1996

Congestion control of gateways used for LAN interconnection

Tippu Hassan
University of Wollongong

UNIVERSITY OF WOLLONGONG
COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author.

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Recommended Citation

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
A Thesis entitled

Congestion Control of Gateways Used for LAN Interconnection

Submitted to the
Department of Electrical and Computer Engineering
University of Wollongong
in fulfilment of the requirements for the degree of
Masters (Hons)

Tippu Hassan
Bachelor of (Electrical) Engineering (Honours)
University of Wollongong 1992

Date of submission: October 1996
Abstract

The need to support distributed applications and the availability of low cost Local Area Network (LAN) equipment has created a profusion of LANs. As the number of LANs increases, the performance of the gateways that are needed to interconnect them become crucial.

Congestion in LAN gateways can lead to a degradation in the quality of service. A literature review of schemes proposed to improve gateway performance has revealed deficiencies which need to be addressed.

Random Early Discard (RED) of packets from the gateway buffer has been proposed in literature as one of the schemes which can improve gateway performance. In this thesis, the performance of this scheme is thoroughly analysed by using a queuing network model. It is shown that contrary to what is claimed in literature, this scheme does not result in any significant improvement in gateway performance.

A novel scheme called "the intelligent packet discard" is proposed in this thesis. It is shown that the intelligent packet discard scheme results in much improved gateway performance.
Acknowledgments

I would like to thank my supervisors Associate Professor Joe Chi­charo and Professor Gary Anido for their support, guidance and encouragement throughout this project.

I am very grateful to AWADI for the financial support. I would like to thank the staff at AWADI, North Ryde, for their interest and the software team for advice in the development of Frame Relay and ATM driver software.

I also thank everyone at the Switched Networks Research Centre which provided a great environment for research work.

Most importantly, I would like to thank my parents for their sacrifice and support.

Tippu Hassan
Abstract

Statement of Originality

Acknowledgments

List of Abbreviations vi

1. Introduction 1
   1.1 Congestion Control In Local Area Network Gateways 1
   1.2 AWANET 100 2
   1.3 Statement of Problem 5
   1.4 Thesis Outline 6
   1.5 Contributions 7
   1.6 Publications 9

2. Literature Review 10
   2.1 Outline 10
   2.2 Congestion in LAN Gateways 11
   2.3 Congestion Detection and Notification. 11
   2.4 Congestion Avoidance and Control 18
   2.5 Deficiencies and Developments. 31

3. Analysis of LAN Gateways 34
   3.1 Introduction 34
   3.2 Interconnection of LANs using gateways 35
   3.3 Congestion in the IP Gateway 38
   3.4 Modelling the LAN 39
   3.5 Conclusion 44

4. Congestion Control Using Random Early Packet Discard. 46
   4.1 Introduction 46
   4.2 The RED Gateway 48
   4.3 Non-Retransmitting Sources and RED Gateways 51
   4.4 Analysis of Non-Retransmitting Sources 54
   4.5 Retransmitting Sources and RED Gateways 58
   4.6 Analysis of Re-transmitting Sources 61
   4.7 Conclusion 70

5. Intelligent Packet Discard 72
   5.1 Introduction 72
   5.2 Data Transmission Using TCP/IP 73
   5.3 IP Fragmentation 79
   5.4 Intelligent Packet Discard Gateway 81
   5.5 Analysis of Results 84
List of Abbreviations

AAL  ATM Adaptation Layer
ATM  Asynchronous Transfer Mode
BECN Backward Early Congestion Notification
CCITT International Telephone and Telegraph Consultative Committee (now the ITU-T)
CIR  Committed Information Rate
CLLM Consolidate Link Layer Management
DE   Discard Eligibility
DLCI Data Link Channel Identifier
FECN Forward Early Congestion Notification
FDDI-II Fibre Distributed Data Interface-II
IP   Internet Protocol
ISDN Integrated Services Digital Network
Kb/s kilobits per second
LAN  Local Area Network
MAC  Medium Access Control
MTU  Maximum Transfer Unit
Mb/s Megabits per second
PC   Personal Computer
PDU  Protocol Data Unit
RAM  Random Access Memory
RED  Random Early Discard
SAR  Segmentation And Reassembly
TCP  Transmission Control Protocol
TDM  Time Division Multiplex
WAN  Wide Area Network
1. Introduction

1.1 Congestion Control In Local Area Network Gateways

This Masters thesis involves the study of congestion in Internet Protocol (IP) gateways which are used to interconnect Local Area Networks (LANs). The aim of this thesis is to:

1. Investigate the efficacy of current gateway congestion control schemes and to establish the merits of various schemes available in current literature.

2. Develop an intelligent packet discard scheme for improving the performance of an IP gateway without having to modify the existing data transmission protocols.

The ever growing importance of LANs as media for data communications has lead to an increased connectivity of the LANs and given rise to Wide Area Networks (WANs). The need to support distributed applications such as databases has lead to large scale use of gateways for LAN interconnectivity. The most widely used data transmission protocols are TCP at the transport level and IP at the network level. As a result, IP gateways are in wide spread use. Chapter 3 will discuss in detail the benefits of interconnecting LANs at the network (IP) level.
Figure 1.1 shows a typical WAN. The most common gateway in use is an IP gateway i.e. the gateway can only utilise the information available in the IP header of the passing packet and all higher layers (such as TCP) are hidden as a result of data encapsulation. The network shown in Figure 1.1 could use either a connection-less or a connection-oriented protocol for data transmission between LANs.

As an example of a typical LAN, the AWANET 100 will be described briefly in the following Section (The reason for the selection of AWANET 100 to represent a typical LAN is that work was carried out, as part of this masters program, on the development of AWANET 100).

1.2 AWANET 100

AWANET 100 is an FDDI II LAN which has the ability to carry ATM traffic. Each of the nodes connected to an AWANET LAN can be a small LAN in itself that connects to the Frame Relay network using one or more IP gateways.
The traffic sources/workstations in the AWANET 100 LAN shown in Figure 1.2 use TCP/IP for data transmission. Each source is connected to an Ethernet and a Time Division Multiplex (TDM) backplane. Whenever data is to be transmitted to a remote node, it is passed on to the FDDI II ring by means of one of the two backplanes. The data is then carried as part of an FDDI II packet and is passed on to the appropriate node for transmission. Any IP address that cannot be located on a local AWANET-100 LAN is passed to the gateway node for transmission to the remote LAN. At the destination LAN, the data packet is passed on to the user via either of the two backplanes mentioned earlier (i.e., the ethernet and the TDM backplanes).

Each of the transmission protocols used in this transfer of data attaches its own appropriate header while passing data down the protocol stack and removes it when passing it onto the higher layer. The receiving node accepts the packets off the backplanes and strips
the headers. This means that the packets passed on to the destination node (or gateway node for data meant for remote LANs) are IP datagrams. As will be discussed further in Chapter 5, this IP datagram might in turn be part of a larger IP datagram which has been fragmented by IP due to the different physical mediums and Maximum Transfer Units (MTUs) involved.

All data meant for a user on a remote LAN, ends up at the IP gateway. As previously mentioned, the gateways provide LAN interconnectivity at the network level. In the case of AWANET 100, the incoming IP datagrams are encapsulated in frames and transported across a Frame Relay network. It should be noted that it is not imperative for a gateway to perform protocol conversion. In another LAN setup, the datagrams could just as well have been transported as IP datagrams. The IP gateway is also capable of fragmenting the IP datagrams if the need arises as a result of a limit being imposed on the MTU size.

The AWANET 100 gateway maintains a call connection table to encapsulate the connection-less datagrams into connection oriented virtual circuits used by the Frame Relay protocol. Frame Relay network distinguishes each virtual circuit by means of it's Data Link Channel Identifier (DLCI). The call connection table maintains a list of DLCI's which are used depending on the destination and the virtual path to be followed across the Frame Relay network. The appropriate DLCI is attached to IP datagrams using the call connection table. The amount of information that needs be stored in the call connection table at the gateway is proportional to the number of networks connected together.

At the remote end the Frame Relay network is connected to the destination LAN by another IP gateway. This gateway receives the frames and strips off the Frame Relay header before passing the IP datagrams down to the FDDI II ring using one of the two backplanes.
The AWANET 100 is an existing LAN. The Frame Relay software and the software drivers for the ATM chips have been developed as part of this masters program. A brief description of the developed Frame Relay software is provided in Appendix B and an overview of the SARA-S and SARA-R drivers for Asynchronous Transfer Mode (ATM) data transmission is given in Appendix C.

1.3 Statement of Problem

A bottle neck exists at the gateway where high speed LANs (such as AWANET 100) join the network. A large number of users trying to access the gateway resources can give rise to highly bursty data transactions. This, combined with the fact that the arrival rate of the packets can differ from the rate at which the gateway is able to access the network, can give rise to congestion at the gateway. Congestion results in a reduction in throughput and therefore the quality of service. Rather then drop an arriving packet simply when the gateway buffer overflows, a more intelligent dropping of packets is required if performance is to be improved.

A number of authors [19 - 24] (described in detail in Chapter 2) have compared different techniques that can be used to drop packets from the gateway buffer once congestion has been detected. A mechanism that monitors the average queue size for each output queue and then randomly discards packets from the queue has also been proposed in literature [19, 20, 23]. It is argued that temporary peaks in traffic will not cause a dropping of packets if good averaging techniques are used. This approach will be discussed in detail in Chapter 4 and it will be shown that while the scheme might work under a few select operating conditions, there is no significant improvement to be obtained from this approach. The deficiencies of papers proposing this scheme will be further discussed in Chapter 2.
A scheme for improving the performance of an IP gateway based on intelligent packet/datagram discard is presented in this thesis. It takes advantage of the fact that IP tends to fragment transport level PDUs into smaller datagrams whose size and number depend on the Maximum Transfer Unit (MTU). Results obtained with statistical testing of an intelligent packet discard gateway will be shown in Chapter 5 indicating that the proposed scheme results in an improvement in throughput and gives better performance than gateways which simply discard packets when their buffer overflows.

1.4 Thesis Outline

This dissertation is divided into six Chapters, with 3 appendices.

Chapter 2 presents an overview of previous work on congestion control. The problem of congestion in gateways is discussed and a number of general congestion detection methodologies are described. The Random Early Discard (RED) gateways proposed in literature [19, 20, 23] to improve IP gateway performance are also described in detail.

Chapter 3 describes the issues involved in the interconnection of LANs by IP gateways. The level of interconnection of LANs at the gateway is discussed in terms of the protocols involved and the advantages/disadvantages of connecting the gateway at each of the levels. In particular, the IP gateway is discussed. Having described the role of a gateway and the level of interconnection, the modelling of LAN network and the gateway is briefly described. This is achieved by translation of a LAN and its gateway into a queuing network model suitable for analysis.

Chapter 4 examines the usefulness of Random Early Discard (RED) of packets in order to avoid congestion at the gateway. A simple simulation model for non-retransmitting sources is described and validated. This is followed by a study of more complex models with
retransmitting sources (using TCP for data transfer). The effect of changing the averaging window size as well as the discard threshold on throughput is described in this Chapter. The issue of traffic synchronisation is also discussed and it is found that while the RED scheme might work under very specific conditions, the algorithm falls apart once exhaustive statistical testing methods are used to measure its performance. Statistical testing, using confidence intervals, analyses the system under a large set of conditions and as a result provide a more accurate picture of the system performance instead of selective specific instances.

Having established that RED gateways proposed in literature do not result in any significant performance improvement when subjected to a large set of test parameters, Chapter 5 proposes a novel intelligent packet discard scheme. This scheme is based in the fact that IP fragments the data passed down from the TCP layer and transmits it over the network. If one of the IP fragments belonging to a packet gets discarded, then the whole packet is retransmitted by the TCP layer which has no means of selectively retransmitting a lost fragment. Chapter 5 presents results which compare the proposed intelligent packet discard scheme with the drop-tail gateways in terms of throughput and percentage of packets discarded. The effect of Protocol Data Unit (PDU) size on the performance of intelligent packet discard gateways is also examined.

Finally, Chapter 6 summarizes the main findings and results of the thesis.

1.5 Contributions

This Section lists the work and contributions contained in this thesis. The Section of the thesis where the work is described is also mentioned.
1. Development of a detailed LAN sources model using TCP and IP for data transmission. The development of this model is of crucial importance if any meaningful and accurate results are to be obtained.

2. Development of a LAN gateway model that implements the drop-tail, RED, and intelligent packet discard algorithms accurately. It uses sliding windows to calculate average packet drop probability. This model is used in Chapter 4.

3. Demonstrating that RED gateways do not give any performance improvement when non-retransmitting sources (Markovian with negative exponential packet generation times) are used. The results are shown in Chapter 4.

4. Demonstrating that RED gateways do not give any noticeable improvement in performance when retransmitting sources are used. Under carefully controlled conditions, RED gateways might give rise to a slight improvement, but in general, results obtained using statistical testing, with confidence intervals, show that the proposed scheme does not result in any improvement in performance. This is shown in Chapter 4.

5. Demonstrating that the performance of a gateway is degraded when IP fragments the data passed down by the TCP layer for transmission. This is discussed in Chapter 5.

