in the fast scenario, the traffic behaviour converges to that of 'constant' traffic. In the slow scenario this is because the on or off periods during which the traffic is constant become longer and longer, whereas in the fast scenario this is because the fluctuations in traffic volume become so rapid that they 'average' out. The fast and slow scenarios detailed above were chosen to be representative of non 'constant' domain.

The simulated ATM switches were modelled along the lines of the Ascend CBX 500, a switch currently being widely deployed in operational networks. More specifically, the link rate was set equal to 365,000 cells/s, which corresponds approximately with

155 Mbps, and the buffer size available for TCP/IP traffic was set equal to 16000 cells. The buffer available for the higher priority background traffic was set equal to 128 cells. The ABR implementation used in the simulations was also modelled along the lines of the implementation used in this switch. As the CBX 500 does not implement GFR, various GFR implementations were simulated, all designed to require a minimum in modifications to the existing switch architecture.

QoS class 3, see [16], was assumed for all VCs. This implies that no bounds are placed on the maximum delay a cell may experience, while the CLR is limited to a maximum in the order of 10^{-6}. In order to ensure that this maximum is respected a CAC algorithm is required. For GFR and GFR lite the most obvious candidate for such a CAC algorithm would be the GAN CAC algorithm, see section 5.1, as this is implemented in the CBX 500 by default. However, as section 5.4 concludes that the EMW lossless CAC algorithm is preferable to the GAN CAC algorithm, the EMW lossless CAC algorithm is used instead. This is not entirely unrealistic as the AEEM implemented by KPN Telecom uses the EMW lossless CAC to admit scheduled PVCs. For ABR the CAC algorithm used in the simulations simply reserves a cell rate equal to the requested MCR for a given ABR VC.

The propagation delay was set to τ=30 ms, a value typical for a WAN or Internet environment. In a LAN environment this delay is much smaller. The actual round trip time of any given TCP session varies however, depending on the level of congestion in the network. If the network is congested then this implies that the switch buffers in the network are full and cells will experience an additional queueing delay.

All simulations were run for a duration of 120 seconds. Increasing this value further was not found to significantly influence the resulting performance results. Furthermore, each simulation was repeated 3 times, each run being initialised with a different set of random seeds. These seeds determine the realisation of the background traffic.

Table 5 summarises the values of the global setup parameters specified so far.

ATM switch	Link rate = 365000 cells/s Buffer TCP/IP traffic = 16000 cells Buffer background traffic = 128 cells
Background traffic, Slow scenario	PCR = 243333 cells/s Average cell rate = 100000 cells/s Mean burst size = 24000 cells
Background traffic, Fast scenario	PCR = 243333 cells/s Average cell rate = 100000 cells/s Mean burst size = 1000 cells
TCP timers	Fast timer = 200 ms Slow timer = 500 ms
Propagation delay	τ = 30 ms
Simulation	Duration = 120 s Random seeds run 1 = 612, 1113, 1025 Random seeds run 2 = 971, 24, 3612 Random seeds run 3 = 4, 9918, 352

Table 5 Global simulation parameters

6.2 Native setup

This section explains the native setup more fully. The specific details of how each ATC was simulated in the native setup are given in the appropriate subsections.

Figure 20 shows the setup that was used to simulate TCP/IP sources directly attached to an ATM network. Two source sets were used, labelled 1 and 2 respectively. Sources in source set 1 each have their own VC passing through switches 1 and 3 and ending at the desired destination. For source set 2 the same holds except that the VCs now pass through switch 2 instead of switch 1. ACKs are transferred from the destinations back to the sources via a different route through switch 4.

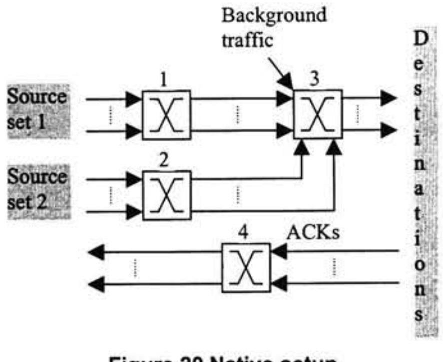


Figure 20 Native setup

Switches 1 and 2 have their full link rate available for the TCP/IP traffic generated by the sources. The traffic passing through these two switches meets at switch 3, which, in addition, is also being fed with high priority DBR background traffic. This background traffic leaves a cell rate of 121667 cells/s unreserved and available for reservation by the source VCs carrying TCP/IP traffic. Switch 3 thus forms a bottleneck in the network and congestion can be expected to occur as a results.

6.2.1 GFR

This subsection completes the details of the setup used for the GFR simulations in the native setup. Four different GFR implementations are introduced as well as the small, medium and large source types which make up the source sets 1 and 2.

The four GFR implementations are all based on a first in first out (FIFO) queue with an EPD discard threshold B_{EPD} set at 3000 cells[19], see Figure 21. The buffer of 13000 cells above this threshold is reserved for QoS-eligible frames.

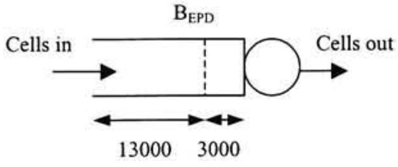


Figure 21 Schematic diagram of GFR buffer implementation

Listed in order of increasing complexity the implementations are:

1. **Basic GFR**, where as soon as the total number of cells in the buffer exceeds B_{EPD}, non-QoS eligible frames are discarded irrespective of the VC to which they belong.
2. **Basic+ GFR**, where as soon as the number of cells belonging to non-QoS eligible frames in the buffer exceeds B_{EPD}, non-QoS eligible frames are discarded irrespective of the VC to which they belong.
3. **Per VC GFR**, where, in addition to B_{EPD}, each VC has it's own discard threshold and a non-QoS eligible frame belonging to a particular VC is discarded only if
   - the total number of cells exceed B_{EPD} and,
   - the number of cells belonging to that particular VC exceeds VC_{EPD}.
4. **Per VC+ GFR**, where, in addition to B_{EPD}, each VC has it's own discard threshold and a non-QoS eligible frame belonging to a particular VC is discarded only if:
   - the total number of cells belonging to non-QoS eligible frames exceed B_{EPD} and,
   - the number of cells belonging to non-QoS eligible frames and that particular VC exceeds VC_{EPD}.

[19] The value of 3000 cells was chosen somewhat arbitrarily. A larger value results in less buffer being reserved for QoS eligible traffic and fewer VCs being admitted onto the network by the CAC algorithm. A smaller value unduly restricts the flow of non-QoS eligible traffic. The value of 3000 cells was found to be a reasonable compromise.

The four implementations described above are valid for both the GFR1 and the GFR2 variant of the GFR ATC, see section 2.3.1. To simplify the simulation of the above implementations, GFR2 was assumed. This implies that all non-QoS eligible frames were tagged as CLP=1, whereas QoS eligible frames have CLP=0. The simulation outcome does not depend on this assumption however, and the results are valid for both GFR1 and GFR2.

Note that no per-VC scheduling mechanisms such as weighted fair queueing, see for example [25], were simulated. The reason for this is twofold. Firstly, such mechanisms are not implemented in the Ascend CBX 500 and would therefore require considerable adaptation of the existing switch architecture. The four implementations above require only a minimum in adaptation of the existing architecture. Secondly, a GFR like implementation using per VC scheduling was simulated and is discussed in section 6.2.3.

Three source types were defined, labelled **small**, **medium**, and **large**. Table 6 lists the values of their respective parameters.

Small	MCR = 150 cells/s
	PCR = 23585 cells/s
	MFS = 32 cells
	MBS = 128 cells
	TCP window scale factor = 0 (65535 bytes)
	e_0 = 1140 cells/s
	VC_{EPD} = 4 cells

Medium	MCR = 4717 cells/s
	PCR = 58962 cells/s
	MFS = 94 cells
	MBS = 376 cells
	TCP window scale factor = 1 (2×65535 bytes)
	e_0 = 4717 cells/s
	VC_{EPD} = 128 cells

Large	MCR = 23585 cells/s
	PCR = 365000 cells/s
	MFS = 190 cells
	MBS = 950 cells
	TCP window scale factor = 4 (16×65535 bytes)
	e_0 = 23585 cells/s
	VC_{EPD} = 642 cells

Table 6 Parameters of the three source types in the native GFR setup

The first four parameters in Table 6 are the standard GFR traffic parameters, see section 2.3.1, and they define the GFR VC. The first three parameters were chosen based on expected typical source types. The small source type corresponds roughly to a typical Ethernet client with a small MCR and MFS, and a PCR equal to the maximum LCR of an Ethernet cable. The large source type corresponds to a possible corporate, native ATM client directly attached to the ATM network with a large MCR and MFS, and a maximum PCR equal to the LCR of the switches. The medium source type fills in the gap between these two. The MBS values were determined using the steady state TCP throughput analysis, see section 4.3.3. The minimum value was set to four times the MFS in order to accommodate the FRR algorithm[20]. For the small and medium sources this value was sufficient. For the large source, however, an even larger value was required.

The fifth parameter in Table 6 is the TCP window scale factor and this determines the maximum TCP window size, see the section on the window scale option in [26] for further details. The value of the TCP window scale factor was chosen in order to allow each source type to increase it's window size to at least W_{max}, the maximum steady state window size based on the MCR and MFS parameters, see section 4.3.2.

The sixth parameter shown in Table 6 is the EMW lossless effective bandwidth e_0, see section 5.2. This value is determined by the PCR< MCR and MBS traffic parameters in combination with the unreserved cell rate[21] and buffer space reserved for QoS eligible frames at switch 3. If for a given mix of source types the sum of the e_0 values is smaller than or equal to the cell rate available for TCP/IP traffic then the mix will be admitted by the EMW lossless CAC algorithm. Table 7 shows one such admissible mix and this mix was used for the GFR simulations. Listed are the number of sources of each source type in each of the source sets 1 and 2. The total number of sources is 23 and the sum of their effective bandwidths can be shown to be equal[22] to the unreserved bandwidth at switch 3. The network is therefore full in the sense that a new VC with an MCR greater than zero will not be admitted by the CAC algorithm.

Source set 1	4 small 3 medium 1 large
Source set 2	8 small 5 medium 2 large

Table 7 Number and type of sources in source sets 1 and 2

The seventh and final parameter listed in Table 6 is the value of the per VC discard threshold, needed for both the per VC and the per VC+ GFR implementations. These values were obtained by dividing the 3000 cells below the B_{EPD} threshold amongst

[20] The FRR algorithm requires 3 duplicate ACKs to be received prior to retransmission of a discarded frame. Therefore a MBS minimum of 3×MFS is desirable. However, simulations indicate that the first ACK triggered by an out of order segment is often not a duplicate ACK due to alterations in the window size field of the TCP header. Therefore, retransmission often takes place only after reception of 4 ACKs triggered by out of order segments, making a minimum MBS of 4×MBS desirable.

[21] An alternative method would be to treat the DBR background traffic as a GFR connection with MCR=PCR in which case e_0 should be calculated with the total link cell rate of the switch and not the unreserved cell rate. In this case this leads to higher values of e_0.

[22] Actually the sum of the e_0 values is slightly larger than the available cell rate of 121667 cell/s at switch 3. However, as EMW lossless is very conservative (CLR=0) this will be ignored.

the sources in source sets 1 and 2 proportional to their respective MCR values. Another option would have been to divide the 3000 cells equally between all sources, or to divide them proportional to the MBS. However, the Ascend CBX 500 allow per VC thresholds to be set proportional to the MCR, so this is what was implemented in the simulations.

6.2.2 ABR

This subsection completes the description of the setup used to simulate TCP/IP traffic over ABR in the native setup. Actually, the native setup used differs slightly from that shown in Figure 20 in that ACKs are no longer routed back independently over a separate switch, see Figure 22. Instead, ACKs are routed back in the opposite direction to the TCP/IP traffic, together with the periodic RM cells generated by the ABR congestion loop, see Chapter 3. The differing setup was used purely for simulation convenience. Routing the ACKs back over a separate switch or together with the RM cells does not significantly influence the simulation results.

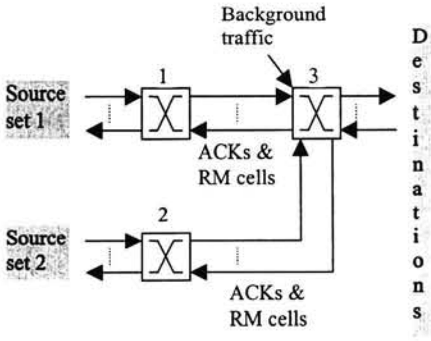


Figure 22 ABR native setup

At switch 1 the ACR value for each individual VC passing through is regulated by the ABR congestion control loop existing between switch 1 and switch 3. Similarly, for switch 2 the ACR for each VC is regulated by the loop between switch 2 and switch 3. Furthermore, each individual source-destination pair has it's own individual ABR congestion control loop to regulate the ACR at the source. Each source periodically sends RM cells into the network and these are looped back by the destination. The intermediate switches along the VC update the explicit cell rate field in the RM cell according to the current value of the ACR in the switch, a mechanism also called explicit rate marking, see [17]. The source then adapts it's own ACR according to the value specified in the explicit cell rate field. The interval between subsequent RM cells, denoted by T_{RM}, was set equal to 30 ms. This value was also used for the congestion loops between the switches.

Similar to the GFR setup, three source types are defined, see Table 8, and the same source sets are used as in the GFR simulation, see Table 7. For ABR a simpler CAC algorithm is used than EMW lossless. Now a VC is admitted only if the sum of the MCR values of all admitted VCs is less than cell rate unreserved in the network.

Source sets 1 and 2 thus again form an admissible set. In fact, more VCs can be safely admitted but this was not done in order to compare the ABR and GFR simulations as fairly as possible.

The first two parameters listed Table 8 are the ABR traffic parameters. The value of these was chosen identical to the values used in the GFR simulations, see Table 6. The third parameter listed is the initial cell rate (ICR) and it's value is given by

$$ICR = MCR + \frac{PCR - MCR}{2^k}, \tag{30}$$

with k=6. The ICR denotes the cell rate at which each source type initially begins sending data, before the ABR congestion control loop has had a chance to adapt the allowed cell rate (ACR). The formula used above corresponds with the formula used to configure the ICR in the Ascend CBX 500. In general, k can vary, allowing the ICR to range anywhere between the MCR and the PCR. A value of k=6 was chosen to limit the ICR in order so as not to immediately flood the network with data.

Small	MCR = 150 cells/s
	PCR = 23585 cells/s
	ICR = 516 cells/s
	RDF = 1/4
	RIF = 1/2048
	VC_{cong} = 4
	TCP window scale factor = 0 (65535 bytes)

Medium	MCR = 4717 cells/s
	PCR = 58962 cells/s
	ICR = 5565 cells/s
	RDF= 1/4
	RIF = 1/128
	VC_{cong} = 114
	TCP window scale factor = 1 (2×65535 bytes)

Large	MCR = 23585 cells/s
	PCR = 365000 cells/s
	ICR = 28920 cells/s
	RDF = 1/4
	RIF = 1/128
	VC_{cong} = 569
	TCP window scale factor = 4 (16×65535 bytes)

Table 8 Parameters of the three source types used in the native ABR setup.

The next two parameters listed in Table 8 are the RDF and RIF factors of the ABR congestion control loop, see Chapter 3. The value of these was determined by using the ABR simulation tool described in section 3.6. To this end, the 23 individual VCs listed in Table 7 were modelled by one aggregate VC, as detailed in section 3.1, and the background traffic was modelled as having fixed instead of exponentially distributed on and off periods. The values of the RDF and RIF were then determined by requiring that the CLR for the aggregate VC was equal to zero in both the slow and fast background scenarios[23]. This required a minimum RDF of at least 1/4 and a maximum RIF of at most 2^{-6}. The values of the individual VC RDF and RIF parameters then follow from these two values in combination with the two fairness equations, see section 3.7, and the following two equations

$$RIF \times PCR = \sum_i RIF_i \times PCR_i ,$$

$$(1 - RDF) \times ACR = \sum_i ACR_i (1 - RDF_i) .$$

Here, the parameters on the left side of the equations refer to the aggregate VC, whereas the parameters on the right side of the equation refer to the individual VCs which are indexed by the letter *i*. The idea behind these two equations is that, at the end of a time interval T_{RM}, the updated ACR for the aggregate VC should be equal to the sum over the updated ACR_i values. The resulting RDF and RIF values were then rounded down to the nearest $1/2^k$ value, with *k* an arbitrary integer. This in line with the configurable RDF and RIF values in the Ascend CBX 500.

The sixth parameter in Table 8 lists the value of the per VC congestion threshold. This is used in combination with the global congestion threshold B_c, set at the Ascend CBX 500 default value of 2662 cells, to determine whether or not a VC is congested. A VC is defined as being congested if both the total number of cells in the buffer is greater than B_c, and the number of cells in the buffer belonging to that VC is greater than VC_{cong}. The value of the threshold was determined by sharing the 2662 cells below the global congestion threshold amongst all 23 VCs, each VC receiving an MCR proportional share. This is in line with the discussion on the per VC thresholds and fairness, see section 3.7.

The final parameter in Table 8 lists the value of the TCP window scale factor, chosen identical to that in the GFR setup, see Table 6.

6.2.3 GFR lite

This subsection completes the description of the setup used to simulate TCP/IP traffic over GFR lite in the native setup. The GFR lite implementation considered below is identical to GFR1 (no cell tagging) in all aspects except that cell marking is not allowed and all cells must be sent as CLP=0. As such, the implementation considered does not coincide with a standardised ATC. However, a proposal for an ATC similar to GFR lite has just recently been submitted to the ITU-T as a simplified variant of the GFR ATC.

The reasons for simulating GFR lite are twofold:

1. GFR lite with per VC scheduling based on the ABR congestion control loop can be implemented without requiring major modifications to the Ascend CBX 500

[23] The slow background scenario was determining for the required RDF and RIF values.

switch architecture. In contrast, a per VC scheduling implementation of GFR would require major modifications, the reason only FIFO scheduling was considered, see section 6.2.1. However, being similar to GFR, the performance of such a GFR lite implementation can be considered indicative of how GFR would perform if a per VC scheduling mechanism were implemented instead of a FIFO scheduler.

2. GFR lite has the potential of being better suited for TCP/IP traffic than the GFR ATC. This is because GFR lite does not use the CLP bit, requiring less complexity from the network as illustrated by the buffer implementations used.

The following two GFR lite implementations were simulated:

1. **UBR+**, based on the buffer implementation shown in Figure 23 in combination with FIFO scheduling. Basically, the implementation is the same as that used for the GFR simulations, see Figure 21, except that the EPD threshold B_{EPD} is now set to 13000 cells[24] instead of 3000 cells. The reason for this difference is that no buffer is now reserved for QoS eligible frames above the threshold. A frame arriving at a switch is accepted if the number of cells currently in the switch buffer is below B_{EPD}. If this is not the case then the frame is discarded, irrespective of which VC it belongs to. From the network viewpoint, UBR+ is thus very much like UBR with EPD. The main difference is that a minimum cell rate is guaranteed.

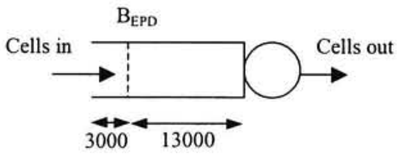


Figure 23 Schematic diagram of UBR+ buffer implementation

2. **ABR-**, also based on the buffer implementation shown in Figure 23 but now in combination with per VC scheduling based on the ABR congestion loop. This implies that cells belonging to the individual VCs are emptied from the buffer at cell rates determined by the individual ACR values of these VCs. From the network viewpoint, ABR- thus resembles ABR with EPD. The main difference is that cell loss can now occur.

Three source types were again defined, identical to those defined in the GFR setup, see Table 6. Also, the EMW lossless CAC algorithm was again used and as the buffer available for cells belonging to QoS frames remains equal to 13000 cells, the two source sets given in Table 7 still form an admissible mix. Finally, the parameters of the ABR congestion control loop were chosen identical to those used in the ABR setup, see Table 8.

[24] The value of 13000 cells was chosen somewhat arbitrarily. Lowering this value only decreases the buffer space available for traffic and unduly restricts the flow. Increasing the value increases the risk of buffer overflow and the subsequent forced discard of cells, leading to corrupt frames. The value of 13000 cells was found to be a reasonable compromise.

6.3 Router setup

This section explains the router setup more fully, highlighting the differences with the native setup. The specific details of how each ATC was simulated in the router setup are given in the appropriate subsections.

Figure 24 shows the setup used to simulate TCP/IP sources indirectly attached to an ATM network via a router. Similar to the native setup, there are two source sets labelled source set 1 and 2 respectively. However, unlike in the native setup, each source no longer has it's own individual VC connecting it to it's desired destination. Instead, all the TCP/IP traffic generated by the sources in source set 1 is transferred over one and the same VC between router 1 and switch 3. Likewise, all the traffic between router 2 and switch 3 is transferred over only one VC. The ACKs are routed back separately over switch 4.

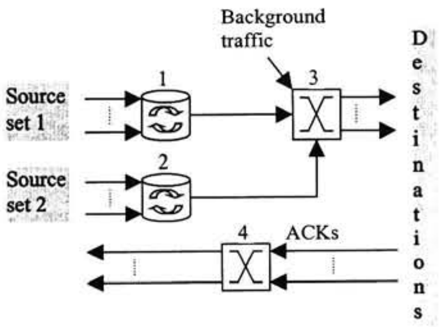


Figure 24 Router setup

Like in the native setup, background traffic is again fed into switch 3. This switch will therefore form a bottleneck in the network and congestion can be expected to occur as a result.

Both routers are assumed to have a link cell rate equal to 365000 cells/s, identical to the link rate of the switches. The buffer size was set to 128000 cells, a value found to be large enough to avoid frame loss in the routers. Typically, router buffers are indeed large but this is primarily done to avoid the router architecture from influencing the TCP/IP performance.

6.3.1 GFR

This subsection completes the details of the setup used to simulate GFR in the router setup. The parameters of the two VCs setup between routers 1 and 2 and switch 3 are given, as well as the source types making up source sets 1 and 2. The setup used was tuned so as to match as closely as possible the GFR simulations for the native setup. In this way the results for both the native and router setup can be compared more readily.

The same GFR implementations as in the native setup were simulated, namely basic, basic+, per VC and per VC+ GFR, see section 6.2.1. Three source types were again defined, also labelled small, medium and large respectively. Table 9 lists their respective parameters.