6. Development of an intelligent packet discard methodology that does not require explicit signalling to the sources in order to work. This also means that the existing protocols need not be modified to implement intelligent packet discard and hence the proposed improvement can be easily integrated into existing gateways. This scheme is explained in detail in Chapter 5.

7. Demonstrating that the greater the size of the transport layer PDU passed down to the IP layer for transmission, the better the intelligent packet discard scheme proposed in Chapter 5 works.
1.6 Publications


2. Literature Review

2.1 Outline

This Chapter describes the current research trends and discusses different schemes that are commonly used to avoid and control congestion in LAN gateways. The limitations associated with previous research are also discussed in this Chapter. The study of congestion has been split into two main categories


2. Congestion Avoidance and Control.

A brief overview of congestion in LAN gateways is provided in Section 2.2. Congestion detection and notification is discussed in Section 2.3. In Section 2.4, work related to the avoidance and control of congestion is described. A number of schemes which look at modifying the LAN gateways in order to control congestion are discussed in Section 2.4. Finally, Section 2.5 identifies the deficiencies in previous research and outlines the developments to be made in this thesis.
2.2 Congestion in LAN Gateways

The users of LANs as well as the LAN service providers both suffer if congestion occurs in LAN gateways. The users lose in terms of the quality of service being provided to them. This is measured in terms of excessive delays or failure to connect to a remote host etc. Poor quality of service appears to the customer as a network failure and might lead to abandonment of the connection if delays due to a congested LAN gateway are excessively large.

A customer can not expect a good quality of service from the LAN service providers if the loss rates of the packets at the gateway is large. As a result, the LAN service providers strive to avoid and control congestion at the gateways. Apart from the packet loss rate, other crucial parameters used by LAN service providers to measure the performance of the gateway are packet delays and throughput.

A point to note when determining congestion in LANs is to identify the bottlenecks where congestion can occur. The dimensioning of resources, such as buffers, gains significance in this regard. The aim is to effectively utilize the available resources and control congestion.

Given the goals of congestion control mentioned above, an attempt can now be made to improve the performance of LAN interconnection gateways. The next Section describes the methods proposed in literature to detect congestion.

2.3 Congestion Detection and Notification.

Before an attempt can be made to avoid congestion and measures taken to control it, the onset of congestion needs to be detected by the entity responsible for taking congestion control measures. This can either be the sources of traffic themselves or gateways with built-in algorithms to control congestion. Broadly speaking, the detection
of congestion can be done using either explicit or implicit means. Both methods are briefly described below.

2.3.1 **Explicit Detection of Congestion.**

As the gateway is aware of the state of its buffers, it can identify which individual link is overloaded. Therefore, it has the ability to inform the users directly about a congestion condition that might arise. This usually involves either a bit to be reserved for notification purposes in the protocol header or a special congestion notification packet which is sent to the users.

A number of protocols have built in mechanisms for explicitly indicating congestion. The ITU recommendation Q.922 can be studied as an example of the explicit congestion notification scheme. Two bits are reserved by ITU Q.922 recommendation to inform the user of congestion [2]. They are the Forward Explicit Congestion Notification (FECN) and the Backward Explicit Congestion Notification (BECN) bits.

2.3.1.1 **FECN/BECN.**

The FECN bit is set on all frames passing through a node experiencing congestion [6, 8, 9, 10, 11]. This indicator is used by the destination to determine whether the nodes in the direction of flow of data are congested. It then becomes the responsibility of the user to act upon the information provided. Bergmann [2] proposes that the congestion control measures should be taken by averaging the number of FECN frames received over a period of time (usually two round trip delays).
Figure 2.1  FECN/BECN and a congested node

Figure 2.1 shows how FECN and BECN bits can be used by a node to indicate congestion to data sources. The BECN bit is used by source controlled transmitters and requests an immediate reaction from the end user [2]. As a result, the response times associated with BECN are smaller then those associated with FECN. Upon arrival of the first BECN bit, the end user is requested to reduce the transmission rate to the Committed Information Rate (CIR). Bergmann [2] suggests the rate should be further reduced if the number of BECN notifications received exceeds a threshold and normal transmission rate should be restored when a set number of frames are received with the BECN bit cleared.

The FECN/BECN schemes are independent of each other and although the network might support both of them, the end user normally only responds to one of them.

2.3.1.2 Consolidated Link Layer Management (CLLM).

A modification of the BECN scheme is the Consolidated Link Layer Management (CLLM) scheme in which an explicit CLLM frame, rather than a piggy backed notification, is sent to the source to indicate congestion [2]. It can be used when congestion occurs at a network node but no reverse traffic is available to carry the BECN notification. This however, has the adverse effect of further adding to the network load.
While most routers can read the explicit notification bits, they cannot set them [8]. Some router vendors (e.g. Timeplex) recommend that the explicit notification option on their router be switched off during operation as no one else uses them [6].

2.3.1.3 **Explicit Binary Feedback.**

The explicit binary feedback proposed by Goldstien [3] is a congestion management technique that relies on a one-bit indicator in the network/transport layer (different from FECN/BECN). This bit is used by the network nodes to indicate their state of congestion to the user. If any node on the virtual path is congested, the user reduces its output by shrinking the size of its window.

The aim of explicit binary feedback scheme is to avoid the phenomenon of "congestion collapse". Congestion collapse can occur in networks which rely on implicit congestion detection [2]. Retransmissions of lost packets can lead to even more congestion and eventual collapse of the network. One way to avoid congestion collapse is to use dynamic window sizing. In the absence of congestion, the window size gradually increases to its maximum limit. As soon as congestion is observed by packet loss, the window size is shrunk immediately to one. Upon successful retransmission thereafter, the window size increases. This is similar to the slow start mechanism used by TCP for congestion control. Dynamic window sizing alone, however, does not provide a good control mechanism and leads to a saw tooth shaped throughput pattern. Goldstien [3] proposes that in order to avoid achieve better throughput, there is a need for explicit binary feedback scheme.

The imminent onset of congestion is enough for the gateway to set the explicit binary bit of the packet passing through it. If at least half the packets arriving at the user have this bit set, the path can be considered congested. Upon detection of congestion, the user reduces the offered load in a multiplicative manner.
If the gateway is not congested, then the user may increase the load offered in an additive manner. This method, according to Goldstien [3], results in the operation of the network between the "knee" and the "cliff" of the throughput plot shown in Figure 2.2. Upon complete collapse, the implicit control (using timers) is expected to take over.

The gateway determines its congestion state by observing the average size of its buffer over a period of time. Goldstien [3] uses this approach in the network/transport layer. He suggests that the use of BECN instead of binary bit should give similar performance.

The disadvantage of this approach however is that it involves modifying the transport layer to support the explicit binary feedback mechanism. The most commonly used transport layer protocol, TCP, has no explicit congestion notification bit and as such would need to be modified for the explicit binary notification scheme to work.

It can be noticed from the explicit detection and notification schemes described above that the explicit schemes are oriented towards inter-network protocols. A major drawback of explicit notification schemes is that most of the existing transport layer protocols do not
support explicit notification of congestion. As such, a LAN interconnection gateway (using Frame Relay, for example, as the network access protocol) can not inform it's LAN users that a congestion indication has been received from the network. This is especially true if the LAN implements the widely used TCP/IP stack as TCP does not support any explicit means of informing users of congestion.

The implicit congestion detection method is described in the next Section. This method of detection is commonly used by the LAN protocols.

2.3.2 Implicit Detection of Congestion.

As the name implies, this scheme makes use of implicit means (such as packet loss rates, call throughput etc.) to detect congestion. The role of implicit congestion detection becomes evident when interworking of different protocols is taken into consideration. As an example, consider the interworking of IP and Frame Relay. Frame relay serves as an important interface for LAN interconnection [14] and can be used to interconnect TCP/IP LANs using IP gateways. This method of encapsulating IP datagrams into Frame Relay frames is known as “tunnelling” and results in higher speeds then connection-less data networks.

However, the disadvantage of tunnelling is that the gateways can not set the FECN/BECN bits inside an IP packet as no field exists which can accommodate such information and relay it to the traffic sources. This problem is similar to the one encountered when Frame Relay and ATM networks interact [8]. The end result is that while the network might be congested, no explicit congestion notification arrives at the sources. The fact that BECN functionality is missing from the ATM/AAL 5 standard [10] makes explicit congestion mapping even more difficult to achieve.

Apart from the protocol mapping difficulties associated with explicit congestion notification, another reason for using implicit congestion
detection is that most gateways do not support the explicit congestion notification bit and most routers simply discard the FECN/BECN bits in Frame Relay [6]. Thus even if a transport layer can be implemented which has the ability to explicitly inform users of congestion, it would be of little use. As a result, in the absence of any explicit notification, implicit detection emerges as the main way to detect congestion.

A few implicit congestion detection and control schemes are described below. A good overview of implicit techniques is provided by Chen and Rege [1]. The following parameters can be used to detect congestion implicitly.

2.3.2.1 Amount of Discarded Packets.

The number of packets that have been rejected by the gateway provide an indication about the state of congestion [12]. Most data transmission protocols regard a loss of packets in the network as a sign of congestion. TCP initiates it's congestion control mechanism (slow start) as soon as a packet discard is detected by means of a time out [13, 15]. However, not all protocols initiate their congestion control mechanisms as soon as a few packets are discarded by the network. They initialize their congestion control mechanisms only if a preset number of frames get discarded by the network in a given time period.

2.3.2.2 Packet Loss Rate.

The number of unsuccessful packet transmissions provides an indication of the state of the gateway. A large number of retransmissions requests indicates that the gateway might be congested and control procedures might need to be initialized.

In TCP/IP when the retransmission timer expires it's value is recalculated and congestion control procedures invoked if necessary. Nguyen and Schwartz [13] show that increased protocol complexity, and the resulting lower throughput, is mostly due to congestion
avoidance and control procedures in TCP and as such, they should be kept as simple as possible. They further propose that the retransmission timer should be not be recalculated when the link deteriorates. Although this results in a larger retransmission timer value (worst case scenario), a net gain in performance is obtained due to the decreased frequency of timer expiration and the associated overhead [13]. Although Nguyen and Schwartz [13] show that an improvement in the throughput can be obtained using this methodology, the scheme has not been subjected to tests in a real network and nearly all current TCP implementations do recalculate the timers as well as implementing slow-start congestion control mechanism [16, 17].

2.3.2.3 The Call Throughput.

The throughput of a call can also provide insight into the state of congestion at the gateways. A large throughput indicates that the gateway is generally free of congestion and thus the transmission rate could be increased by the sources. Hamid [4] and Waters and Hamid [18] propose the use of a recursive digital filter, known as the Kalman one-step predictor, to detect and avoid congestion using throughput measurements. This scheme will be described in detail in Section 2.4.3.2.

2.4 Congestion Avoidance and Control

Once congestion has been detected using either the explicit or implicit detection procedures described above, it needs to be controlled. The methods found in literature can be divided into the broad categories of bandwidth management, stop duration, window sizing and gateway packet discard. Each of these will be discussed in the following Sections.
2.4.1 Bandwidth Management.

The best solution for congestion control in networks is based on congestion avoidance rather than simple detection and recovery [2, 12]. This has now been stated in some of the ITU recommendations as well. One way to avoid congestion is to implement a rate based scheme. These rely on limiting the amount of traffic passed on to the gateway.

2.4.1.1 Traffic Policing and Shaping.

Traffic "policing" and "shaping" are access rate monitoring schemes by which user data transmission rates are restricted and thus congestion is avoided. Petr and Frost [5] discuss this widely used approach to congestion avoidance in detail. At call setup, the user negotiates a limit on the rate of traffic that it commits itself to transmit at. This limit is necessary to maintain the quality of service for all users and helps in fair resource usage.

A "policer" is a device which resides at the network premises and enforces the negotiated limits on traffic by either penalising, or discarding all together, any traffic that exceeds the negotiated limit. For example, in the case of interworking with Frame Relay, the network can set the Discard Eligibility (DE) bit if a pre-negotiated traffic limit is exceeded by the user [5].

A "shaper" is a device which resides at the user premises (gateway) and ensures that the transmission rate is within the negotiated data rate limits before it is presented to the network policing point. Buffering can be used to accomplish the traffic shaping function [5].

The transmission rate negotiated between the user and the network provider is known as the Committed Information Rate (CIR) and is defined as

\[
CIR = \frac{Bc}{T} \quad \text{(Eqn 2.1)}
\]
where $B_c$ is the maximum number of bits allowed to pass through the gateway in the time interval $T$. The "policer" accepts traffic up to the CIR for transmission by the network. Usually another limit, $B_e$, is also negotiated between the user and the network and is defined as the bandwidth which the network tries to accommodate but is not committed to provide. Any traffic above $B_c$ and up to a maximum of $B_e$ bits becomes eligible for rejection upon the onset of congestion. All traffic above the rate of $B_e$ bits in time interval $T$ is automatically discarded.

Two "policing" options exist and are defined below.