Small	LCR = 23585 cells/s
	MSS = 1488 bytes
	TCP window scale factor = 0 (65535 bytes)

Medium	LCR = 58962 cells/s
	MSS = 4464 bytes
	TCP window scale factor = 1 (2×65535 bytes)

Large	LCR = 365,000 cells/s MSS = 9072 bytes TCP window scale factor = 4 (16×65535 bytes)

Table 9 Parameters of the three source types in the GFR router setup

The first parameter listed in Table 9 is the link cell rate (LCR) and denotes the maximum cell rate at which the source type can send frames to it's router. This value was chosen identical to the value of the PCR in the native setup, see Table 6.

The second parameter in Table 9 is the TCP maximum segment size (MSS). A segment of this size is augmented with a TCP header of 20 bytes, an IP header of 20 bytes and an AAL5 header of 8 bytes before being chopped in blocks of 48 bytes for placement in the payload of an ATM cell, see section 2.1. Therefore, a segment of MSS bytes results in (MSS+48)/48 ATM cells. The value of the MSS in bytes was chosen so as to result in the MFS number of cells given listed in Table 6.

The third and final parameter listed in Table 9 is the TCP window scale factor already discussed in section 6.2.1. The values listed are identical to those in the native setup, see Table 6.

The composition of the source sets 1 and 2 was kept identical to that in the native setup, see Table 7.

Table 10 lists the values used for the parameters of VC 1, connecting router 1 to switch 3, and VC 2, connecting router 2 to switch 3. The first four parameters are the standard GFR traffic parameters, see section 2.3.1. The MCR and MBS values were chosen equal to the sum over the source sets of the individual e_0 and MBS values in the native setup[25], see Table 6. The MFS value was set equal to the maximum value

[25] Another option would have been to set the MCR equal to the sum of the MCRs of the individual VCs. However, as each VC has a dedicated cell rate of e_0 at it's disposal in the native setup, it was deemed fairer for the sake of comparison to set the MCR equal to the sum of the individual e_0 values.

used in the native setup. The PCR value was set equal to the maximum possible, namely the link cell rate of the routers.

VC 1
(router 1 - switch 3)
	MCR = 42296 cells/s
	PCR = 365000 cells/s
	MFS = 190 cells
	MBS = 2590 cells
	e_0 = 42296 cells/s
	VC_{EPD} = 1042 cells

VC 2
(router 2 - switch 3)
	MCR = 79875 cells/s
	PCR = 365000 cells/s
	MFS = 190 cells
	MBS = 4804 cells
	e_0 = 79875 cells/s
	VC_{EPD} = 1956 cells

Table 10 Parameters of the two VCs in the GFR router setup

The fifth parameter in Table 10 is the EMW lossless effective bandwidth, see section 5.2. The sum of these two effective bandwidths is identical to the sum of the individual effective bandwidths in the native setup. The two VCs can therefore be admitted onto the network by the EMW lossless CAC, as was the case with the VCs in the native setup.

The sixth and last parameter in Table 10 is the per VC discard threshold which is set equal to the sum of individual per VC discard thresholds in the native setup, see Table 6. As there are only two VCs in the router setup, the effect of the per VC thresholds on the performance is expected to be less than that in the native setup where there are 23 VCs.

6.3.2 ABR

This subsection completes the description of the setup used to simulate TCP/IP traffic over ABR in the router setup. Actually, the router setup used differs slightly from that shown in Figure 24 in that ACKs are no longer routed back independently over a separate switch, see Figure 25. Instead, ACKs are routed back in the opposite direction to the TCP/IP traffic, together with the periodic RM cells generated by the ABR congestion loop. Like in the case of the ABR native setup, the difference is purely for simulation convenience. Routing the ACKs back over a separate switch or together with the RM cells does not significantly influence the simulation results.

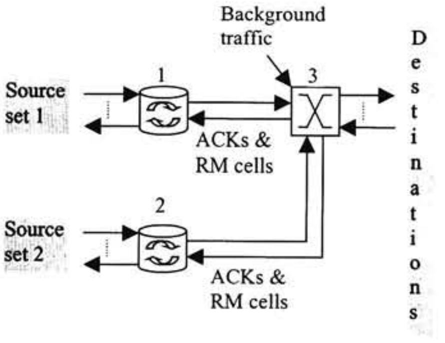


Figure 25 ABR router setup

In contrast to the native ABR setup, no RM cells are now generated by the TCP/IP sources. Only switch 3 periodically generates RM cells, sending them to either router 1 or router 2. The interval T_{RM} between subsequent RM cells was set equal to 30 ms, identical to the native setup.

The same three source types as in the GFR setup were used, see Table 9, as well as identical source sets, see Table 7. Table 11 lists the values of the parameters used for the two VCs. The first two parameters are the ABR traffic parameters and their values were chosen identical to the values used in the GFR setup, see Table 10. The third parameter is the ICR and it's value was determined using equation (30), as in the native ABR setup. Likewise, the values of the RIF and RDF parameters were determined by using the ABR simulator described in section 3.6, as was the case in the native ABR setup. Finally, the value of the per VC congestion threshold is listed, each VC again receiving an MCR proportional share of the 2662 cells below the global congestion threshold B_c.

VC 1
(router 1 - switch 3)
	MCR = 42296 cells/s
	PCR = 365000 cells/s
	ICR = 47338 cells/s
	RDF = 1/4
	RIF = $1/2^8$
	VC_{cong} = 922 cells

VC 2
(router 2 - switch 3)
	MCR = 79875 cells/s
	PCR = 365000 cells/s
	ICR = 84330
	RDF = 1/4
	RIF = $1/2^7$
	VC_{cong} = 1740 cells

Table 11 Parameters of the two VCs in the ABR router setup

6.3.3 GFR lite

This subsection completes the description of the setup used to simulate TCP/IP traffic over GFR lite in the router setup. Only the UBR+ implementation was now considered as the ABR- implementation requires at least one additional switch to be added between the source and destination in the router setup in order for the ABR congestion control loop to function.

The three source types used were identical to those used in the GFR setup, see Table 9. Likewise, the source sets used were identical to those in Table 7 and the parameters of the two VCs are given by Table 10.

6.4 Results

This section presents and discusses the results of the simulations detailed in the previous sections. In subsection 6.4.1 the performance of the different ATC implementations are compared with respect to the number of cells transferred. In subsection 6.4.2 the same is done with respect to the number of frames. Subsection 6.4.3 compares the implementations with respect to how efficiently each succeeds in utilising the total cell rate available in the network. Then, in subsection 6.4.4, the implementations are compared on the basis of how fairly the cell rate available in the network is shared between the competing VCs. Subsection 6.4.5 compares the implementations on the basis of how fairly identical sources are treated and finally, subsection 6.4.6 comments briefly on the CLR.

For completeness the seven different ATC implementations that were simulated are again listed here:

1. Basic (GFR),
2. Basic+ (GFR),
3. Per VC (GFR),
4. Per VC+ (GFR),
5. ABR,
6. UBR+ (GFR lite),
7. ABR- (GFR lite).

In addition two setups were considered, the native and the router setup, as well as two background scenarios, the slow and the fast scenario. This leads to a total of 4 different setup scenarios aptly named:

1. Router slow,
2. Native slow,
3. Router fast,

4. Native fast.

The ABR- GFR lite implementation forms an exception in that it was only simulated for the native setup and not the router setup, see subsection 6.3.3.

6.4.1 MCR guarantee

This subsection compares the different ATC implementations on the basis of either the mean cell rate achieved by the three different source types in case of the native setup, or the mean cell rate achieved by the two VCs in case of the router setup. Of particular interest is whether or not a mean cell rate equal to at least MCR is achieved. The differences between the results for the two background scenarios are also discussed.

6.4.1.1 Native setup

Figure 26 shows the mean cell rate achieved by the small source types in the native setup for both the fast and slow background scenarios. The horizontal line indicates the MCR equal to 150 cells/s. Clearly, all the ATC implementations under consideration achieved a mean cell rate higher than the MCR although, for the case of ABR and ABR-, this was only barely the case.

Figure 27 similarly shows the mean cell rate achieved by the medium source types in the native setup as well as the corresponding value of the MCR. Clearly, all ATC implementations achieve a mean cell rate in excess of the MCR. For the large source types, however, this is not always the case as is illustrated in Figure 28. Clearly, both the basic and per VC GFR implementations fail to utilise the guaranteed cell rate of MCR in the slow background scenario, and the basic+ GFR implementation succeeds in doing so only barely. The per VC+ GFR implementation, along with the other remaining ATCs do succeed in achieving a mean cell rate clearly in excess of the MCR.

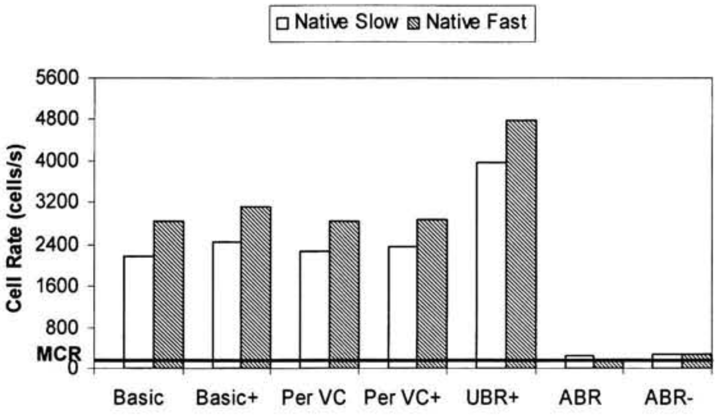


Figure 26 Mean cell rate achieved by the small source types in the native setup

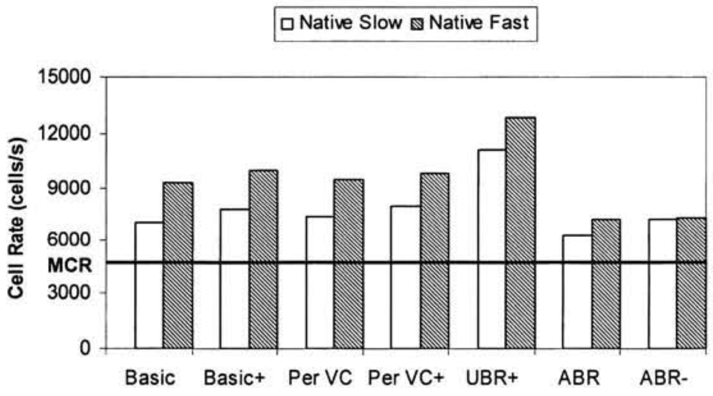


Figure 27 Mean cell rate achieved by the medium source types in the native setup

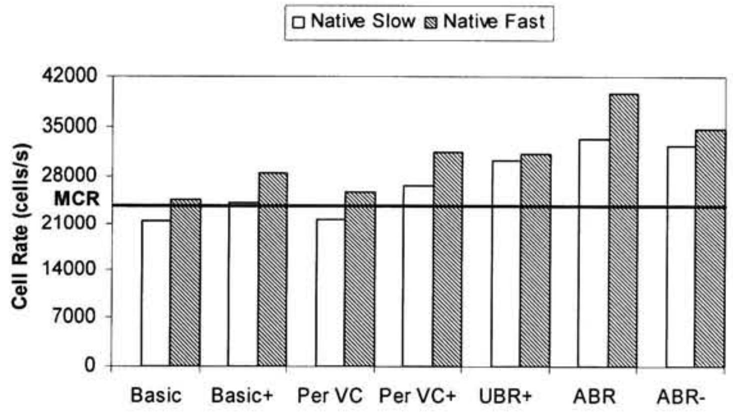


Figure 28 Mean cell rate achieved by the large source types in the native setup.

6.4.1.2 Router setup

Figure 29 shows the mean cell rate achieved by VC 1 in the router setup for both the slow and fast background scenarios. The horizontal line indicates the MCR equal to 42296 cells/s. Clearly, all ATC implementations achieve a mean cell rate in excess of the MCR. Figure 30 similarly shows the mean cell rate achieved by VC 2 along with the corresponding MCR value. Now the basic and per VC GFR implementations achieve a mean cell rate only just equal to the MCR, whereas the other implementations have no problem achieving a cell rate in excess of the MCR.

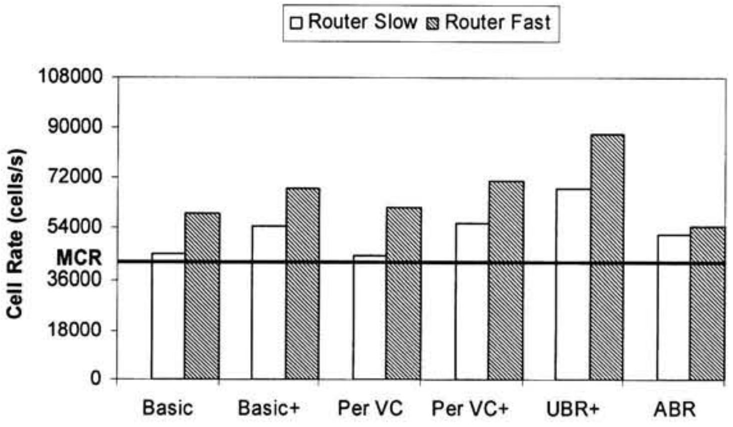


Figure 29 Mean cell rate achieved by VC 1 in the router setup.

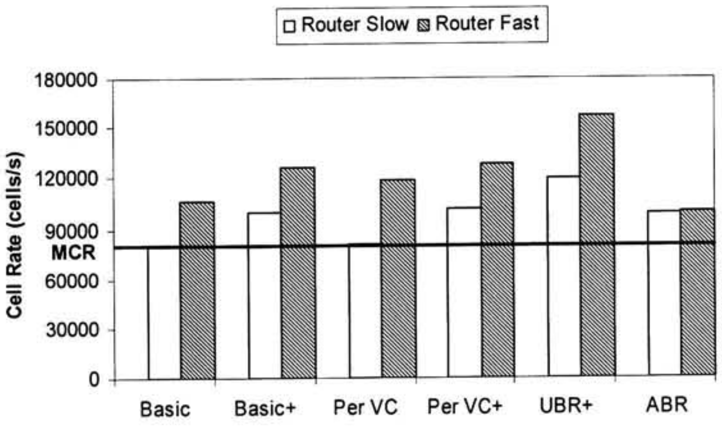


Figure 30 Mean cell rate achieved by VC 2 in the router setup.

In general it is clear from all five figures that the cell rate achieved in the slow background scenario is always lower than that achieved in the fast background scenario. This indicates that the slow scenario resulted in sources performing a larger number of TCP slow starts. This is due to the severe congestion and subsequent frame loss which occurs when the volume of background traffic suddenly increases after being small for a relatively long period, see the ramp down analysis in section 3.4. For the fast scenario this also occurs but as the fluctuations in traffic volume are more rapid they average out and the congestion resulting from a sudden increase in traffic volume is less severe.

6.4.2 TCP goodput

This subsection compares the different ATC implementations with respect to the mean TCP goodput achieved by the three different source types. Here TCP goodput is defined as the number of bits of data received per second in complete, non duplicate frames at the destination. This does not include the overhead produced by TCP, IP and AAL5 headers. The differences in the results between the native and router setup are discussed, as well as those between the slow and fast background scenario.

If no partial or duplicate packets arrive then the ATM cell rate and the TCP goodput are related as follows:

$$\text{TCP goodput} = (\text{ATM cell rate}) \times 48 \times 8 \times \frac{\text{MSS}}{\text{MSS} + 48}.$$

The factor 48 comes from the 48 byte payload of the ATM cell whereas the factor 8 performs the conversion from bytes to bits. The MSS/(MSS+48) takes into account the TCP, IP and AAL5 headers, 48 bytes in total, see section 2.1.

Figure 31 shows the mean goodput achieved by the small source types. The horizontal line indicates the goodput that one would expect on basis of an MCR equal to 150 cell/s using the above conversion formula[26]. This value will be called the MCR goodput. Clearly, all ATC implementations succeed in achieving a higher mean goodput than the MCR goodput, although the ABR and ABR- implementations succeed in doing so only barely.

Figure 32 shows the mean goodput for the medium source type along with value of the corresponding MCR goodput. It is clear that for the native setup all ATC implementations achieve a goodput in excess of the MCR goodput. For the large source types, however, this is not the case, as can be seen in Figure 33. Both the basic and basic+ GFR implementations do not succeed in achieving the MCR goodput in the native slow setup scenario. The basic+ GFR implementation does succeed in doing so but only barely.

In general it is clear from all three figures that the mean goodput is lower for the slow background scenario than it is for the fast background scenario. This is line with the lower mean cell rate in the slow scenario, see the previous section. With respect to difference between the native and router setup the three figures indicate a different trend. The small source types generally achieve a higher goodput in the router setup than in the native setup, the notable exception being UBR+. This behaviour is reversed for the large source types. For the medium source types the goodput is roughly equal for both setup, at least for the GFR implementations. The reason for this behaviour is that in the router setup no distinction is made between different sources with respect to CLP tagging. All the traffic flowing to the routers from the individual sources is tagged as if originating from one virtual source. The net result of this is that the number of frames that are tagged belonging to small source types will be lower in the router setup than in the native setup where tagging occurs for each source type individually, on the basis of the MCR. Similarly, the number of frames tagged belonging to large source types will be higher in the native setup as the large source types are no longer 'protected' by a large MCR value.

[26] The MCR value applies only to the native setup as in the router setup the three source types have no MCR traffic parameter defined.

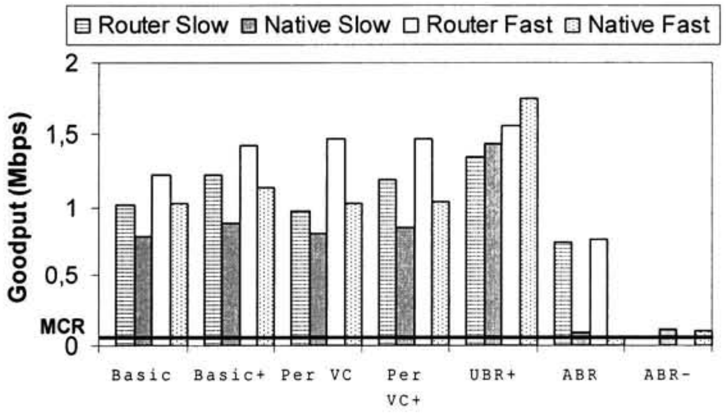


Figure 31 Mean goodput achieved by the small source types

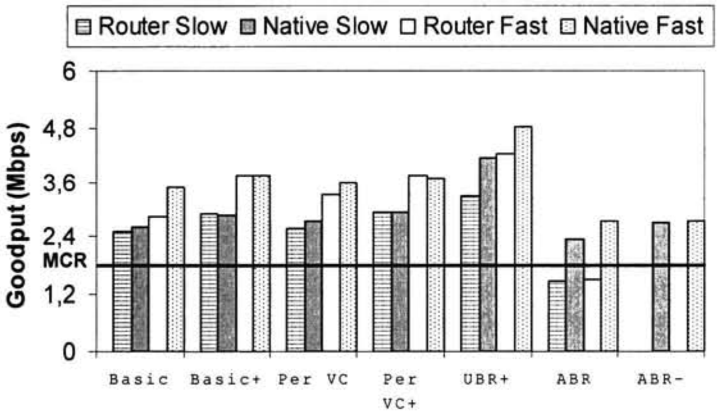


Figure 32 Mean goodput achieved by the medium source types

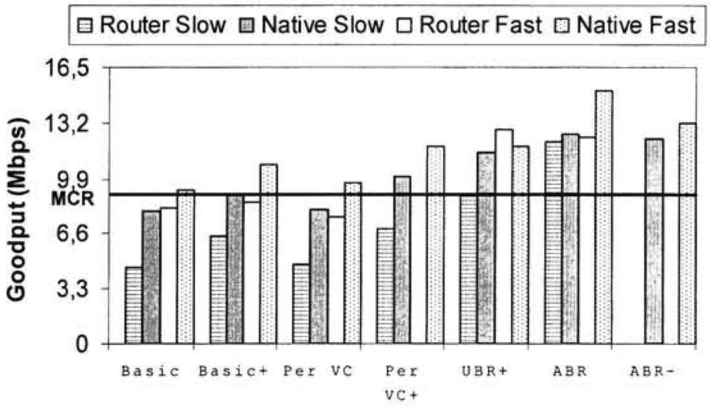


Figure 33 Mean goodput achieved by the large source types

6.4.3 Total cell rate

This subsection compares the different ATCs with respect to the total cell rate achieved by all the TCP/IP sources together. Here the total cell rate is defined as the sum of the cell rates achieved by the individual VCs. As the background traffic consumes an average cell rate of 100000 cells/s, see Table 5, and the total switch capacity is equal to 365000 cells/s, this leaves on average a cell rate of 265000 cell/s available for TCP/IP traffic.

Figure 34 shows to what extent each ATC implementation was able to utilise the available cell rate of 265000 cells/s. Clearly, none of the implementations succeed in doing so completely, although UBR+ comes close. Also evident from Figure 34 is that the ranking of the various setup scenarios with respect to the total achieved cell rate is independent of the ATC implementation. Last in line, with a lowest total achieved cell rate, comes the router slow setup scenario, followed by the native slow and router fast setup scenarios. First in line, with a highest total achieved cell rate, comes the native fast setup scenario. That the slow scenario results in a lower total cell rate than the fast scenario comes as no surprise given that the mean cell rate for the individual VCs was also lower, see section 6.4.1. That the native setup results in a higher total cell rate than the router setup is less obvious but can be explained by the drop in goodput achieved by the large source types, see the previous section. The increase in goodput achieved by the small source types is obviously not sufficient to compensate this.

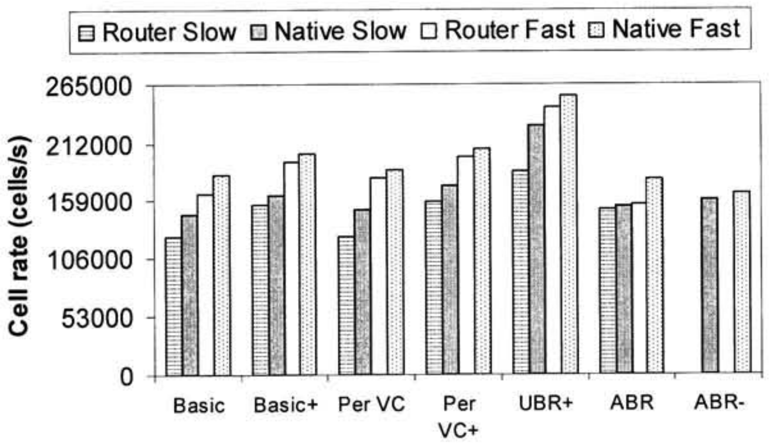


Figure 34 Mean total cell rate achieved by all source types

6.4.4 Fairness

This subsection compares the various ATC implementations on the basis of fairness. Two definitions of fairness are considered, equal fairness and weighted fairness, see section 3.7. For both definitions a fairness index is constructed and subsequently used to compare the performance of the different ATCs. The differences in fairness between the various setup scenarios are also discussed briefly.