1. Leaky Bucket.

This is a token based scheme. A "bucket" of tokens is used which fills up at a rate of $B_c/T$, where $B_c$ and $T$ are defined above. Each packet allowed onto the network requires a token. Packets are dropped if no tokens are available [4]. Two variations of the leaky bucket are shown in Figures 2.3 and 2.4 below:

![Figure 2.3 Leaky bucket 1](image)

Note that in the case shown in Figure 2.3 above, the scheme is enforced after the packets have been accepted by the gateway buffer [5].
Another variation of the leaky bucket scheme is shown in Figure 2.4 above. In this case, the packets are rejected, if required, before they enter the gateway buffer.

Two leaky buckets can be used in tandem if both Be and Be limits are to be imposed on the traffic. Petr and Frost [5] describe a way to determine how many bits should leave the "shaper" for a proper implementation of the leaky bucket mechanism.

According to Petr and Frost [5] the number of bits, $B_2$, that may be allowed to leave the buffers is given by:

$$B_2 = B_1 + \text{MIN}[X, \left(\frac{F}{R_\alpha}\right) \times CIR]$$

(Eqn 2.2)

where $B_1$ is the bucket fill level upon packet arrival, $F$ the packet length, $R_\alpha$ the link access rate and $B_2$ the "look ahead" buffer fill level.

If $F$ is less than $B_2$, the entire packet may leave immediately but if $F$ is greater than $B_2$, then the packet must be delayed by

$$\text{DELAY} = \frac{B_3}{R_\alpha} + \frac{\langle B_3 - B_2 \rangle}{CIR}$$

(Eqn 2.3)

where $B_3$ is defined as:

$$B_3 = \frac{F}{1 + \frac{\langle CIR \rangle}{R_\alpha + \langle CIR \rangle}}$$

(Eqn 2.4)

According to Petr and Frost [5], the calculated value of delay is the minimum possible delay that must be used to enforce rate control.
2. Quantized Moving Window.

This is a variation of the leaky bucket scheme and is described in detail by Petr and Frost [5]. It is stricter than the leaky bucket in its policing of traffic and is a quantized version of an algorithm that keeps track of the total number of bits accepted during the last T time period.

The "shaper" uses only the leaky bucket algorithm due to its simplicity while the "policer" can use either the leaky bucket or the Quantized Moving Window (QMW) scheme. The leaky bucket can build up credits to cater for bursty traffic whereas the QMW is stricter and does not allow credit build-up. Hence the QMW scheme results in greater packet loss than the leaky bucket.

Having discussed the bandwidth management schemes used for congestion avoidance, the modified gateways will be described in the next Section. The modified gateways refer to schemes which can discard arriving packets even though there might be buffering space available in the gateway. The advantages and disadvantages of different modified gateway schemes are also discussed in the next Section.

2.4.2 Modified Gateways

The congestion avoidance and control schemes described in the previous Section are either protocol specific or somehow rely on the ability of the gateway to inform the LAN users that congestion has occurred in the gateway. They also base their results and congestion control schemes on "drop-tail" gateways. A drop-tail gateway discards packets only when it's buffer overflows. The modified gateways described in this Section use packet discarding strategies other than simple buffer overflow to obtain improvement in performance.
Three different congestion avoidance and control schemes, which propose modifying the FIFO drop-tail gateways, will be examined in this Section.

2.4.2.1 Threshold based congestion control with explicit notification

This is a congestion control scheme based on link buffer occupancy thresholds. The gateway indicates its congested state to the sources by sending explicit messages. Nagarajan [21] proposes the use of three gateway buffer thresholds, all based on buffer occupancy. This scheme is shown in Figure 2.4 below.

![Threshold based congestion control](image)

*Figure 2.5* Threshold based congestion control.

If the buffer size exceeds “discard threshold” D, then packets coming into the buffer are discarded. If the buffer size exceeds the “onset threshold” T, then an explicit message is sent to the sources indicating that they should throttle back the transmission rates. Another explicit message is sent to the sources once the buffer size falls below the “abatement threshold” A. This indicates to the sources that normal transmission rates can now be resumed. The values of the three thresholds, A D and T, are determined by adjusting their values until the desired performance is obtained form the gateway [21].

This scheme conforms to the ITU specification for the SS7 which includes a congestion control protocol based on link buffer occupancy thresholds.
In this scheme, the delay in congestion control messages from the buffer to the sources have an impact on the system performance and throughput. If the thresholds are set at the extreme buffer limits, then there is significant throughput loss. On the other hand, if the threshold are set away from the buffer size limits, there is no significant change in throughput due to delays in control messaging [21]. Also, the control overhead, defined as the number of extra explicit messages that need to be sent indicating the onset or abatement of congestion, is dependent on the threshold positions [21].

The thresholds also have an impact on the average messaging delays. Results presented by Nagarajan [21] indicate that the average message delay is fairly sensitive to the congestion onset threshold, \( T \), and the abatement threshold \( A \). However, the effect of delay due to control messages on average delays is relatively insignificant provided the throttling and abatement delays are comparable.

Though Nagarajan [21] examines the effect that threshold positioning and link delays have on performance of a gateway, the sources used are modelled as Poissonian instead of a more realistic source model which implements transmission protocols (e.g. TCP/IP). Another draw back of the presented results is that the effect of retransmissions, which can have a major impact on performance, is not taken into consideration.

2.4.2.2 Random Discard Gateways

A number of authors [19, 20, 21, 22, 23, 24] have investigated the effect of early packet discard from the gateway buffer to improve performance. In this Section, Floyd and Jacobson's paper [19] will be discussed. They propose that Random Early Discard (RED) gateways be used in conjunction with TCP/IP sources. Their model of the gateway detects the incipient congestion by computing the average queue size. When the average buffer occupancy exceeds a preset threshold, the gateway assumes that congestion is about to occur. The gateway reacts to congestion by either randomly discarding
incoming packets or randomly "marking" them by setting the discard eligibility bit if that option is available.

Certain aspects of the RED gateways have been specifically designed to work in conjunction with TCP sources. An example of this is that a single "marked" or dropped packet should be sufficient to indicate to the transport layer that congestion has occurred. Floyd and Jacobsen [19] only discuss the dropped packet scenario since TCP packets can not be "marked". The TCP sources react to a single dropped packet by initiating the "slow start" congestion control mechanism. Synchronisation of traffic caused by drop-tail gateways, which can cause all TCP sources to shrink the transmit window at the same time, is also addressed by the RED gateways. The use of RED gateways is meant to break the possibility of synchronization resulting in better throughput performance [19].

The algorithm of RED gateways can be written as [19]:

```
Calculate the average buffer size;
If (min_threshold <= avg_buffer size <= max_threshold)
   Calculate drop probability P;
   With probability P, drop the arriving packet;
Else If (max_threshold <= avg_size)
   Drop the arriving Packet;
```

The average buffer size is determined by a simple averaging mechanism. Floyd and Jacobsen [19] did not implement a more accurate averaging mechanism such as a sliding window. Furthermore, a rather simple network was used in their simulation which included only four TCP sources feeding into an IP gateway while some results are obtained with just two sources feeding the gateway. Further investigation of this scheme is carried out in Chapter 4 with a large number of sources.
Floyd and Jacobson [19] propose that the RED scheme keeps the average buffer size low while at the same time allowing occasional bursts of traffic through. While the users could detect congestion by means of end-to-end delay or throughput calculations, it is the gateway alone which has the global view of the interconnection traffic and as such is ideally placed to make any decision regarding congestion.

A major advantage of RED gateways, according to Floyd and Jacobson [19], is that they provide a simple mechanism for avoiding congestion and improve throughput of current TCP networks with no changes required to the transport protocol. Another advantage pointed out by the authors is that it helps break traffic synchronisation in hosts using TCP as the transport protocol.

Floyd and Jacobson [19] do not perform any statistical testing of the presented results. Therefore, while the plots shown indicate an improvement in performance by RED gateways over the drop-tail gateways, nothing can be said for certain as to how the scheme behaves under operating conditions which can vary greatly over time.

2.4.2.3 Dynamics of TCP traffic in early discard ATM gateways

This is an early packet discard scheme proposed by Romanov and Floyd [20] and aims to reduce the congestion in gateways when TCP traffic is carried by Asynchronous Transfer Mode (ATM) cells.

Although the scheme focuses on the dynamics of TCP traffic over ATM, it provides useful insight into gateway behaviour when it is subjected to fragmented TCP traffic. ATM segments TCP data into smaller cells. This is similar to IP's segmentation of TCP data into smaller fragments which is discussed in Chapter 5 and forms the basis of the intelligent packet discard gateways proposed in this thesis.
Romanov and Floyd [20] note that the effective throughput of TCP over ATM can be drastically reduced when the gateway gets congested. The reduction in throughput is due to wasted bandwidth because the congested links continue to transmit cells belonging to "corrupted" packets. Corrupted packets are defined as TCP packets made up of ATM cells that have already been dropped at the gateway. Romanov and Floyd [20] propose that this effect can be corrected and high throughput attained if the gateway monitors the size of the output buffer queue and drops whole TCP packets when the queue reaches a threshold.

Romanov and Floyd [20] also propose early packet discard. This mechanism is similar to the RED gateways. Early packet discard allows the gateway to discard whole packets by looking at the first cell belonging to a packet arriving at the gateway. If the threshold, based on average buffer size, has been exceeded then the gateway discards the first cell and all preceding cells of the packet even though there might be buffer space available in the gateway. The stated goal of early packet discard is to prevent frequent buffer overflow by dropping complete packets before the gateway overflows.

A partial packet discard strategy is also discussed by Romanov and Floyd [20] which is based on dropping all subsequent ATM cells belonging to a packet as soon as one cell of the same packet has been discarded by the gateway. The proposed partial packet discard scheme concentrates on TCP over ATM traffic and does not examine the effect of discarding whole TCP packets from IP gateways. The analysis is based on the performance of an ATM switch. Also, the results presented by Romanov and Floyd [20] have not been subjected to statistical testing. The performance of IP gateways using a partial packet discard scheme is examined in Chapter 5 and a method presented which allows the partial discard scheme to be implemented in IP gateways without the need to modify existing data transmission protocols.
2.4.3 Other Schemes.

A number of other schemes and methods [1, 4, 18, 25] proposed to control congestion are presented in this Section. A separate Section has been devoted to these schemes as they did not fit in the categories described earlier.

2.4.3.1 Sliding Window Techniques.

Use of sliding window technique to facilitate higher speeds is common in data transmission protocols. During no congestion, the throughput increases until the full window size is reached. At the onset of congestion, the window size is reduced which effectively reduces the throughput. As throughput is dependent on window sizing, a number of algorithms exist that determine the transmission window size during 'shut-down' and 'recovery' phase. A few of these are listed below [1].

1. Upon congestion detection, the window size is reduced by one. Recovery is by incrementing the window size by one when N consecutive packets have been successfully delivered (N is any non-zero positive integer value).

2. Upon congestion detection, the window size (W) is reduced to $W_{\text{min}}$. Recovery is similar to 1 above ($W_{\text{min}}$ is any non-zero positive integer value less than W).

3. Upon congestion detection, the window size (W) is reduced to $\alpha W$. Recovery is similar to 1 and 2 above ($\alpha$ is a multiplying factor in the range of $0<\alpha<1$).

It has been shown by Chen and Rege [1] that scheme 3 is the most effective in handling congestion over a wide range of traffic patterns. TCP, however, uses a modified version of scheme 2.

2.4.3.2 Kalman Prediction.

The Kalman congestion prediction method [4, 18] is a scheme proposed for gateways that don’t support explicit congestion notifica-
tion. It is a congestion avoidance scheme that is based on a recursive digital filter known as the Kalman one-step predictor. The Kalman predictor is used to detect the onset of congestion [4, 18] thus allowing maximum utilization of the gateway resources. The implementation of the scheme is dependent on throughput measurements of the sources.

Congestion avoidance is a preventive mechanism whereas congestion control is a recovery mechanism. The Kalman congestion avoidance technique revolves around congestion avoidance rather than control. However, congestion control is still needed to protect the network in case congestion does take effect. Hamid [4] and Waters and Hamid [18] claim that the Kalman technique allows the gateway to operate in regions of high throughput.

The Kalman technique follows four steps [18]:

1. Observation

Throughput is used as the observation parameter for congestion. The aim of the Kalman technique is to maintain the throughput between the “knee” and the “cliff” regions shown in the throughput versus load graph presented in Figure 2.2.

2. Prediction.

The Kalman equations used to predict the onset of congestion have been derived by Hamid [4]. A scalar approach to the Kalman filter is applied. Interested readers may refer to [4] for additional details.

3. Decision making process.

Once prediction about the state of congestion has been made the decision making process decides whether to increase or decrease the load based upon the region of operation and the number of successful transmissions.

4. Window sizing.
The Kalman scheme assumes that end users implement a window based transmission strategy. The window size is increased in an additive manner but decreased in a multiplicative manner. This window sizing methodology ensures a rapid response at the onset of congestion but results in a slow recovery. The difference between this approach and that used by TCP is that TCP shrinks its window size to unity instead of a multiplicative reduction in window size.

Hamid and Waters [18] propose that improved performance can be achieved if a different window level is used for each region of operation. The results presented [18] indicate that this scheme provides even better performance than explicit control schemes such as FECN/BECN notifications [18]. The drawback of this scheme is that it is based on prediction of traffic which is not always successful.

2.4.3.3 MTU Probing.

IP segments TCP packets into smaller fragments whose size depend on the MTU of the physical connection media. The aim of MTU probing scheme is to determine the smallest MTU along a data path and thus avoid repeated fragmentation by IP gateways during data transmission.

Tsudik [25] proposes a method which can be used by the TCP/IP sources to determine the smallest value of the MTU in the path. The author suggests that special “probe” packets be sent which determine the MTU of the path before any data transfer takes place. The disadvantage of this scheme is that while it guarantees fragmentation avoidance, it requires changes in all gateways to support probing. Another major disadvantage of this method is that all packets are limited by the smallest MTU along the path. This means that the small sized packets would unnecessarily congest the network and not make use of the benefits associated with large packet transfer such as low overhead.
The deficiencies of papers discussed in this Chapter will be summarised in the next Section. The developments made in this thesis are also outlined.

2.5 Deficiencies and Developments.

2.5.1 Deficiencies of previous work

An overview of congestion in LAN gateways was provided in Section 2.2. In Section 2.3, the explicit and implicit means of detecting congestion were defined. Section 2.4 discussed various schemes of congestion avoidance and control under the headings of bandwidth management, modified gateways and other schemes. A few of the deficiencies that came to light as a result of the literature review are listed below.

1. In the case of interconnected LANs, there is no means for the gateway to explicitly notify the users that congestion has occurred. An explicit notification scheme would require modifying the currently existing protocols. This would effectively eliminate any chances of the gateway being able to work in current environments.

2. A vast majority of the papers reviewed either concentrate on modelling either the sources or the gateway. Simplified sources, such as Poissonian, do not fully model a TCP/IP LAN environment. Very few papers, such as [19, 20, 21] implement the full protocol stack at both the source and the gateway.

3. As a result of 2 above, most papers fail to take into consideration the effects of retransmission when discussing LAN gateways. Only a few of the described papers [19, 20, 21] implement the full TCP stack and thus take the effect of retransmissions into consideration.
4. While a lot of work has been done in studying the performance of TCP/IP, none of the authors [13, 15, 16, 17, 19, 20] have considered the effect of IP fragmentation on modified gateways.

5. Amongst the papers that do implement the full TCP/IP protocol stack at the source [19, 20, 21], very few consider the impact of a large number of sources on the gateway. Floyd and Jacobson [19] only consider four sources while Romanov [21] does a bit better by taking into account 11 sources. Modelling a small number of sources can lead to erroneous results.

6. The results presented by some authors [19, 20] need to be statistically tested. This is of critical importance amongst the modified gateway papers because instantaneous results, without confidence percentiles, can be misleading.

2.5.2 Developments in this Thesis

Each of the six deficiencies noted above is addressed in this thesis:

1. Chapter 5 describes an intelligent packet discard scheme being proposed in this thesis to improve the gateway performance. This scheme does not require explicit signals to be sent to the users, hence removing the need to develop customised protocols. As a result the proposed scheme can be easily be integrated into existing gateways.

2. The full TCP/IP stack right down to the physical layer has been modelled at the sources whereas the protocol stack up to the IP layer has been implemented at the IP gateway. TCP has been left out of the gateway's simulation model as it is not required by IP gateways.

3. Since the full TCP/IP protocol stack has been modelled, the retransmission have been taken into consideration. Also, as a result of the detailed implementation of the model, the TCP timers
and the congestion control mechanism of the transport layer are fully taken into account.

4. The effect on interconnecting LANs using TCP/IP is fully examined in Chapters 4 and 5. IP fragmentation is further discussed in Chapter 5.

5. A large number of sources, up to 24, have been implemented to provide a realistic picture of the TCP/IP gateway behaviour. As a result of the large number of sources, any effect that traffic synchronization might or might not have on performance is automatically taken into consideration.

6. All results presented have been subjected to statistical testing with confidence intervals of 95% or greater.

The major part of this thesis is concerned with achieving the best possible performance out of an IP gateway used to interconnect LANs with sources using TCP/IP for data transfer. The following Chapter describes the interconnection of LANs using IP gateways and presents a model that can be used to study gateway performance.
3. Analysis of LAN Gateways

3.1 Introduction

The intent of this Chapter is to describe a typical LAN interconnection environment. The flow of traffic in a LAN and the use of queuing models to analyse the performance of an IP gateway is also described. This Chapter deals with the problem of how best to analyse the performance of a LAN IP gateway and describes the protocols used in a typical LAN/WAN environment for data transmission.

As described in Chapter 2, congestion control in LANs is a vital issue and is one of the most challenging aspects of data transmission. Most existing LANs use the TCP/IP protocol stack for transmission of data. The vast majority of existing LANs are connected to other LANs giving rise to a large number of Wide Area Networks (WANs). The point at which the LAN accesses the network is called the LAN gateway. In order to improve the performance of a LAN gateway, it is imperative that some knowledge of LAN protocols be provided.

The layout of the Chapter is as follows. Section 3.2 describes the protocol stacks commonly used to interconnect LANs. The interaction of different data transmission layers and the level of interconnection provided by the gateway is also described in this Section. Section 3.3 briefly discusses the associated congestion control
issues at the IP gateway of a LAN. Section 3.4 describes the development of a simulation model, using queuing theory, which can be used to study congestion in the LAN gateway. Finally, a conclusion of the Chapter is given in Section 3.5.

3.2 Interconnection of LANs using gateways

The explosion in the number of PCs and the proliferation of the LANs has given rise to large scale interconnectivity. Interconnecting these diverse LANs using different data transmission protocols, which also operate at different speeds, has given rise to a number of issues and problems. The access point of the LANs to the network is called the gateway. Interconnectivity provides users the ability to communicate with remote networks and make use of their services. When the gateway is used to connect networks running at different speeds, issues such as addressing, routing, congestion control and fragmentation need to be addressed.

LANs not only differ in geographic scope but also in the types of services they provide and the transmission protocols they use. Thus for a LAN interconnection strategy to succeed, these differences in individual LANs need to be accommodated and care taken in the design of the LAN gateways.

3.2.1 Level of Interconnection

Before the performance of a LAN gateway can be analysed and an attempt made to improve it’s performance, the level in the protocol
hierarchy at which the LANs are interconnected should first be decided. Consider the protocol stack options shown in Figure 3.1:

<table>
<thead>
<tr>
<th>Network Protocols</th>
<th>IP</th>
<th>DECNET</th>
<th>OSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Protocols</td>
<td>FDDI</td>
<td>Ethernet</td>
<td>Token-Ring</td>
</tr>
<tr>
<td>Physical Transmission</td>
<td>Fibre</td>
<td>Copper</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.1** The protocol stacks in a LAN

The gateway can be connected at any of the three protocol levels shown in Figure 3.1 above and the performance of the gateway is dependent upon the level of interconnection. The complexity involved in conversion and connecting together protocols at the gateway is also determined by the level of interconnection chosen. The basic services provided by each level of the protocols are given below.

**Table 3.1 The bottom three layers of the LAN protocol stack**

<table>
<thead>
<tr>
<th>LAYER</th>
<th>FUNCTIONALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Provides transparent routing of messages between two transport entities. Does not ensure reliable transmission of data</td>
</tr>
<tr>
<td>Link</td>
<td>Provides rules for transmission onto the physical layer such as packet formats, error detection etc.</td>
</tr>
<tr>
<td>Physical</td>
<td>Provides mechanical and electrical level interconnection</td>
</tr>
</tbody>
</table>

In preparing an outgoing packet for transmission, each layer appends its own header to the data passed down by the upper layers. The issues associated with interconnecting LANs at each of the protocol levels are briefly described below.
**Physical Level:** Interconnection of LANs at the physical level is done by devices called repeaters [31]. They forward individual bits of packets as they arrive, translating from co-axial to fibre, for example, if so required. They are generally used to interconnect several physically separate segments of a LAN, perhaps separated by a point-to-point link. The resulting system functions essentially as a single system and does not fulfil the requirements of a WAN.

**Link Level:** The major use of interconnection at the link level, is to interconnect LAN segments with different Medium Access Control (MAC) protocols and to increase the capacity of the network by filtering out the incoming packets. The devices that perform this function are known as bridges.

**Network Level:** The devices that provide interconnection at the network level are called gateways. Their main use is to interconnect LANs which are running different protocols for a variety of functions such as routing, congestion control, error handling and segmentation. Of particular interest in this regard are the commonly used IP gateways. These can be used for transmitting IP packets by other transmission protocols such as Frame Relay. They work by placing an IP layer on top of other protocols.

It can be seen from the interconnection level description above that network level interconnectivity is most appropriate for the purposes of LAN interconnection. Choosing this level makes available the services of the network level protocols to solve gateway problems (e.g. routing, segmentation etc.) as well as allowing the implementor to take advantage of the protocol standards. This level on interconnection does, however, give rise to questions about the ability of gateways to implement congestion control functions.
3.3 Congestion in the IP Gateway

From the brief description provided in Chapter 1 of a typical LAN (AWANET 100) it can be noticed that a large number of sources feed into a gateway. This can give rise to highly bursty data transactions which pass through the gateway. Bursty data, and the differences between the arrival rate and service rate of the gateway can lead to congestion. Differences in transmission speeds and arrival rates become all the more important when high speed LANs, such as those using FDDI II, are connected together using slower speed WAN services, such as Frame Relay, running on narrow band ISDN. A congested LAN gateway results in a bottleneck in the transmission of data and can lead to a degradation of the quality of service.

A number of authors [15, 16, 17, 19, 20, 25] have looked at improving the performance of the IP gateway. Since the packets in a connection-less system do not necessarily follow a given data path, questions have been raised about the ability of these systems to providing effective congestion control. This issue is of particular significance when connection-less "datagrams" are used to support connection oriented higher level services. Early discard of IP packets has been proposed in literature [19, 20, 25] to achieve better performance from the IP gateway. Random discarding of IP packets, once an average packet discard threshold is exceeded is examined in greater detail in Chapter 4.

When networks with differing maximum packet sizes are connected together, the need to fragment large packets for transmission through the network must be taken into consideration. This issue is of critical importance in IP gateways because IP fragments packets into datagrams whose size depends upon the MTU of the physical media. The reassembly of the fragments, however, is facilitated in the case of Frame Relay transportation of datagrams across the network as it's connection-oriented nature ensures that the all fragments belonging to a packet arrive in sequence. A intelligent packet
discard scheme, which makes use of IP data fragmentation to improve the performance of IP gateway will be proposed in Chapter 5.

3.4 Modelling the LAN

The interconnection of LANs using gateways has been discussed in the previous Sections of this Chapter. It was seen that the choice of gateways operating at the network level is fairly common and in particular, the IP gateway was briefly described. Correct modelling of the LAN is essential if accurate results are to be obtained. In this Section, the translation of a realistic physical LAN, with TCP/IP stacks running at the sources, into a network model that can be analysed will be described.

Queuing models are extensively used in network research to model traffic in a system. The translation of a LAN into a model that can be studied using queuing models involves substituting each of the network elements with a queue and is described below.

Consider a general representation of a LAN inter-connected by IP gateways and shown in Figure 3.2. Each of the sources uses TCP/IP for transmission of data.

![Figure 3.2](image)
This LAN now needs to be translated into a queuing model which can be simulated and studied. A realistic queuing model of the above network is shown in Figure 3.3. As the scope of this thesis is limited to the performance of the gateway, only the gateway, sources and the links between the gateway and the sources have been modelled. All data transmission links have been modelled as full duplex. However, for reasons of clarity, Figure 3.3 only shows the transmission links for the sources on the right and the receiver links for the sources on the left. The OPNET simulation package is used to implement the developed model.

![Figure 3.3 Queuing Model of interconnected LANs](image)

**3.4.1 Using the queues**

Once the network has been modelled in terms of queues as shown in Figure 3.3, the queuing parameters are then calculated. This requires defining the traffic characteristics.

The queuing parameters required to model a network vary depending on the network being simulated. For example, if an M/D/1 system is to be analysed, the following parameters are required for the model:
• Mean arrival rate.

• The service time of each packet.

• The buffer capacity of the queue.

These parameters are derived from the traffic modelling of each queue. For the gateway, the arrival rate is the sum of the arrival rates from each source to the gateway. Similarly, the service time is the processing time requirement for each packet buffered up at the gateway.

3.4.2 Gateway Parameters

The allowable speed at which a gateway can access a network is important in performance studies. This is represented by the packet service rate of the gateway model. The storage capacity of the gateway is modelled as a simple FIFO queue for a drop-tail gateway but more complex algorithms for handling the incoming traffic have been implemented for the RED and intelligent packet discard gateways.

Each of the links from a traffic source of the gateway is modelled as a buffer and incorporated in the physical layer model shown in Figure 3.3. Each link has an adjustable delay parameter. In order to avoid dropping packets at the physical layer, the size of the link buffer can be modelled to be infinite. The total delay from source to destination is given by adding together source-to-gateway delay, delay in the gateway buffer, packet service time and the gateway-to-destination delay.
The following parameters have been used to model the gateway:

**Table 3.2 The Gateway model parameters.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Rate</td>
<td>Adjustable</td>
<td>Determines the speed at which packets are serviced by the gateway</td>
</tr>
<tr>
<td>Route File</td>
<td>ip_rte_gwy</td>
<td>The routing file used by the IP module in the gateway to determine the link to send the serviced packet on</td>
</tr>
<tr>
<td>Network Number</td>
<td>0</td>
<td>Identifies the network number for the IP packets routing</td>
</tr>
<tr>
<td>Node Number</td>
<td>0</td>
<td>Identifies the node number for IP packet routing</td>
</tr>
<tr>
<td>Buffer Capacity</td>
<td>Adjustable</td>
<td>Definable in terms of either bit size or number of packets</td>
</tr>
</tbody>
</table>

As an IP gateway is being modelled, the gateway incorporates within it an IP module which determines the routing path of the serviced packets. Note that the modelled gateway only implements a single class of traffic. Priority packets are not allowed and unless rejected, the packets get served in a FIFO manner.