For both equal and weighted fairness the following fairness index is defined:

$$\text{Fairness index} = \frac{\text{Achieved cell rate - (MCR + Fair share)}}{\text{MCR + Fair share}},$$

where the definition of what constitutes a fair share depends of the definition of fairness used. The closer the index is to zero, the higher the level of fairness. For equal fairness a fair share is given by

$$\text{Fair share} = \frac{\text{Sum achieved cell rates - Sum MCRs}}{\text{Number of VCs}},$$

whereas for weighted fairness it is given by

$$\text{Fair share} = \frac{\text{MCR}}{\text{Sum MCRs}}(\text{Sum achieved cell rates - Sum MCR}).$$

Here summation refers to the summation over all the individual VCs.

Figure 35 shows the mean value of the both indexes for the three source types in the case of the native slow setup scenario. Clearly, ABR and ABR- tend more towards weighted fairness than they tend towards equal fairness. This is not unexpected given the choice of RIF and RDF parameters, see section 6.2.2. As to the various

GFR implementations and UBR+, these tend neither towards equal fairness nor towards weighted fairness. Clearly, the small source types receive more than their weighted fair share. Of the various GFR implementations, the per VC+ implementation performs only marginally better, much less than one would expect on the basis of the per VC thresholds which are set proportional to the MCR. Worse still, the per VC implementation, which also implements per VC thresholds, shows no improvement whatsoever.

These results are confirmed in Figure 36, which shows the mean value of both indexes for the native fast setup scenario.

Figure 37 and Figure 38 show the indexes of the two VCs in the router slow and router fast setup scenarios respectively. The various ATC implementations now show neither a clear tendency towards equal fairness nor towards weighted fairness. In general VC 2 gets more than it's equal share but less than it's weighted share, and VC 1 vice versa.

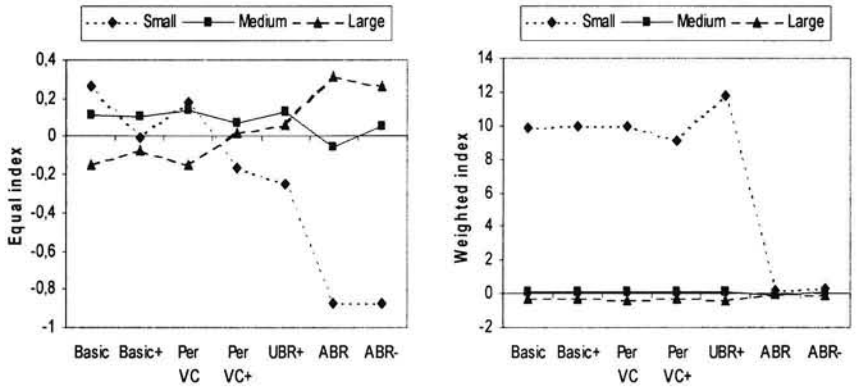


Figure 35 Equal and weighted fairness indexes for the three source types in the native setup and slow background scenario

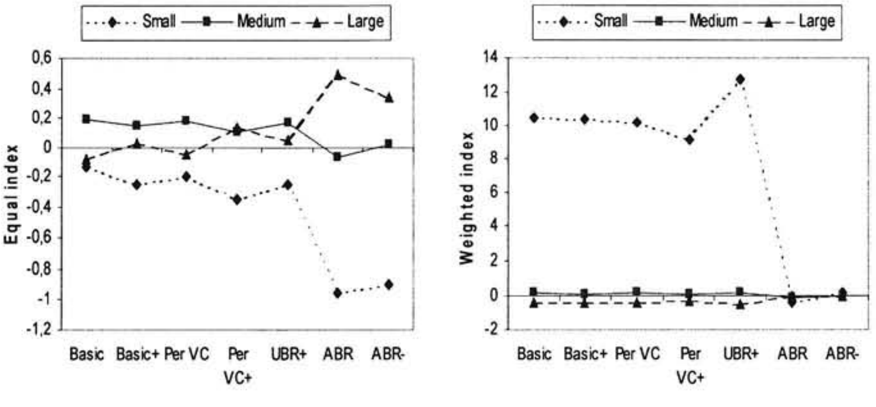


Figure 36 Equal (left) and weighted (right) fairness indexes for three source types in the native setup and fast background scenario

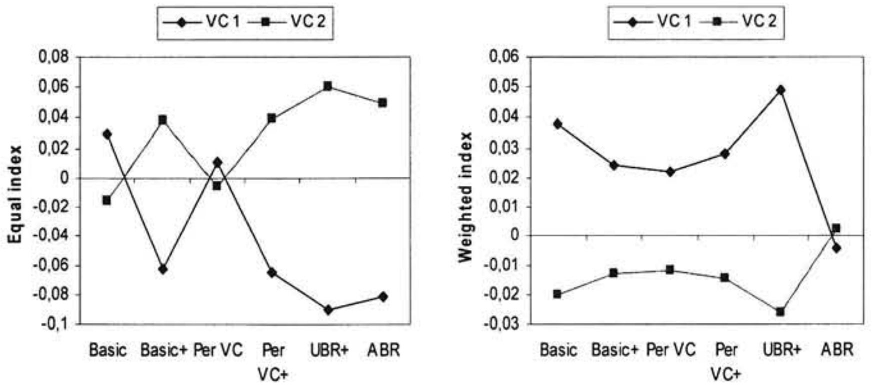
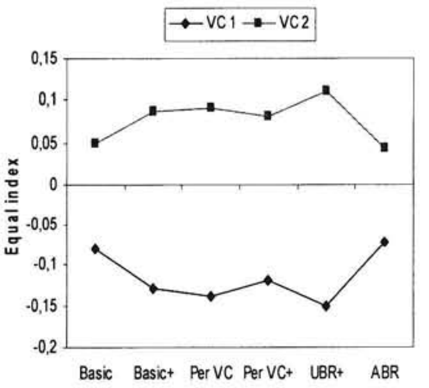


Figure 37 Equal (left) and weighted (right) fairness indexes for the two VCs in the router setup and slow background scenario



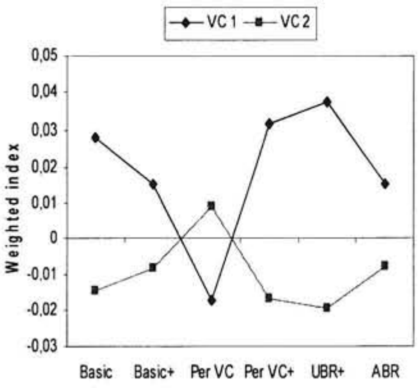


Figure 38 Equal (left) and weighted (right) fairness indexes for the two VCs in the router setup and fast background scenario

Clearly, the above figures illustrate that the value of the fairness index is smaller for the router setup than for the native setup. This is only to be expected as in the native setup there are 23 separate competing VCs, whereas in the router setup there are only 2. The issue of fairness is much less relevant for the router setup than it is for the native setup. The background scenario does not seem to be of significant influence on the fairness index.

6.4.5 Variation

This subsection compares the various ATC implementations on the basis of the variation in the mean goodput achieved by sources of the same source type. Basically, two such identical sources should each achieve an equal goodput and the variation is thus also a measure of fairness. A variation index is introduced and subsequently used as a basis for comparison. The difference between the variation in the various setup scenarios is also briefly discussed.

The variation index is defined as follows:

$$\text{Variation index} = \frac{\alpha_{95}}{\text{Mean achieved cell rate}},$$

where α_{95} denotes the 95% confidence interval for the mean achieved cell rate, see Appendix 1. Ideally, if all identical sources received the same goodput then the variation index would be equal to zero.

Figure 39 shows the value of the variation index for both the native fast and router fast setup scenarios. Clearly, ABR shows the least variation, with GFR lite in the native setup showing the greatest. Figure 40 shows the same behaviour for the slow background scenario.

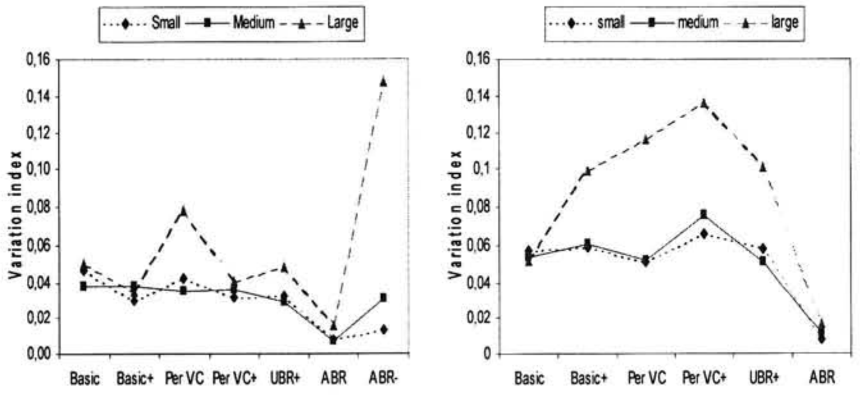


Figure 39 Variation index for native (left) and router(right) setup in the fast background scenario

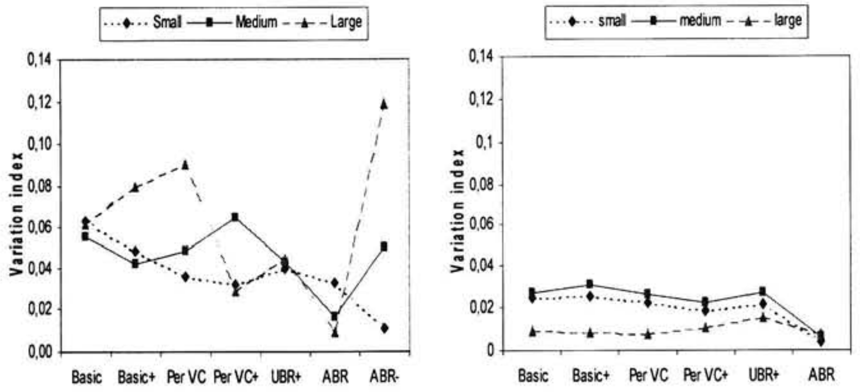


Figure 40 Variation index for native (left) and router(right) setup in the slow background scenario

Clearly, Figure 40 illustrates that the variation in the router setup is smaller than that in the native setup for the case of the slow background scenario. This result is somewhat surprising as, given that the individual sources are all lumped together and treated as equal, one would expect the variation to be larger in the router setup. In the fast background scenario this is indeed the case although only slightly so. A possible explanation for this behaviour is that the TCP traffic cycles of the various sources become synchronised in the router slow setup scenario. This can occur if during a period in which the network is congested each source loses at least two of it's frames. All the sources then initiate a new TCP slow start at approximately the same time. As a result the traffic pattern generated by identical source types is virtually identical, resulting in a smaller variation. For the native slow setup scenario this TCP synchronisation is less likely to occur as each VC is tagged individually implying that the chance of each VC losing at least two frames during a congestion period is smaller.

6.4.6 CLR

This subsection comments briefly on the issue of the cell loss ratio (CLR) for the various ATC simulations. As a maximum CLR in the order of 10^{-6} is the only QoS commitment specified by QoS class 3, a violation of this maximum CLR thus implies a violation of the QoS commitment.

For GFR the EMW lossless CAC algorithm was used and as a result the CLR for cells belonging to QoS eligible frames should be equal to zero, much lower than the upper bound of 10^{-6}. This was indeed the case for all four implementations as only CLP=1 cells were discarded and no CLP=0 cells.

For the GFR lite implementations EWM lossless was also used but the distinction between cells belonging to QoS and non-QoS eligible frames can now no longer be based on the CLP value as all cells have CLP=0. Instead, a formal distinction based on the F-GCRA is required, as in section 2.3.1. However, an alternative way to specify the QoS commitment is to say that the cell rate at which cells are delivered should at least be equal to the MCR, and that all these cells should belong to entire IP packets only. In this respect GFR lite fulfills it's QoS commitment.

For ABR the CAC algorithm used simply reserves a cell rate equal to the MCR for each ABR VC. However, there are now no non-QoS eligible frames and all frames are eligible for QoS commitments. It is up to the ABR congestion control loop to limit the maximum CLR experienced and this can be achieved by tuning the RIF and RDF parameters, see Chapter 3. The CLR experienced was equal to zero for all simulations except the simulation in the native slow setup scenario. The CLR experienced in this simulation was in the order of 10^{-3} which is in violation of the maximum allowed CLR. This indicates that the values of the RIF and RDF parameters chosen based on the ABR simulator, see section 3.6, were not chosen conservatively enough[27]. This result is indicative of the difficulty of tuning the ABR congestion loop and the complexity of providing QoS commitments over ABR.

Note that for ABR the CLR pertains to the total number of cells lost. For GFR and GFR lite, on the other hand, the CLR pertains only to the total number of cells lost belonging to QoS eligible frames. The total number of cells lost for GFR and GFR lite was always higher than the total number lost for ABR.

6.5 Conclusions

This section draws conclusions regarding the suitability of the various ATC implementations simulated for the transfer of TCP/IP traffic. Based on the results discussed in the previous section conclusions are first drawn for the native setup, and

[27] For a discussion on the reasons for this discrepancy between the Excel ABR simulator and the actual simulation of TCP/IP over ABR see Appendix 1.

then for the router setup. These conclusions are made based on the results for both the fast and slow background scenarios combined. The reason for this is that both scenarios represent extremes of the possible background traffic in a real network. The various implementations should therefore be able to perform for both scenarios.

It is important to realise that these conclusions are only valid within the simulation framework used. For example, only a limited number of GFR, ABR and GFR lite implementations were considered while many more are possible, a notable example being per VC scheduled GFR implementations. Also, it was assumed that the TCP version used by each individual TCP/IP source was same. In a general internet environment various TCP versions, for example TCP Vegas, will be in use and this can greatly influence the performance results. Similarly, the same propagation delay was assumed between each TCP/IP source-destination pair. In a general internet this value will fluctuate depending on the physical location of source and destination. Another assumption is that there are no malicious users intent on violating their ATM traffic contract. Certain ATC implementations, for example UBR+, are notably less suited for protecting the network and other non-malicious users from such violators.

6.5.1 Native setup

In this subsection conclusions are drawn about the suitability of the various ATC implementations for transferring TCP/IP traffic in the case of only one TCP session per ATM VC. The implementations are compared with respect to the CLR, the mean achieved cell rate and goodput, as well as the total achieved cell rate, fairness and variation. Table 12 summarises most of these results, using a "+" and "-" to indicate whether a particular implementation performed well or not, and a "0" to indicate that the implementation performed neither good not bad.

Implementation	MCR Guarantee	Goodput	Fairness	Variation	CLR (QoS)
Basic	-	-	-	-	+
Basic+	0	0	-	-	+
Per VC	-	-	-	-	+
Per VC+	+	+	-	-	+
ABR	+	+	+	+	-
UBR+	+	+	-	-	+
ABR-	+	+	+	-	+

Table 12 Summary of performance results for native setup simulations

Of all seven implementations considered only the ABR implementation was unable to fulfill the QoS commitment in terms of a maximum CLR. One way to avoid this QoS violation would be to decrease the RIF and increase the RDF of the ABR congestion control loop. However, this leads to an inefficient use of network resources and does not preclude cell loss from occurring for a different simulation setup. This is due to inherent difficulty of having both the TCP and ABR congestion control mechanism operating simultaneously. As long as no cells are lost, only the ABR congestion control mechanism functions. However, as soon as cell loss does occur, the TCP congestion control mechanism also kicks in. This combination is potentially disastrous if both mechanisms are out of phase. To explain this, note that due to the TCP slow start algorithm a TCP/IP source does not behave as a greedy source from the viewpoint of the ATM network. A slow start decreases the volume of traffic generated by a source suddenly. This implies that the ACR of the source will increase linearly when the number of frames in the network drops below the per VC congestion threshold value. The traffic volume, in the meantime, increases exponentially implying that the network will be flooded by a sudden surge of frames. This leads to more cell loss and new TCP slow starts, resulting in a vicious cycle.

The only implementations that failed to utilise the cell rate equal to MCR reserved for them, were the basic and per VC GFR implementations. The reason for this failure lies in the TCP congestion control mechanism which is triggered after frame discard has occurred. The TCP FRR algorithm efficiently retransmits frames but if it fails then no new frames are sent until the timeout of the retransmission timer. This implies that the TCP/IP source is idle for a time equal to the TCP round trip time. After retransmission a new TCP slow start is initiated and it takes a while for the volume of traffic to build up again. The relatively long retransmission time combined with the TCP slow start mechanism greatly reduce the number of frames sent into the network. The reason the basic and per VC GFR implementations fail to utilise the reserved MCR while the other implementations succeed, is due to the greater number of retransmission timeouts which apparently occur for these two implementations.

TCP retransmission timeouts can also lead to duplicate frames arriving at the destination. These duplicate frames represent a waste of the cell rate used to transfer them and lead to a discrepancy between the achieved cell rate, an ATM performance parameter, and the achieved goodput, a TCP performance parameter. However, there were no implementations that achieved a cell rate in excess of the MCR but failed to achieve a goodput in excess of the MCR goodput. The basic+ GFR implementation, however, succeeded in doing so only barely.

The UBR+ GFR lite implementation succeeded best in utilising the cell rate available in the network. Compared to the other GFR, FIFO implementations, this was because the UBR+ implementation used the switch buffers more efficiently. However, if a less conservative CAC algorithm than EMW lossless is used, then this difference should become less pronounced. The reason for this is that, for the buffer setup shown in Figure 21, a less conservative CAC algorithm will admit a larger number of VCs than those listed in source sets 1 and 2, see Table 7. For GFR this implies that the buffer space above the EPD threshold can be used more efficiently. For UBR+, however, a less conservative CAC algorithm has little to no effect on the buffer usage. However, a less conservative CAC algorithm could have a detrimental effect on the CLR and the cell rate achieved as more VCs compete for the same resources. The ABR and ABR- GFR implementations succeed least in utilising the available cell rate. This is due to the conservative ABR congestion loop which severely restricts the ACR.

Only the ABR and ABR- GFR lite implementation are fair in the sense of weighted fairness. All the other FIFO based implementations fail in sharing the available cell rate amongst the VCs in a fair way, either weighted or equal.

The ABR implementation shows the least variation in goodput achieved by identical sources. Surprisingly, the ABR- GFR lite implementation shows the greatest variation. This can be explained by the location at which cell loss occurs. For all the implementations other than ABR-, cell loss only occurs at switch 3, see Figure 20. For ABR-, on the other hand, cell loss mainly occurs at the boundary switches, switches 1 and 2. Given that the number of sources in source set 1 is unequal to the number of sources in source set 2, and that the network resources available in switch 1 and switch 2 are equal, the amount of competition between the sources in both source sets differs. For a given source in source set 2 therefore, cell loss will occur in switch 2 more often than it will occur in switch 1 for a source of the same source type in set 1. This results in a greater variation in the achieved goodput between sources of the same source type.

In conclusion, the basic and per VC GFR implementations are not suitable for the transfer of TCP/IP traffic as they cannot provide the required MCR rate guarantees. The ABR implementation would be ideal if no cell loss occurred but this can only be avoided if the ABR congestion control loop is very conservative, resulting in an inefficient use of network resources. Besides, if cell loss does occur then the double TCP and ABR congestion control mechanisms can cause the QoS commitments to be violated. In all, ABR is difficult to use for the transfer of TCP/IP traffic.

More suited is the ABR- GFR lite implementation, especially if fairness is important. In section 6.2.3 it was argued that the performance of ABR- could be considered indicative of the performance of a GFR implementation using per VC scheduling. Such a GFR implementation should therefore also be suitable. However, in section 2.3 it was already concluded that the GFR CLP bit distinction is of no use in the case of only one TCP session per ATM VC. GFR lite does not have this distinction and as a consequence is also easier to implement in the network than GFR.

If fairness is not an issue then the UBR+ GFR lite implementation is most suited due to the efficient use of network resources. Of the two remaining GFR implementations the per VC+ implementation performs better. Again, the required CLP distinction unnecessarily complicates the GFR implementations.

6.5.2 Router setup

This subsection draws conclusions about the suitability of the various ATC implementations for transferring TCP/IP traffic in the case of multiple TCP session per ATM VC. The implementations are compared with respect to the CLR, the mean achieved cell rate and goodput, as well as the total achieved cell rate, fairness and variation. Table 13 summarises most of these results, using a "+" and "-" to indicate whether a particular implementation performed well or not, and a "0" to indicate that the implementation performed neither good not bad.

Implementation	MCR Guarantee	Goodput	Fairness	Variation	CLR (QoS)
Basic	0	-	-	-	+
Basic+	+	-	-	-	+
Per VC	0	-	-	-	+
Per VC+	+	-	-	-	+
ABR	+	0	-	+	+
UBR+	+	+	-	-	+

Table 13 Summary of performance results for router setup simulations

Both VCs in the router setup were able to fulfill the requested QoS commitments in terms of a maximum CLR for all of the implementations considered. Likewise, both VCs succeed in achieving a mean cell rate in excess of the reserved cell rate equal to the MCR in all of the implementations. With respect to the goodput achieved by the individual sources transferred over these two VCs, only the UBR+ GFR lite implementation succeeded in always achieving a goodput in excess of the MCR goodput as defined for the native setup. This implementation also succeeded best in utilising the cell rate available in the network. Of the four GFR implementations, per VC+ performed the best in this regard.

With respect to fairness there is no clear reason to prefer one implementation over another. With respect to the variation of goodput between sources of the same source type, the ABR implementation clearly performs the best. This implementation also showed the least variation in the performance results for the slow and fast background scenarios.

In conclusion, in contrast to the native setup, all the ATC implementations considered are suitable for the transfer of TCP/IP traffic. The UBR+ GFR lite implementation, however, is preferable due to the simple implementation and the efficient use of network resources. More complex implementations using ABR congestion control mechanisms or per VC thresholds provide little added benefit. The reason for this is that there are relatively few competing VCs in the router setup due relatively large amounts of network resources reserved by each VC.

The per VC+ GFR implementation performs the best of all GFR implementations. This implementation has the added advantage over the UBR+ GFR lite implementation of the CLP distinction. In the case of multiple TCP sessions over the same ATM VC the CLP bit can be used to some advantage, as concluded earlier in section 2.3.