### 3.4.1 Source modelling

Apart from the gateway, the other network entity that requires careful modelling are the sources. Sources play an important role in determining the performance of the LAN gateway. Due to its widespread use, the TCP/IP stack has been modelled for the sources.

The protocol stack at the sources has been modelled in a modular fashion, with the protocol stacks represented by their respective buffers. A brief description of how the translation from a real source into a queuing model is performed is given below for each layer.

**Application Layer Model:**

The application layer model generates data packets which are transmitted to the remote LANs using the TCP/IP protocol stack. The inter-arrival times between the generated traffic has been modelled to be negative exponentially distributed. The application
layer model also specifies the remote host and port number for connection. This allows multiple TCP applications to run on a single host model. For the rest of this thesis, unless otherwise specified, two applications have been modelled per node.

**Transport Layer Model:**

This protocol layer model is based on TCP and provides the following functionality:

- End to end reliability based on acknowledgments and retransmissions. The retransmissions are based on timers. Note that this has been modelled, as shown in Figure 3.3, as a queue due to the need to keep a copy of the transmitted packet while waiting for an acknowledgment from the remote host.

- Flow control based on the availability of the remote buffering resources (TCP's receive window).

- Reordering of out of sequence data.

- Connection establishment and termination.

- Slow start congestion avoidance mechanism.

The transport layer model receives data from the application model and passes it down to the network layer for transportation.

**Network Layer Model:**

The network layer model is based on IP. IP has been chosen for modelling purposes due to it's wide usage and ease with which it can interact with TCP. As shown in Figure 3.3, this has also been modelled as a queue and it stores the packets received from the TCP layer, fragments them if required, and then transmits them at a user selectable transmission rate. All received datagrams are also stored and reassembled before they are passed on to the transport layer model. This module expects the network and node numbers to be defined for routing purposes.
Physical Layer Model:

The physical layer has been modelled as two separate queues. This can be seen from Figure 3.3 which shows both a transmitter and a receiver at the physical level. This layer represents the transmission medium and associated drivers. The use of a queue to represent this layer becomes clear if the example of ethernet is considered. Ethernet has its own separate buffer where the packets received from higher layer for transmission, and packets received from remote locations, are kept until needed by the appropriate layer. For the sake of simplicity, no contention for the physical medium has been modelled as each link feeds directly into the gateway. This simplicity in modelling is valid as the bottleneck caused by the gateway is of interest in this thesis rather than the contention issues associated with a shared link.

The simulation model developed using the guidelines described above is used in Section 4.5. and throughout Chapter 5 to obtain results. This comprehensive model reflects a real TCP/IP interconnected LAN environment and therefore the results obtained can be expected to accurately depict a real system.

3.5 Conclusion

In this Chapter, the interconnection of LANs by IP gateways was discussed in detail. The level of interconnection to be used by the gateways was studied and the IP gateway selected for further study. It was noticed that congestion can occur in the gateway when the arrival rate at the gateway and the connection rate to the wide area network differ in speed. A typical interconnected LAN environment was also described and a step-wise progression made towards a modelled LAN. This was achieved by translating the physical entities into queues. In this regard the modelling of the gateway and the sources was discussed and a comprehensive simulation model intro-
duced which will be used in Chapters 4 and 5 to study the performance of LAN gateways.

4.1 Introduction

It was mentioned in Chapter 2 that random dropping of packets from the gateway buffer has been proposed in literature [19, 20, 23] in order to avoid traffic synchronisation. The authors stated that traffic synchronisation can result in non-efficient utilisation of the gateway buffer resources. In the RED scheme, the gateway randomly discards the newly arrived packets once the average queue packet occupancy has exceeded a set threshold. As a result of the literature survey, the following issues came to light which need further investigation:

- The issue of a large number of sources with differing link delays was not considered in previous research and needs to be investigated.

- The results of random packet discard gateways need to be subjected to statistical testing. Instantaneous results can be misleading when it comes to performance measurements.
The effect of variations in the packet drop threshold and size of the averaging window on gateway performance needs to be investigated.

This Chapter addresses the issues raised above. The main points which will be investigated in the following Sections are:

1. Examine the usefulness of gateway congestion control based on random discarding of packets.

2. Investigate the random packet discard scheme for both non-retransmitting and retransmitting sources.

3. Describe a simulation setup and use it to compare the throughput performance of the random discard gateway with a drop-tail gateway.

4. Investigate the effect of changing the averaging window size, as well as changing the discard threshold, on random packet discard gateways.

It will be shown in this Chapter that the method of random packet discarding, as proposed in the literature, to avoid gateway congestion results in no significant performance improvement for gateways used to Interconnect LANs. An investigation of this scheme will be followed in the next Chapter by a proposal for a better congestion control algorithm.

The simulation model described briefly in Chapter 3 has been used to simulate systems such as retransmitting and non-retransmitting sources, drop-tail gateways, random early packet discard gateways, and intelligent packet discard gateways. The simulation model will be validated by comparing the simulation results with analytic results for a system such as M/D/1. The M/D/1 system has been chosen for validation due to the availability in literature of its analytic results.
The layout of the Chapter is as follows. The Random Early Discard (RED) gateways are described in Section 4.2. Simulation results for non-retransmitting sources with different blocking probability threshold values are presented in Sections 4.3 and 4.4. The results obtained for RED gateways are compared with those for a gateway not implementing the random discard algorithm. A simple Markovian source is used to represent a non-retransmitting source. TCP/IP sources are used in Section 4.5 to study the case where source reacts to a dropped packet by re-transmitting it after a timeout. TCP has been selected to model the traffic source due to its widespread use in real networks. It has a retransmission strategy capable of dealing not only with the lost packets but gateway congestion as well. The results obtained will be analysed in Section 4.6.

4.2 The RED Gateway

The random early discard gateway drops packets arriving at its node randomly with a set dropping probability once the buffer size exceeds a maximum threshold. Hashem [23] has investigated the early packet discard gateways and points out that the drop-tail method, most commonly used in the leaky bucket scheme, can result in synchronisation of the transmitting sources. This can result in a degradation of network performance.

A number of authors [19, 20, 26, 27, 28] have stated that traffic synchronisation causes oscillations in traffic patterns. Lawrence et al. [29] found that the synchronisation effect is maximum when the feedback delay is constant. The synchronisation of traffic occurs due to the fact that as the packets are dropped from the tail of a full buffer, all the effected sources reduce the size of their flow control windows simultaneously. This leads to a less efficient use of resources and results in reduced throughput.
Jacobson and Floyd [19] propose a modification of the early detection method proposed by Hashem [23]. Their method, known as the Random Early Discard (RED) gateways is also based on random dropping of packets when the buffer size exceeds a set threshold. It differs from the method proposed by Hashem [23] in that the probability of random packet discard increases as the buffer size increases. The analysis presented by Jacobson and Floyd [19] indicates that their method breaks the traffic synchronisation and results in an improved throughput when compared with the simple drop-tail gateways.

In order to investigate the RED gateways, two types of discard methodologies were studied. The first methodology based the random discard on the "buffer size" (as described in literature). The second methodology was a slight modification of the first and based random packet discard on the "packet discard probability".

It was observed that similar results were obtained for both schemes. The reason for focusing on the "discard based on loss probability" is that it provides a much more direct means of controlling the packet discard probability, which is the parameter that needs to be minimized.

The result presented in this chapter evaluate the RED scheme as proposed in literature [19] where appropriate reference is also made of the modified discard scheme which bases discard on packet discard probability.

![Figure 4.1 The Drop Tail Gateway](image)
Figure 4.1 shows a normal “drop-tail” gateway with no random packet discard. The packets arrive at the buffer with an arrival rate $\lambda$ packets/second and are served from the buffer at the rate of $\mu$ packets/second. When a packet arrives at the gateway and finds other packets waiting to be served, it is stored in the gateway buffer. However, if the buffer is already full then the drop-tail gateway discards the newly arrived packet ($B_d$). In the case of retransmitting sources the higher layer protocols detect that the packet has been rejected and as a result the blocked packet is retransmitted.

The disadvantage of the drop-tail scheme is that once the buffer is full, all new packets arriving at the gateway are rejected. Under certain conditions, this can give rise to traffic synchronisation whereby all traffic sources back off and then resume normal transmission at about the same time. This synchronisation effect can lead to an inefficient use of buffer resources.

In the RED scheme, two thresholds based on packet loss probabilities are used to determine whether a newly arrived packet should be accepted into the gateway buffer or rejected. These are known as the lower ($T_L$) and upper thresholds ($T_H$). The RED gateway is shown in Figure 4.2 below.

![Figure 4.2 The Random Early Discard Gateway](image)

The RED gateway behaves in a similar manner to the drop-tail gateway described earlier as long as the average packet dropping probability is less than a preset lower threshold. Once the blocking
probability exceeds the lower threshold, the newly arrived packets are dropped from the buffer with a probability \( P_R \). Note that a packet can be discarded under the RED scheme even if there is buffer space available in the gateway. The packet dropping probability increases linearly as the average packet blocking probability approaches the upper threshold. If the blocking probability continues to rise beyond the upper threshold, then all subsequent arriving packets are discarded. The next Section investigates RED gateway performance for non-retransmitting sources.

4.3 Non-Retransmitting Sources and RED Gateways

The simulation models developed to study the performance of gateways have been kept as simple as possible while ensuring that the model provides realistic results. The models used for the non-retransmitting sources will be briefly described in this Section. In order to ensure that results being obtained are correct, the simulated system needs to be validated. Validation has been achieved by converting the model into an M/D/1 system and comparing the analytic results obtained from literature [30] with those obtained using the simulation model. Note that only the input conditions need be changed in order to convert the model for non-retransmitting sources into a M/D/1 system.

4.3.1 Non-Retransmitting Model

The model used consists of a simple source of packets transmitting into a gateway. The gateway serves the packets in a First In First Out (FIFO) manner, forwarding them on to the packet sink after service as shown in Figure 4.3.
The reason for selecting a simple source model is that the implementation is straightforward. The source generates packets at inter-arrival times (specified at run-time) and has no means of detecting packet loss. The gateway implements a FIFO queuing of packets with the ability to implement both the drop-tail as well as the RED gateway models.

4.3.2 Validation

Validation has been achieved by comparing the model shown in Figure 4.3 to a M/D/1 system. In order to generate the Markovian arrival process required for M/D/1 queues, the inter-arrival times of the packets generated by the source model are negative exponentially distributed. All packets are of the same size resulting in a deterministic service process. As only one server has been modelled at the gateway, the requirements of an M/D/1 system are satisfied. The packets arrive at the gateway (modelled by a FIFO buffer) and are accepted if the buffer has enough storage capacity available. If the packet cannot be stored, it is discarded. No random discarding of packets takes place. The sink serves as a packet discard process destroying the packets once they have been served by the gateway and freeing up the system memory.

The parameters required by the non-retransmitting model for operation are shown in Table 4.1. If a drop-tail gateway is to be modelled then the minimum and maximum threshold values are set to infinity which has the effect of converting a random discard gateway into a
drop-tail gateway. The window size parameter is used by the RED gateways to decide the width of the averaging window.

The variable parameters in Table 4.1 below can be changed at runtime to investigate the effect on gateways performance of changing one parameter while keeping others constant. The gateway’s service rate has been kept constant and different load conditions can be obtained by varying the inter-arrival time of the packets being generated by the source.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>Variable</td>
<td>Used to change utilisation</td>
</tr>
<tr>
<td>Gateway’s Service Rate</td>
<td>16000 bps</td>
<td></td>
</tr>
<tr>
<td>Minimum Threshold</td>
<td>500</td>
<td>Large value so no random drops</td>
</tr>
<tr>
<td>Maximum Threshold</td>
<td>600</td>
<td>As above</td>
</tr>
<tr>
<td>Window Size</td>
<td>N.A.</td>
<td>Not used during validation</td>
</tr>
<tr>
<td>Buffer size</td>
<td>infinite</td>
<td>For comparison to M/D/1</td>
</tr>
</tbody>
</table>

During simulation, the mean delay encountered by packets as they pass through the gateway is measured. This includes the time spent in the buffer waiting to be served as well as the packet service time. The delay has been plotted versus queue utilisation where utilisation (\(\rho\)) is defined as the arrival rate of the packets at the gateway divided by its service rate. The results are shown in Figure 4.4.
Figure 4.4 Validation: Simulated and analytical results

The plot in Figure 4.4 has been obtained for a gateway with infinite buffer capacity. It has been generated with 95% confidence intervals (shown on the plot). According to Klienrock [30], the mean delay encountered by packets in the buffer for the M/D/1 system is given by:

$$E(T) = \frac{1}{\mu(1-\rho)} \cdot \left(1 - \frac{\rho}{2}\right)$$

(Eqn 4.1)

The analytic results given by equation 4.1 above have been plotted alongside the simulation results in Figure 4.4. Notice that the analytical results agree closely with the results obtained from the simulation model thus proving the validity of the model.