Chapter 7 Conclusions & Rec ommendations

7.1 Conclusions

Of all of the current ATM transfer capabilities (ATCs), the Guaranteed Frame Rate (GFR) and Available Bit Rate (ABR) ATCs are the most suited for the transfer of TCP/IP traffic. Chapter 2 reaches this conclusion after making a qualitative comparison of the current ATCs on the basis of the formal ITU-T standard[17][28]. As a result, the Deterministic Bit Rate (DBR), Statistical Bit Rate (SBR) and Unspecified Bit Rate (UBR) ATCs are all deemed less suited for the task of transferring TCP/IP traffic.

A further comparison of the GFR and ABR ATCs requires the performance of TCP/IP traffic over both ATCs to be quantified. This is done in Chapter 6 where TCP/IP traffic is simulated over both ATCs and the simulation results are compared. In addition, a simplified variant of the GFR ATC, not using the cell loss priority (CLP) bit, is proposed under the name GFR lite and is also simulated. In order to be able to perform the simulations, choices were made as to how the ATCs should be implemented. Such choices are necessary because the formal ITU-T standard does not specify any specific implementation and there are many compliant implementations possible. Of the possible implementations four GFR implementation were considered, all based on a first-in-first-out (FIFO) buffer in combination with a frame discard mechanism, one ABR implementation was considered, based on the Ascend CBX 500 ABR implementation, and two GFR lite implementations were considered, one based on a FIFO buffer and the other on the Ascend ABR implementation

A general conclusion that can be drawn from the simulation results is that more than the ATC it is the implementation of the ATC that determines the actual performance. This is most clearly illustrated by the four GFR implementations considered. Of these four, only two implementations enable a VC to always succeed in achieving a TCP goodput in excess of what one would expect based on the cell rate reserved in the network for that VC. The effect the implementation has on the performance is also illustrated in Table 14 which clearly shows the trade-off that exists between the network efficiency and the fairness, along with the complexity of the implementation required. Increasing the fairness of a given ATC requires a more complex

[28] GFR is currently still being standardised.

implementation of that ATC and this generally implies a less efficient usage of network resources.

Implementation	Efficiency	Fairness	Complexity
UBR+ (GFR lite)	↑	↓	↓
Basic (GFR)			
Per VC (GFR)			
Basic+ (GFR)			
Per VC+ (GFR)			
ABR- (GFR)			
ABR			

Table 14 Trade-off between efficiency and fairness, along with the required implementation complexity.

A more specific conclusion that can be drawn from the simulation results is that, within the framework of the simulation setup used and the ATC implementations considered, the GFR ATC and GFR lite variant are preferable to the ABR ATC. There are a number of reasons for this. Firstly, ABR is not always able to fulfill it's quality of service (QoS) commitment in terms of cell loss. Of course, this may be due to the somewhat primitive ABR implementation considered. Secondly, GFR and GFR lite, especially the UBR+ variant, succeed in using the available network resources more efficiently than ABR does. The only advantage ABR has is that it is fair, sharing the available network resources amongst competing virtual connections (VCs) proportional to the cell rate reserved by each VC. However, the ABR based GFR lite implementation also has this advantage and it can be expected that a GFR implementation based on per VC scheduling would also be comparably fair. Finally, GFR and especially GFR lite do not require an 'intelligent' ATM network interface card (NIC) as a NIC which marks all cells as CLP=0 suffices. In contrast, ABR requires the NIC to be ABR compatible, implying that the NIC is required to be capable of generating resource management (RM) cells and adapting it's output cell rate. GFR and GFR lite are thus ideally suited for old, 'legacy' NICs and do not require a router or TCP/IP source to update it's NIC.

7.2 Recommendations

- The GFR1 variant of the GFR ATC should be implemented in operational ATM networks and used for the transfer of TCP/IP traffic between IP routers. The GFR2 variant is less suited for this purpose as it does enable the router to mark the traffic it considers important as CLP=0.
- A simplified version of the GFR1 variant of the GFR ATC, not using the CLP bit, should be standardised and used for the transfer of TCP/IP traffic between TCP/IP end systems. This report proposes such a variant under the name GFR lite.
- No more effort should be made to further standardise the GFR2 variant of GFR.

Summary

Of the traffic currently transferred over the Internet, the largest part is generated by sources which implement the transmission control and internet protocols, TCP and IP for short. This report investigates how such traffic can best be transferred over a network based on the asynchronous transfer mode (ATM). Such a network differs from a network based on IP in numerous ways, most notably in being able to provide quality of service (QoS) guarantees.

There are a variety of ATM transfer capabilities (ATCs) which could be used to transfer TCP/IP traffic over an ATM network. Of these, this report concludes that the Available Bit Rate (ABR) ATC and the new Guaranteed Frame Rate (GFR) ATC promise to be best suited for the transfer of TCP/IP traffic.

ABR is unique as an ATC in that it uses a congestion control mechanism in order to provide QoS guarantees. This report investigates an example of such a mechanism and it's effects on the transfer of TCP/IP traffic. Of special interest is the interaction between the ABR and TCP congestion control mechanisms.

GFR is unique as an ATC in that it specifically designed for the transfer of frame based traffic such as IP. GFR provides it's QoS guarantees by means of a connection admission control (CAC) algorithm. This report compares two such algorithms commonly used in operational networks in order to determine which is preferable.

In addition, this report introduces a variant of the GFR ATC under the name GFR lite. This non-standardised ATC variant has the advantage of being simpler than GFR without losing it's suitability for TCP/IP traffic.

In order to quantify the performance of TCP/IP traffic over these three ATCs, a performance analysis is made based on a simplified network model. To quantify the expected performance in a real network environment simulations are performed.

Based on the simulation results obtained, this report concludes that both GFR and GFR lite are best suited for the transfer of TCP/IP traffic. The ABR ATC is considered less suited because it is difficult to both efficiently use the available network resources and to provide the required QoS guarantees.

List of Abbreviations

AAL	ATM adaptation layer
ABR	Available bit rate
ACK	Acknowledgement
ACR	Allowed cell rate
ATC	ATM transfer capability
ATM	Asynchronous transfer mode
CAC	Connection admission control
CDV	Cell delay variance
CLP	Cell loss priority
CLR	Cell loss ratio
DBR	Deterministic bit rate
EPD	Early packet discard
F-GCRA	Frame based GCRA
FIFO	First in first out
FRR	Fast recover and retransmit
GCRA	Generic cell rate algorithm
GFR	Guaranteed frame rate
ICR	Initial cell rate
IP	Internet protocol
ITU	International telecommunication union
ITU-T	Telecommunication standardisation sector of the ITU
LCR	Link cell rate
Mbps	Mega bits per second
MCR	Minimum cell rate
MFS	Maximum frame size
MSS	Maximum segment size
PCR	Peak cell rate
QoS	Quality of service
RDF	Rate decrease factor
RIF	Rate increase factor
RM	Resource management
SBR	Statistical bit rate
SCR	Sustainable cell rate
TCP	Transmission control protocol
UBR	Unspecified bit rate
VC	Virtual connection

References

[1] Anick D., Mitra D., and Sondhi M.M. , "Stochastic theory of a data-handling system with multiple sources", The Bell System technical journal, October 1982, page 1871.

[2] ATM Forum specification, "Traffic management specification, version 4.1", af-tm-0121.00, 1999, http://www.atmforum.com.

[3] Bonaventure O. and Nelissen J., "Guaranteed frame rate: a better service for TCP/IP in ATM networks", http://www.info.fundp.ac.be/~obo.

[4] Bonaventure O., "A Simulation study of TCP with the proposed GFR service category", Daghstuhl seminar on high performance networks for multimedia applications, June 1997, http://www.info.fundp.ac.be/~obo.

[5] Bonomi F.,Mitra D., and Seery J.B., "Adaptive algorithms for feedback-based flow control in high speed, wide-area ATM networks", IEEE journal on selected areas in communications, vol. 13, no. 7, september 1995, page 1.

[6] Brakmo L., O'Malley S. and Peterson L., "TCP Vegas: New Techniques for Congestion Detection and Avoidance", Proceedings of the SIGCOMM '94 Symposium, August 1994, pg. 24-35.

[7] Du, D.H.C., Hsieh, J., Lin, M. and Thomas, J.P., "Distributed network computing over local ATM networks", http://www.eantc.de/Documents/OtherPapers/IPoverATM

[8] Elloumi O. and Afifi H., "Evaluation of FIFO-based buffer management algorithms for TCP over guaranteed frame rate service", Proceedings IEEE ATM98 workshop, Fairfax, USA, May 1998.

[9] Elwalid A., Mitra D., and Wentworth R.H., "A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node", IEEE journal on selected areas in communications, volume 13, no. 6, august 1995, page 1115.

[10] Ewy, B.J., Evans, J.B., Frost, V.S. and Minden G.J., "TCP/ATM experiences in the MAGIC testbed", University of Kansas, http://www.ittc.ukans.edu/Projects/AAI/KU_TCP_over_ATM_Papers.

[11] Goyal R., Jain R., Fahmy S. and Vandalore B., "Buffer management for the GFR service", ATM Forum contribution 98-0405, July 1998, http://www.cis.ohio-state.eud/~jain

[12] Goyal R., Jain R., Fahmy S. and Vandalore B., "Providing rate guarantees to TCP over the ATM GFR service", Proceedings of 23rd Annual Conference on Local Computer Networks 1998 (LCN'98), Lowell, MA, October 11-14, 1998, pp. 390-398, http://www.cis.ohio-state.eud/~jain

[13] Guerin R. and Heinanen J., "UBR+ service category definition", ATM Forum contribution ATM96-1598, December 1996, http://www.atmforum.com

[14] Guerin R., Ahmadi H., and Naghshineh M., "Equivalent capacity and its application to bandwidth allocation in high-speed networks", IEEE journal on selected areas in communications, volume 9, no. 7, September 1991, page 968.

[15] Heinanen J., "Multiprotocol Encapsulation over ATM Adaptation layer 5", RFC 1483, Telecom Finland, July 1993, http://www.ietf.org

[16] ITU-T recommendation I.356, "B-ISDN ATM layer cell transfer performance", 1996.

[17] ITU-T recommendation I.371, "Traffic control and congestion control in B-ISDN", 1999.

[18] Kleijnen, J.P.C., "Statistical techniques in simulation", parts I and II, Marcel Dekker Inc, 1975.

[19] Lakshman T.V. and Madhow U., "Performance of window-based flow control using TCP/IP: Effect of high bandwidth-delay products and random loss", High performance networking, V (C-26), Elsevier science B.V., 1994.

[20] Li H., Siu K.Y., Tzeng H.Y., Ikeda C. and Suzuki H., "A simulation study of TCP performance in ATM networks with ABR and UBR services", INFOCOM 1996, page 1269.

[21] Manthorpe S., "Implications of the transport layer for network dimensioning", PhD thesis, 1997, ftp://lrcftp.epfl.ch/pub/people/manthorpe/thesis.ps

[22] Pappu S.K. and Basak D., "TCP over GFR implementation with different service disciplines: a simulation study", ATM Forum contribution 97-0310, 1997, http://www.atmforum.com.

[23] Prycker, M. de, "Asynchronous transfer mode. Solution for broadband ISDN", Prentice Hall Europe, 1995.

[24] Ren W., Siu K.Y., Suzuki H. and Ramamurthy G., "Performance of TCP in AP/ATM internetworks", Computer Communications 21, 1998, page 1610.