4.4 Analysis of Non-Retransmitting Sources

This Section presents the results of simulation experiments undertaken to analyse the performance of RED gateway for the non-retransmitting sources. As the name suggests, a non-retransmitting source has no means of detecting if a transmitted packet has been
lost during transmission. Consequently, there are no retransmissions of lost packets.

In the following set of simulation runs, the drop-tail gateway has been replaced by the modified RED gateway. The RED gateway behaves like the drop-tail gateway until the average packet blocking probability exceeds the lower threshold. Once this threshold is exceeded, the gateway starts to randomly discard the arriving packets. The average packet blocking probability is calculated using a moving window of ten samples. The moving window acts like a circular buffer and discards the oldest measured sample replacing it with the new values. The counter in each of the moving windows is incremented by one each time a packet is dropped. The total packet arrivals during each sample period is also recorded.

![The sliding window](image)

**Figure 4.5** The sliding window

It is clear from the above description that a small window size means a quick reaction time and a larger window size means a slower response to any changes in traffic conditions. This property can be utilised to determine what burst size of traffic the gateway should be allowed to accept. Also, the window size can be varied in order to investigate the reaction time of the gateway to changes in traffic conditions

### 4.4.1 Effect of Window Sizing

In order to study the effect of window sizing on RED gateway performance, packet delay through the gateway is measured for different window sizes. The mean delay encountered by the packets in
RED gateways is compared with the packet delay for a drop-tail gateway. Two different window sizes were chosen in order to test the RED gateway in various set up configurations. Enough simulation runs were made in order to obtain results with 95 percent confidence intervals.

![Mean Delay vs Utilisation](image)

**Figure 4.6 Effect of Window Sizing**

Figure 4.6 above compares the mean delay values of the drop-tail gateway with two RED gateways. One RED gateway has a sliding window time period of half a second while the other a period of five seconds. The average packet blocking probability is calculated by taking an average of the number of packets blocked from entering the buffer and dividing by the total number of packet arrivals during each sliding window time period.

It can be noticed from Figure 4.6 that there is little difference in performance between the RED gateway and the drop-tail gateway. This is due to the negative exponential nature of the incoming traffic and the fact that once the packets are dropped from the gateway, there is no mechanism for re-transmission. The source has no way of know-
ing whether the transmitted packet has reached the destination or has been lost in the system. Another explanation of the results shown in Figure 4.4 is that no traffic synchronisation can occur in non-retransmitting sources as they do not use transmission windows. In the absence of traffic synchronisation, RED gateways do not offer any advantage over drop-tail gateways.

4.4.2 Effect of Threshold Setting

To investigate the effect of the packet blocking probability thresholds on RED gateways, a set of simulations are performed for three different threshold settings. Figure 4.7 below plots the mean delay versus the gateway utilisation.

![Mean Delay vs Utilisation](image)

Figure 4.7 shows results for three gateways having lower thresholds of 0.005, 0.4 and 0.8 and upper thresholds of 0.006, 0.5 and 0.9 respectively. It is apparent from the results that changing gateway threshold values has little effect on the gateway performance in the case of non-retransmitting sources.
Based on the results obtained by changing window size and those obtained by varying the threshold settings, it can be concluded that the random discarding of packets does not improve the performance of gateways if there are no retransmissions of the dropped packets. This can be attributed to the fact that in the absence of retransmissions, sources can not develop synchronisation in traffic. The simulation scenario described in the next Section takes retransmissions into account.

4.5 Retransmitting Sources and RED Gateways

In order to study the performance of RED gateways when subjected to re-transmitting sources, a full implementation of the TCP/IP simulation model is used in this Section. Use of this model also allows accurate comparisons to be made with various schemes proposed in literature [19, 20, 23] which have also used TCP/IP to study random packet discard.

All results presented for the retransmitting model in this Chapter have been subjected to extensive statistical testing. TCP/IP, being a transport layer protocol, implements retransmissions by using timeout to detect packet loss (Chapter 5 provides more detail on the workings of TCP/IP). This feature of the protocol is of interest and will be examined in the following Sections. A diagram of the model being used is shown in Figure 4.8.
Notice from Figure 4.8 that a large number of sources have been implemented in order to give a realistic view of a network. Floyd and Jacobson [19] have stated that a large number of sources, when feeding into a gateway can cause traffic synchronisation. This effect will be examined in detail when the results are analysed in the Section 4.6.

The parameters required by the gateway model to function properly are listed in Table 4.2. Note that Table 4.2 lists the parameters used and shows if the value can be changed for different simulation runs.
The exact value used for each parameter in a simulation run will be provided when results are presented.

**Table 4.2** The Parameters used by the re-transmitting TCP model

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>Variable</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>Fixed</td>
</tr>
<tr>
<td>Minimum Threshold</td>
<td>Variable</td>
</tr>
<tr>
<td>Maximum Threshold</td>
<td>Variable</td>
</tr>
<tr>
<td>Window Size</td>
<td>Variable</td>
</tr>
<tr>
<td>Weightage Value</td>
<td>Variable</td>
</tr>
<tr>
<td>Gateway Buffer size</td>
<td>Variable</td>
</tr>
</tbody>
</table>

The load on the gateway is changed by keeping the service rate constant while changing the packet generation time of the sources. The two threshold values (minimum and maximum) shown in Table 4.2 are used by the gateway for implementation of the RED algorithm. The random packet dropping probability of packets arriving at the gateway is given by the equation [19]:

\[
\text{DropProbability}(P) = RDW \times \frac{\text{WindowAverage} - \text{MinThreshold}}{\text{MaxThreshold} - \text{MinThreshold}}
\]  

(Eqn 4.2)

Where RDW is the “Random Drop Weightage” and is used in calculating a weighted moving average. In RED gateways, the degree of burstiness that the gateway is able to accommodate is incorporated into the calculation of the drop probability [19] and equation 4.2 uses the RDW to incorporate the burstiness of arrivals [19]. The RDW value can have a minimum value of zero, in which case no packets are randomly discarded, and a maximum value of 1, in which case all packets over the minimum threshold are discarded.

The “window size” parameter in Table 4.2 determines the size of the sliding window which maintains the dynamic running value of the ratio of packets being dropped from the tail to the total number of packets arriving at the gateway.
The average packet blocking probability is compared to two thresholds; the minimum and the maximum. When the average drop probability is less than the minimum threshold, the gateway accepts all incoming packets. When the average drop probability exceeds the maximum threshold, all arriving packets are dropped. Packets are randomly discarded with dropping probability P when the average blocking probability is between the two thresholds.

After obtaining results for packet discard based on buffer size, a slight modification of this scheme was investigated in which the random discard thresholds are based on the packet blocking probability. One of the issues investigated during these simulations was whether all dropped packets (those dropped randomly and those dropped off the tail due to buffer overflow) should be used in the calculation of average blocking probability. Simulation runs using all dropped packets to calculate the average blocking probability showed that this approach is not suitable. The reason is that once the buffer overflows in this scheme, the packet drop probability never drops below the threshold and keeps on rising. This results in all packets being rejected by the gateway once a batch of packets has been rejected. Clearly this is not the approach to follow and was not pursued further.

4.6 Analysis of Re-transmitting Sources

Before presenting results of RED gateway performance for retransmitting sources, TCP's retransmission and congestion control mechanisms will be briefly described as they have a major impact on gateway performance.

TCP is a reliable transmission layer protocol. It relies on each end point acknowledging that data has been received from a source. The packets lost during transmission are detected by means of a timer called the retransmission timer. This timer is used when expecting
an acknowledgment from the other end of the TCP connection. Whenever the time-out for acknowledgment of a transmitted packet expires, the protocol assumes that the packet has been lost and retransmits the lost packet. At the same time, apart from retransmitting the lost packet, TCP also puts into place its congestion control mechanisms. These consist of the "slow-start" and the "congestion-avoidance" mechanisms.

The slow-start mechanism is used by the TCP sources to initiate data flow across a connection. Once the connection is up and running, slow-start is used in conjunction with the congestion-avoidance mechanism to prevent and recover from a congested link. At the onset of congestion, the size of the TCP sliding window is reduced to one. The slow-start mechanism then takes over once again and the size of the sending window is incremented each time an acknowledgment (ACK) is received. Slow-start causes an exponential increase in the window size such as one, two, four etc. When half the size of the "congestion window" is reached, the additive congestion-avoidance mechanism takes over and the window size is increased at rates lower than those used during the "slow-start" phase. TCP uses two windows to determine how many packets can be sent before waiting for an ACK from the receiver. The first is the "congestion window" and is determined by the sender. The second is the "advertised window" which is set by the receiver and used for the purpose of flow control. The sender transmits up to a minimum of the congestion window and the advertised window before waiting for an ACK.

This congestion-avoidance and recovery mechanism of the protocol can cause synchronisation in traffic. As previously mentioned in Chapter 2, this has lead a number of authors [19, 20, 21, 23] to propose ways to improve gateway performance by preventing traffic synchronisation. Having already investigated the RED scheme for non-retransmitting sources, it will be shown in the following Sec-
tions that the RED scheme does not give rise to any significant improvements for the retransmitting TCP sources either.

### 4.6.1 RED Gateways and Traffic Synchronisation.

If a limit is placed on the gateway buffer, the drop-tail gateway starts discarding packets when the buffer overflows. This is shown in the Figure 4.9 below. The plots were obtained using the parameters values included in Table 4.3.

**Table 4.3** The gateway parameters for synchronized buffer.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>0.001</td>
<td>Sets Gateway Utilisation</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>100 Pkts/sec</td>
<td>Packet Size of 1K bytes</td>
</tr>
<tr>
<td>Minimum Threshold</td>
<td>N.A.</td>
<td>Not Used</td>
</tr>
<tr>
<td>Maximum Threshold</td>
<td>N.A.</td>
<td>Not Used</td>
</tr>
<tr>
<td>Window Size</td>
<td>N.A.</td>
<td>Not Used</td>
</tr>
<tr>
<td>Weightage Value</td>
<td>N.A.</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

**Figure 4.9** Drop Tail Gateway and Traffic Synchronization
The plot in Figure 4.9 shows the gateway buffer occupancy for a limited buffer size. The measurements have been taken while the gateway was subjected to traffic which would cause no traffic synchronization if its size was not limited. It can be seen from the results that when a limit is placed on the gateway buffer size, synchronization of traffic occurs and the gateway remains idle for a period of time. This behaviour is due to the congestion control algorithm of TCP and can be explained as follows.

All TCP sources initiate the data transfer phase with small windows and gradually increase it in accordance with TCP's "slow-start" congestion control mechanism which has been described earlier. The gateway has sufficient capacity to handle small window sizes. Due to successful transmission of packets, the sources keep on increasing the transmission window size which causes a rise in buffer occupancy. This is apparent from the rising ramp behaviour of the buffer occupancy plot in Figure 4.9. As the window size of each TCP source begins to increase, a point is reached when the buffer starts to overflow. As all the source transmission windows have opened up by this point, packets from each of the sources get discarded by the gateway. This causes all the sources to reduce their congestion windows and start all over again with a window size of one in accordance with TCP's "slow-start" mechanism. This phenomenon is known as traffic synchronization.

The next set of results investigate the efficacy of RED gateways in preventing traffic synchronization. Two average packet blocking probability schemes are used to test the RED gateways. One uses the "total average" scheme (window size of infinity) and the other uses a "sliding window" averaging scheme with ten sampling windows.

In order to investigate if the size of the average window sizing has an impact on the RED gateways, the next set of simulations are run
with an infinite averaging window size. Table 4.4 lists the parameters used and the results are shown in Figure 4.10.

Table 4.4 The parameters for infinite averaging window size

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>0.3 sec</td>
<td>Total of 12 sources</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>100 Pkts/sec</td>
<td>Packet Size of 1K bytes</td>
</tr>
<tr>
<td>Minimum Threshold</td>
<td>0.01</td>
<td>Other values tried as well</td>
</tr>
<tr>
<td>Maximum Threshold</td>
<td>0.1</td>
<td>As above</td>
</tr>
<tr>
<td>Window Size</td>
<td>N.A.</td>
<td>Not Used</td>
</tr>
<tr>
<td>Weightage Value</td>
<td>Variable</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.10 shows the throughput obtained for RED and drop-tail gateways using infinite averaging window. The input traffic has been chosen such that it would cause the sources to synchronise their traffic under drop-tail gateway conditions as shown in Figure 4.9. The random discard thresholds, given in Table 4.4, have been
set such that the upper threshold is high enough to utilise the full length of the buffer and the lower threshold small enough to let the random dropping of packets have it's effect. Setting the upper thresholds value too low effectively results in a reduction of the usable gateway buffer. This can lead to most of the packets getting discarded randomly rather than dropped due to buffer overflow. On the other hand, choosing the lower random discard threshold too close to the upper threshold results in not enough random discards, rendering the RED scheme ineffective.

It can be noticed from Figure 4.10 that the throughput remains more or less constant as the RDW and hence the dropping probability increases.

An improved average packet dropping probability algorithm utilizing sliding windows is next investigated to observe any effects this might have on the RED gateway performance.

![Figure 4.11 Throughput Vs Drop weightage (Moving Window Average)](image-url)
The input parameters for the simulation are kept the same as those used to obtain results shown in Figure 4.10 except for the size of the averaging window. The results obtained (with 95% confidence intervals) for the moving window averaging scheme are shown in Figure 4.11. Note that the throughput of the RED gateway is within 1% of the drop-tail gateway throughput. As a result, it can be concluded that no noticeable improvement in throughput can be observed as the packet discard probability is increased. Similar results were obtained for a range of different dropping threshold values and have not been shown here in order to avoid duplicity.

Based on results shown in Figure 4.10 and Figure 4.11, it can be seen that regardless of the averaging technique used to calculate packet discard probability, RED gateways do not give any significant performance improvement over drop-tail gateways. The interesting case of RED gateway performance with the server idle-time is examined next.

4.6.2 RED Gateways With Server Idle Time

The next set of simulations examines the RED gateways with idle server. At low utilisation, the gateway server has some idle time. The buffer occupancy of a gateway with the server remaining idle some of the time is shown in Figure 4.12. The plot has been obtained using the gateway parameters listed in Table 4.5.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>0.01</td>
<td>Total of 12 sources</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>250 Pkts/sec</td>
<td>Packet Size of 1K bytes</td>
</tr>
<tr>
<td>Minimum Threshold</td>
<td>0.0025</td>
<td>A number of values tried</td>
</tr>
<tr>
<td>Maximum Threshold</td>
<td>0.01</td>
<td>As above</td>
</tr>
<tr>
<td>Window Size</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Weightage Value</td>
<td>Variable</td>
<td></td>
</tr>
</tbody>
</table>
The results shown in Figure 4.12 have been obtained by setting the buffer size to a large value and subjecting the gateway to lower utilisation. Selection of a large buffer size ensures that no packets are discarded by the gateway due to buffer overflow. Depending upon the gateway packet service rate, Figure 4.12 shows that the gateway remains idle for periods of time. This has the effect of reducing the throughput of the gateway. If higher loads are now applied, the gateway server never goes idle as there is always a packet in the buffer waiting to be served. Figure 4.13 shows the gateway buffer occupancy at higher utilisation (service rate reduced to 100 pkts/s) and still with a large buffer size. The rest of the parameters are as listed in Table 4.5.
Results shown in Figure 4.13 indicate that no synchronisation can be detected in the traffic. This can be explained as follows. Due to the selection of a large enough buffer size, few packets get rejected at the gateway assuming none get lost in the transmission links. Link loss rates have been set to zero in the simulation model in order to ensure this. As a result of few packets being discarded, the TCP sources do not have to shrink their transmission windows at the same time as dictated by the slow-start congestion control mechanism. This prevents any traffic synchronisation of the sources.

Next, the throughput of a RED gateway with idle server time is examined. Figure 4.14 below shows the throughput plots obtained with 95 percent confidence intervals. The gateway parameters for the simulation are as listed in Table 4.5.
It can be seen from Figure 4.14 that the utilization of the RED gateway drops as the packet discard probability is increased. Note the interesting result that the throughput obtained from the RED gateways in the case of server remaining idle some of the time is lower than that obtained from drop-tail gateways under the same conditions. The reason for this is that as the RED gateways reject more and more packets before the buffer has a chance to fill up, the more the TCP sources back off resulting in lower throughputs.

4.7 Conclusion

This Chapter has analysed the performance of RED gateways which have been proposed in literature. A simulation model developed for investigating the performance was used to study the performance of the gateways. Both non-retransmitting and retransmitting sources
were analysed under varying traffic conditions. It was shown that in a real network, the RED gateways do not give any significant improvement in performance and can in fact result in a performance degradation if either the thresholds or the RDW are not carefully selected. To summarise, the following observations were made in this Chapter:

1. The RED gateways do not give any significant performance improvement of the gateway for both the non-retransmitting and retransmitting sources.

2. When "total average" or the "sliding window average" calculation techniques are used, the RED gateways provide a slight improvement over drop-tail gateways under carefully selected conditions but this improvement (of the order of 1 percent) is not significant enough to warrant their use.

3. The RED gateways actually result in a degradation in throughput for gateways with some server idle time.

A scheme will be proposed in the next Chapter which results in improved throughput and packet blocking performance of a gateway.
5. Intelligent Packet Discard

5.1 Introduction

It was shown in Chapter 4 that the random discarding of packets based on average packet blocking thresholds does not result in any significant improvement in the performance of the gateway. It was also shown that gateway with idle times actually suffers a degradation in performance when the RED scheme is implemented. In this Chapter, a scheme for improving gateway performance called intelligent packet discard will be presented that relies on the ability of the gateway to monitor the incoming packets and selectively discard them.

IP segments the data passed down from higher layers into segments whose number and size depends on the physical transmission media. It will be shown that if an IP fragment of a TCP packet is discarded by the gateway, then discarding all following fragments belonging to the same TCP packet results in a much improved gateway performance. Note that TCP has no means of retransmitting part of a sent packet and once the packet loss has been detected by time-out, the whole packet is retransmitted. This Chapter addresses the following issues:
1. Examine how sources using TCP for data transmission implement congestion and flow control. TCP's response to retransmission of lost data is also studied.

2. The issue of IP fragmentation is discussed.

3. Based on the knowledge gained from the above two, an intelligent packet discard scheme is proposed.

4. The performance of intelligent packet discard gateway, in terms of throughput and percentage packet loss, is investigated for IP traffic.

The layout of the Chapter is as follows. A brief overview of the TCP/IP protocol is provided in Section 5.2 and a model presented for analysing the performance of IP gateway using the proposed intelligent packet discard methodology. Section 5.3 describes segmentation of data by IP into smaller fragments based on MTU. A scheme to improve IP gateway congestion called "Intelligent packet discard" is proposed in Section 5.4. Gateway performance results obtained from the simulations are presented and discussed in Section 5.5. Finally, Section 5.6 concludes the Chapter.

5.2 Data Transmission Using TCP/IP

As previously discussed in Chapter 2, there have been attempts to improve the performance of IP gateways by different authors with varying results.

This Chapter presents a selective packet discard mechanism for the IP gateways. This Section explains the protocols involved and points out the parameters which can be used by the gateway to implement the intelligent packet discard scheme.

The TCP/IP protocol being studied allows computers to communicate with each other even though they might be running on different
platforms and operating systems. It is also the basis of the Internet and it is due to this wide acceptability and usage that TCP/IP gateways have been chosen to study the proposed intelligent packet discard scheme. The data transfer in the network and data arriving at the IP gateway can best be described by Figure 5.1.

![Figure 5.1](image)

**Figure 5.1** The data arriving at the gateway

Each of the data layers in Figure 5.1 performs a different function during the transmission of data. In the simulation model, the link layer emulates the hardware card (e.g. Ethernet card) and also incorporates within it the device driver. On top of the link layer sits the network layer which in this case is IP. The network layer does not ensure reliable transmission of data and is more concerned with transporting a packet from source to destination by routing. The transport layer, TCP in this case, sits between the application and the network layer and provides functions such as retransmission and reliable transmission of data.

It was mentioned in Chapter 4 that TCP is a complex transmission protocol and there is a need for proper simulation modelling if any
valid results are to be obtained. Leaving out some features of the data transmission protocols from the simulation model can lead to erroneous results. As a result, a full implementation of TCP has been used in the simulation model. The gateway model used is shown in Figure 5.2 below.

![Intelligent Packet Discard Gateway](image)

**Figure 5.2** The TCP/IP Gateway setup used to study the intelligent packet discard scheme.

In the simulation model, each work-station feeding into the gateway runs two applications which use TCP to send and receive data from one of the other work-stations via the IP gateway.

Each work-station is modelled as shown in Figure 5.3 below:
The arrows in Figure 5.3 above indicate the direction of flow of data. Note that the ARP module is only required for outgoing data. Each workstation is assigned an IP address with the physical transmitter/receiver acting as the physical layer. All packets received by the workstation via the receiver "phy_rx" are examined for their destination addresses. If the packet is meant for the local workstation, it's header is striped and it is sent up to the TCP layer on the data stream connecting IP to TCP. At the gateway, all arriving packets are meant for a remote station and get passed down to the gateway's physical layer and transmitter on the appropriate data stream.

Whenever an application source transmits data, it gets sent down the protocol stack. Each lower layer of the stack adds its own headers to the data passed down and these headers get stripped as the data moves up the protocol stack to the application running at the
The TCP layer attaches the header shown in Figure 5.4 to all packets before passing them down to the IP layer.

<table>
<thead>
<tr>
<th>Source Port Number</th>
<th>Destination Port Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td></td>
</tr>
</tbody>
</table>

- `header`: the header contains the source and destination port numbers, sequence and acknowledgment numbers.
- `reserved`: the reserved field is currently unused.
- `length`: the length of the header.
- `Window size`: the window size indicates the amount of data that the receiver is prepared to accept.
- `TCP checksum`: the checksum is used to detect errors in the transmission.
- `Urgent Pointer`: the urgent pointer indicates a need to forward data quickly.

**Figure 5.4** TCP Header

The unit of data sent by the TCP layer to the IP layer is called the "segment" while the data sent down by the IP to the link layer is referred to as the "datagram". A checksum is maintained in the TCP header and a corrupted packet is simply discarded by the sources upon reception. The source then relies on the retransmission mechanism in the transport layer to detect packet loss and retransmit the packet. Each end of an established connection has a finite amount of buffer space (receive window in TCP). The receiver only allows the sender to send as much data as it can handle without overflowing the buffer thus exercising flow control.

The sequence number field in the TCP header identifies the first byte in the stream of data from the sending TCP to the receiving TCP. The acknowledgment number is the sequence number + 1 of the last successfully received byte. TCP provides a full duplex data transmission service so that the data can flow in each direction.

TCP can be described as a sliding window protocol without selective or negative acknowledgments. This means that there is no way to acknowledge selected pieces of data stream. Thus whenever a TCP packet is lost, the whole segment needs to be retransmitted.
Note from the TCP header shown in Figure 5.4 that none of the parameters can be used to detect if part of the TCP packet has been lost in transmission. This situation can arise if IP divides the data passed down by TCP into smaller fragments. This will be discussed in more detail when IP fragmentation of data is discussed later in the Section 5.3.

IP attaches its own header to the TCP packet. The eventual format of the datagram passed down by IP to the link layer is shown in Figure 5.5.

![Figure 5.5](image)

**Figure 5.5** Data encapsulation by IP of a TCP packet

All IP data meant for remote sources passes through the gateway which implements the intelligent packet discard scheme described later in the Chapter. The gateway buffer is limited in size and packets are rejected if the buffer overflows. Note that no limit has been placed in the simulation model on the IP buffer at the work-stations as the study of work-station bottlenecks is beyond the scope of this thesis.

Since the modelled gateway is an IP gateway, the data only goes up to the network layer as shown in Figure 5.1. The intelligent packet discard scheme proposed in this Chapter is implemented at the IP level. This is an added advantage because no higher layers need to be implemented at the gateway. This results in a reduction in the time delays due to data processing at the gateway.

Note that none of the parameters in the TCP header can be used to selectively discard packets from an IP gateway. As a result, if an intelligent packet discard scheme is to be implemented, the next layer down in the protocol stack (IP), would have to be utilised. The
following Section examines the suitability of IP to implement intelligent packet discard and also discusses the issue of IP fragmentation of data.

5.3 IP Fragmentation

Most physical layers in networks impose a limit on the maximum data size that can be transmitted. For example, Ethernet and 802.3 layers limit the maximum encapsulation size to 1500 and 1492 bytes respectively. This maximum data size limit of the physical layer is called the Maximum Transfer Unit (MTU).

IP is widely used in conjunction with TCP for data transmission and fragments the TCP packets into smaller datagrams whose size and number depends on the MTU. This fragmentation results in datagrams which are smaller than the MTU, making it possible for the link layer to transmit the data. Some typical MTU values are shown in Table 5.1.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>MTU (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperchannel</td>
<td>65535</td>
</tr>
<tr>
<td>16 Mbits/sec Token ring</td>
<td>17914</td>
</tr>
<tr>
<td>4 Mbits/sec Token ring (IEEE 802.5)</td>
<td>4464</td>
</tr>
<tr>
<td>FDDI</td>
<td>4352</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1500</td>
</tr>
<tr>
<td>IEEE 802.3/802.2</td>
<td>1492</td>
</tr>
<tr>
<td>X.25</td>
<td>576</td>
</tr>
<tr>
<td>Point-to-Point (Low delay PPP)</td>
<td>296</td>
</tr>
</tbody>
</table>

When two sources on the same network communicate with each other, the MTU of the network plays a crucial role. On the other hand, if the communicating hosts are exchanging data across multiple types of networks, then each link can have a different MTU. As
an example, consider the AWANET 100 network described in Chapter 1 where data gets communicated via either the ethernet or the TDM backplanes across an FDDI II LAN to a 2Mbps ISDN link which in turn is connected to another FDDI II LAN. Under such circumstances, the TCP segments are divided up into smaller fragments by the IP layer at each successive IP point, depending on the MTU of the next link. In such a case, the smallest MTU of the data link plays an important role as explained below.