[25] Stallings W., "High-speed networks, TCP/IP and ATM design principles", section 10.3, Prentice-Hall Inc., 1998.

[26] Stevens W.R., "TCP/IP Illustrated, Volume 1", Addison-Wesley publishing company, 1994.

[27] Tanenbaum A.S., "Computer Networks", Prentice Hall Inc., 1996.

[28] Wright G.R. and Stevens W.R., "TCP/IP illustrated, Volume 2", Addison-Wesley publishing company, 1995.

Appendix 1

In order to simulate TCP/IP traffic over an ATM network a modified version of the STCP simulator developed by Sam Manthorpe at the Laboratoire de Réseaux de Communication was used. STCP is an event-driven, queue based simulator based on the original TCP code found in BSD 4.4 Lite, also known as TCP-lite, see [28]. All important TCP mechanisms such as slow-start, congestion avoidance and fast retransmit and recover are supported. The source code of STCP can be freely downloaded from
http://lrcwww.epfl.ch/~manthorp/stcp/.

The modifications made to STCP source code are listed below:

- Frame based tagging(F-GCRA) was added for the GFR simulations.
- A CLP=1 EPD mechanism was added for the GFR simulations.
- Per VC accounting in the switches was added for the GFR simulations.
- IP router emulation was added for the router setup.
- The ABR congestion control mechanism implemented in the Ascend CBX 500 was added for the ABR simulations.

Verification

As a first step the above modifications were fully tested to ascertain that each performed as expected. As a second step the entire simulation package was tested in order to verify that it worked as expected. Two different methods were used to achieve this:

- **Theoretical**

Simulation results for a steady state network were compared with the theoretical results for a steady state network obtained in Chapter 4. Similarly, the simulated behaviour of the ABR congestion control loop was compared with the behaviour expected from the analysis in Chapter 3. In the first case the simulation results were shown to closely match the theoretically predicted results, as Figure 41 clearly illustrates. However, in the latter case discrepancies occur as can be clearly seen from Figure 42.

Initially, the simulated behaviour of the ACR in Figure 42 is identical to the theoretical behaviour. However, after a short while the behaviour begins to deviate. This is not unexpected, however, for the following reasons:
The theoretical analysis in Chapter 3 assumes that at the moment that the switch is ready to update the ACR, it also receives an RM cell informing it whether or not it may increase the ACR. In a real operational network this is an unrealistic assumption as an RM cell may arrive at a switch at any time and trigger an immediate ACR update. As a result the period between subsequent ACR updates fluctuates. This is clearly illustrated in the above figure where the deviation first occurs because in the

simulations the ACR starts increasing again earlier than it does in the theoretical analysis.
The theoretical analysis is based on the assumption of a greedy ABR source which always sends as much data as the ACR allows. However, from the viewpoint of the ATM network a TCP/IP source is far from greedy as the TCP congestion control mechanism can lower the flow of TCP/IP traffic to below that allowed by the ACR. Due to these differences, discrepancies in the behaviour of the ACR occur and propagate in time, resulting in widely differing behaviour. The discrepancies between the theoretical and simulated ACR behaviour therefore do not indicate that the STCP simulator is incorrect. On the contrary, the fact that the behaviour is initially identical and that the subsequent behaviour is comparable is taken to indicate that the STCP simulator functions correctly.

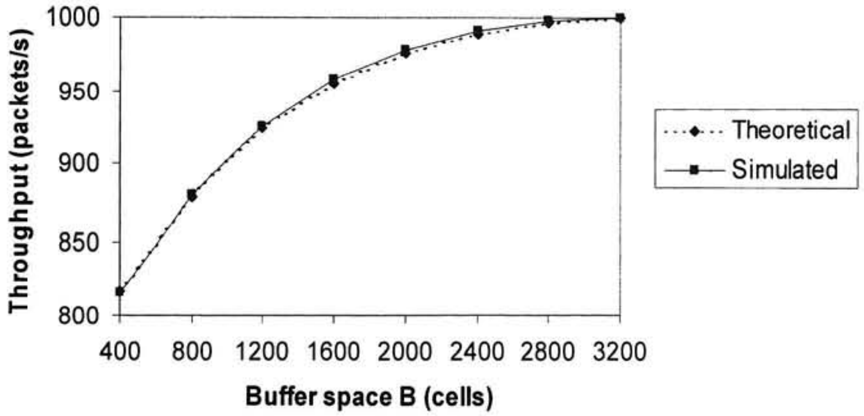


Figure 41 Theoretical and simulated goodput in steady state model

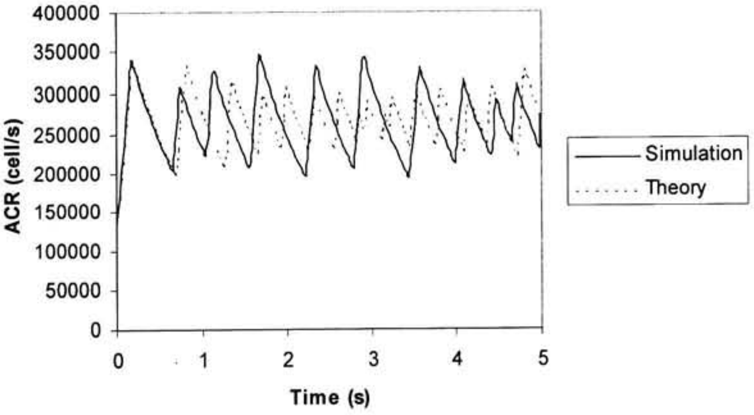


Figure 42 Simulated and theoretical behaviour of the ACR as function of time

- **Simulation**
  The following TCP/IP over ATM simulations described in literature were reconstructed and simulated, allowing the obtained results to be compared:

- 'A simulation study of TCP with the proposed GFR service category', Olivier Bonaventure, http://www.info.fundp.ac.be/~obo/biblio.html. In this article a similarly modified version of STCP is also used. Equivalent results were obtained to those listed in the article.
- 'Evaluation of FIFO-based buffer management algorithms for TCP over guaranteed frame rate service', Omar Elloumi and Hossam Afifi, http://www.rennes.enst-bretagne.fr/~elloumi/. In this article the simulation package that was used is not listed but, as different TCP versions are used than TCP-Lite (TCP Reno and TCP SACK), it is safe to assume STCP was not used. The GFR-FIFO simulations were rerun and comparable results were obtained.
- 'Providing rate guarantees to TCP over the ATM GFR service', Rohit Goyal, Raj Jain, Sonia Fahmy, and Bobby Vandalore, http://www.cis.ohio-state.edu/~jain/papers.html. In this article the simulation package used is not listed but as TCP SACK is implemented it is safe to assume that STCP was not used. The per VC threshold simulations were rerun and comparable results were obtained.

Batch simulation

STCP allows simulations to be divided into batches. A batch can be considered a sub-simulation of the same model, but with a different sequence of pseudo-random numbers. The state of the simulation model is reset at the start of each batch. Dividing a simulation into batches allows one to estimate the batch mean and the batch variance of most measurements, which in turn allows confidence intervals to be constructed.

Consider a sequence of M measurements, X_m (m=0,..,M-1), of the same variable in the model. If these measurements are divided into N batches of size b_n *(*n=0,..,N-1) then let Y_n denote the average of X within batch n. To calculate an estimate of the variable X and the confidence intervals, the sample mean <Y> and an estimation of the variance of Y, s_Y^2, are calculated as follows:

$$< Y >= \frac{1}{N}\sum_{n=0}^{N-1} Y_n \,,$$

$$s_Y^2 = \frac{1}{N-1}\sum_{n=0}^{N-1} \left(Y_n - < Y >\right)^2 \,.$$

Confidence intervals can then be constructed using <Y> $\pm \alpha_k$, with $\alpha_k = k\ s_Y$ and k is the appropriate ordinate from the t-distribution with N-1 degrees of freedom. In this report only the 95% confidence intervals (α_{95}) are used. For more details on batch simulations see for example [18].

Appendix 2

The ABR congestion control mechanism implemented in the Ascend CBX 500 ATM switch differs somewhat from the standard as defined in either [17] or [2]. The binary feedback mechanism is based on either BCM or CCRM cells, both proprietary RM cells. By means of these cells each VC can establish it's own ABR congestion control loop between any two given Ascend switches in the network. Both BCM and CCRM are backwards RM cells. The Ascend congestion loop does not use forward RM cells.

The CCRM cells are generated periodically by a switch and the congestion indication (CI) bit is marked in case of switch congestion. The BCM cells, on the other hand, are only generated periodically during periods that a switch is congested and thus indicate congestion by default. The interval between subsequent CCRM or BCM cells will be denoted by T_{RM}.

Upon receiving either a CCRM cell with it's CI bit marked or a BCM cell, a switch must lower the allowed cell rate (ACR) for the VC to which the cells belong. Otherwise, the ACR may be increased. The ACR update rules are as follows:

$$\mathrm{ACR}_{n} = \mathrm{ACR}_{n\text{-}1}\,(1 - \mathrm{RDF}),$$

$$\mathrm{ACR}_{n} = \mathrm{ACR}_{n\text{-}1} + \mathrm{RIF} \times \mathrm{PCR}\ ,$$

where the index *n* numbers the subsequent updates.

All ABR VCs share a common buffer but are served as if each VC had it's own private buffer, cells being emptied from this buffer at the ACR. Each VC also has it's own per VC congestion threshold which is used in combination with the global congestion threshold to determine whether a particular VC is congested. A particular VC is defined as being congested if both the total number of cells in the buffer exceeds the global congestion threshold, and the number of cells belonging to this VC exceeds the per VC threshold.

The CBX 500 also implements a throttle queue. Cells served from the common ABR buffer are first placed in this queue which in turn is served by FIFO scheduler. The rate at which the throttle queue is emptied depends on the volume of the higher priority, non-ABR background traffic. The throttle queue implements a throttle threshold and while number of cells in the throttle queue exceeds this threshold then the common ABR buffer is not emptied. In addition, if the ACR of a particular VC is above the "fair bandwidth" it is decreased. Here the "fair bandwidth" for a given VC is defined by:

$$Fair_i = \frac{MCR_i}{\sum_j MCR_j}(LCR - BCR),$$

where BCR is short for the background cell rate and denotes the cell rate reserved for the background traffic, and the index *i* denotes VC_i.

In addition, an idle VC factor is also defined. This factor determines the maximum number of T_{RM} intervals a VC is allowed to send nothing before being declared 'idle'. Once declared 'idle', the ACR of the VC is lowered to the initial cell rate (ICR). The idle VC factor protects the network from a source which is idle for a relatively long period, during which the ACR increases, and which then suddenly floods the network with traffic. An example of such a source is a TCP/IP source; the TCP slow start algorithm generates an exponential flood of traffic straight after the relatively long idle period it takes for the retransmission timer to expire.

The Department of Mathematics and Computer Science at Delft University of Technology offers a selected group of people a two-year post-Master's program featuring specialized instruction in practical mathematical modeling and consultancy skills (in Dutch: Wiskundige Beheers- en Beleidsmodellen). Participants follow courses in the first year of the program and apply their skills in a project in the second year. Projects are carried out under contract with a company or independent research institute. The WBBM Report Series contains the final reports of these projects.

TU Delft

Delft University of Technology
Wiskundige Beheers- en Beleidsmodellen

ISBN 90-407-1968-3