It was mentioned in Chapter 2 that while there are proposals in literature which describe methods to determine the smallest MTU along the data path [25], there are disadvantages associated with such schemes. One major disadvantage of using the smallest MTU to transmit all data is that the connection fails to take advantage of the larger packet sizes that can be transmitted by other media along the path. Another point to note is that the path MTU between any two communicating nodes need not be constant but can vary depending on the network route being adopted.

The diagram of an IP datagram divided into four smaller fragments is shown in Figure 5.6 below.

![Figure 5.6 IP datagram and fragmentation](image)

The reassembly of the fragmented datagrams is performed at the destination hosts which maintain a list of the received IP fragments. As the IP fragments travelling over the network can arrive in any arbitrary order, each fragment is examined for its "ident" field in the
IP header to determine which datagram they belong to. Once a datagram has been reassembled from all the fragments, the data is passed on to higher layers such as TCP after stripping off the IP header. This property of IP to segment data into smaller datagrams, depending on the physical layer MTU, forms the basis of the intelligent packet discard scheme described in the following Section.

5.4 Intelligent Packet Discard Gateway

The previous Section showed how TCP passes data down to the IP layer for transmission. TCP provides the reliable transmission of data and detects packet loss whereas IP does the routing and interfaces with the link layer. In the simulation model used to study intelligent packet discard, only the sources implement the TCP layer whereas intermediate nodes and gateways only go up to the IP layer. There is no need for the protocol stack at the intermediate nodes to go all the way up to TCP as this would impart a processing delay on each passing packet. If any performance improvement in the IP gateways is to be made, then care should be taken that only information available at the network layer (IP) is used.

It will be shown in the following Section that the intelligent packet discard gateway being proposed in this Chapter not only provides significant improvement in gateway performance in terms of increased throughput, it also results in reduced packet loss and congestion. It will also be shown that the implementation ensures that the protocol stack at the gateway does not go above the network (IP) level. This means that the intelligent packet discard scheme only requires information contained in the IP header to work resulting in less processing overhead.

The intelligent packet discard scheme relies on the fact that if a TCP segment is broken down by the IP into smaller fragments and one of those fragments gets rejected by the gateway, then any other IP frag-
ment arriving at the gateway and belonging to the same TCP packet can be rejected by the gateway. Any fragments belonging to a rejected packet which are already in the gateway buffer are served normally but no new fragments belonging to that packet are accepted. The reason for not discarding the already buffered fragments is that it would impart an extra overhead which can lead to a deterioration in packet service time and hence gateway performance. Figure 5.7 shows the implementation of intelligent packet discard gateway.

![Figure 5.7 The intelligent packet discard gateway](image)

All datagrams arrive at the gateway with the IP header attached. The fields of the IP header are shown in Table 5.2 below. The "ident" field in the header shown Table 5.2 has a unique value for each datagram that gets fragmented by IP. Thus, all fragments arriving at the gateway from a source and having the same value of the ident field belong to one single IP datagram which has been fragmented. Note however that fragments arriving from other sources can have the same fragment number. Thus when selectively discarding the packets from the gateway, care needs to be taken in identifying the source of the data stream. Note from Table 5.2 that this information
is easily obtained from the IP header using the “source net” and the “source node” fields which uniquely identify a particular data source.

As a result, a combination of “source node” and “ident” fields uniquely identify a packet and therefore all fragments have the same “source node” and “ident” fields can be safely discarded. This includes the last fragment belonging to the packet as well.

The intelligent discard gateway algorithm can be written as shown in Figure 5.8.

```c
if (fragment_ident == Discard_Table[source_identifier].lost_fragment)
{
    Discard the fragment
}
else
{
    if (packet dropped due to buffer overflow)
    {
        Update Discard_Table
    }
    Queue in gateway buffer
}
```

**Figure 5.8** The intelligent packet discard gateway algorithm
The discard table mentioned in the intelligent packet discard algorithm keeps track of the fragments of a packet that have previously been discarded by the gateway due to buffer overflow. In order to conserve memory, once a packet has been completely discarded, the entry can be over-written and reused.

The next Section presents some performance results obtained for the intelligent packet discard gateway. Gateway performance for both small and large sized transport level PDUs will be examined. Also, the results obtained will be compared with the “non-intelligent” drop-tail gateway to obtain a measure of performance improvement.

5.5 Analysis of Results

The aim of this Section is to compare the performance of the intelligent packet discard gateways with normal drop-tail gateways. The first batch of simulations are run so that an upper bound on the improvement in performance could be obtained. In the first scenario, the intelligent discard gateway was studied under the conditions which caused the IP to fragmented the transport layer PDU into six IP fragments. The throughput at the sources under these conditions was measured. Another batch of simulation was run next, keeping all other variables constant and with the IP layer not fragmenting any packets passed down by the transport layer. The results for both cases were obtained with 95 percent confidence intervals and are plotted in Figure 5.9 below.
The non-fragmented traffic describes the upper bound of the gateway performance due to the fact that the IP datagram only gets accepted into the gateway buffer if it can accommodate the whole transport layer PDU. If the whole packet cannot be accommodated due to insufficient buffer space, it gets discarded and the source retransmits after a time-out. The non-fragmented traffic is the best possible scenario since the source need not transmit any datagrams which cannot be accepted by the gateway. Another reason the non-fragmented traffic defines the upper bound is that in the case of fragmented traffic, the gateway might already have accepted some of the fragments belonging to a packet before one gets discarded by the gateway. As the gateway does not discard fragments already queued up in the buffer, the gateway capacity being used by such fragments is wasted in the case of fragmented sources.

**Figure 5.9** Throughput comparison of fragmented and unfragmented sources
Figure 5.9 has been obtained by plotting the throughput versus the applied load to the gateway, where the applied load is measured in terms of gateway utilization. It can be observed from the plot that the fragmented and non-fragmented sources have the same throughput until the buffer overflows at applied load of around 0.6. The fragmented traffic throughput then falls rapidly whereas the non-fragmented traffic throughput keeps on increasing. This is in agreement with the expected results as explained in the previous paragraph. After attaining the peak throughput, both the traffic types then level off to a limit determined by the gateway buffer size at high utilization.

It is clear from Figure 5.9 that the intelligent packet discard gateway has the potential to outperform the drop-tail gateway. It should be mentioned that the above scenario, where non-fragmented traffic is compared to a case with packets divided into 6 fragments, is only an indicator that the intelligent discard gateways can work. An increase in the number of fragments further lowers the throughput of the fragmented packets scenario when compared to the non-fragmented case. Therefore, the non-fragmented case serves as an upper limit indicating the maximum throughput that can possibly be obtained by the sources under the traffic conditions being investigated. The performance comparison between the intelligent packet discard gateway and the drop-tail gateway are presented next.

The plot in Figure 5.10 below compares the drop-tail gateway with the intelligent packet discard gateway. Both gateways have been subjected to the same traffic conditions and 1Kbyte packets are used. The simulation parameters used are shown in Table 5.3.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>Variable</td>
<td>To change load on gateway</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>819200 bits/sec</td>
<td>Nearly 100 pkts/sec</td>
</tr>
<tr>
<td>Packet Size</td>
<td>1K bytes</td>
<td>A number of values investigated</td>
</tr>
<tr>
<td>Buffer Size</td>
<td>126000 bits</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.10 Throughput comparison of drop-tail and intelligent discard gateways

The plot in Figure 5.10 shows the throughput comparison of the intelligent discard and the drop-tail gateways with 95% confidence intervals. Note that the intelligent packet discard gateway shows improved performance over the drop-tail gateway for IP packets which have been divided into six fragments. It can be seen from Figure 5.10 that the throughput for both cases levels off at high load with the intelligent discard gateway throughput remaining higher than the drop-tail gateway. As mentioned earlier, the limiting factor on throughput is the incoming traffic and the gateway buffer size which overflows at higher loads. High traffic loads result in high packet blocking probabilities. This causes the TCP sources to reduce their transmission window sizes as the congestion control algorithms take effect. As a result, the throughput is reduced. More detail of TCP's congestion control mechanism is provided in Appendix A.
Next, the percentage of packets discarded by the intelligent packet discard gateway is compared with those discarded by the drop-tail gateway.

![Applied Load vs % packet drops](image)

**Figure 5.11** Comparison of percentage packet discards by the intelligent discard gateways and drop-tail gateways

The plot in Figure 5.11 shows the percentage of packets dropped by the two gateways under consideration with 95% confidence intervals. The term “packet” refers to the full PDU which the transport layer passes down to the IP layer for transmission. Each complete PDU discarded by the gateway is regarded as a discarded packet. The fragments following the first discarded datagram and belonging to the same transport layer PDU, i.e. with the same “ident” field in the IP header, are simply rejected by the gateway and the packet discard counter is not incremented. The reason for using this method to calculate discarded packets is that even if a single IP datagram belonging to a larger transport layer PDU is discarded by the gateway, the transport layer has to retransmit the whole packet.
Note from Figure 5.11 that the intelligent discard gateway results in around 15 percent fewer packet discards. It can also be noticed from the plots that at higher loads, the number of packets being dropped tends to level off. This, once again is due to the TCP sources cutting back on transmission rates as the TCP congestion control mechanism takes hold. Even so, the intelligent packet discard gateway outperforms the drop-tail gateways.

5.5.1 Larger Transport Layer PDUs

The results show in the previous Sections were obtained for a transport layer PDU size of 1K bytes. The effect of increasing the PDU size on the performance of the intelligent discard gateways is now examined. It will be shown in this Section that larger transport layer PDUs result in even better performance from the intelligent packet discard gateways when compared with the normal drop-tail gateway. However, for any given PDU size, the greater the fragmentation the lower the maximum throughput that can be achieved by the system under similar traffic conditions. This dependence of maximum throughput on the degree of fragmentation, is due to the fact that each fragment dropped in systems with packets divided up into a large number of fragments results in all the other fragments being retransmitted. Another reason is that the greater the number of fragments, the more significant the size of the header becomes when compared with the actual payload. This is due to each fragment having to carry it's own IP header of 20 bytes which does not count towards the throughput of the connection. Consider the results shown in Figure 5.12 below generated by using parameters shown in Table 5.4.
Table 5.4 The gateway parameters for Large transport layer PDUs

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter_arrival Time</td>
<td>Variable</td>
<td>To change gateway utilisation</td>
</tr>
<tr>
<td>Gateway Service Rate</td>
<td>3276800 bits/sec</td>
<td>Nearly 100 pkts/sec</td>
</tr>
<tr>
<td>Packet Size</td>
<td>4K bytes</td>
<td></td>
</tr>
<tr>
<td>Buffer Size</td>
<td>400000 bits</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.12 Comparison of Intelligent discard gateway and drop-tail gateway for a transport layer PDU size of 4K bytes.

Plots shown in Figure 5.12 have been obtained (with 95% confidence intervals) by running the simulation for transport layer PDUs of size 4K bytes. Note from the results shown that the difference in performance between the drop-tail and the intelligent discard gateway is much more significant than that for transport layer PDU of size 1K bytes shown in Figure 5.10. From Figures 5.10 and 5.11, the maximum average improvement in throughput, for example at applied
load of 0.65, can be calculated and is shown below for both 1K bytes and 4K bytes PDU sizes.

For a transport layer PDU of size 1K bytes:

\[
\text{Improvement} = \frac{3.7 - 3.5}{3.5} \times 100 = 6.0\text{percent}
\]

For a transport layer PDU of size 4K bytes:

\[
\text{Improvement} = \frac{1.635 - 1.249}{1.249} \times 100 = 30.9\text{percent}
\]

It can be concurred from the results above that while the intelligent packet discard gateways result in improved performance when compared to the drop-tail gateways, the improvement in performance becomes even more pronounced as the transport layer PDU size increases.

Recall that it was shown in Chapter 4 that the RED gateways do not result in an improvement in the throughput of the gateway when compared to the drop-tail method. The intelligent discard gateways proposed in this Chapter on the other hand not only results in better throughput performance of the gateway but, as shown in Figure 5.11, also results in a reduction in the percentage of packets discarded by the gateway.

5.6 Conclusion

The concept of intelligent discard gateways was presented in this Chapter. It was shown that use of TCP/IP for data transfer can result in a fragmentation by IP of the data passed down by the transport layer. The size of the IP fragments depends on the MTU size of the physical media. As the transport protocol (TCP) has no means of detecting if a single fragment has been discarded by the gateway, it
has to retransmit the whole packet. The effect of the congestion control mechanism of TCP was also mentioned and was used to explain the levelling of the throughput response at high gateway utilisation. It was shown that given the congestion control and retransmission mechanism of the TCP protocol, the intelligent packet discard gateways result in improved throughput performance when compared to the drop-tail gateways. It was also shown that the proposed scheme results in a reduction in the percentage of packets discarded by the gateway as compared to the drop-tail method when traffic has been fragmented. Finally, the results for larger transport layer PDUs were presented and it was shown that the larger the PDU, the more pronounced the improvement in performance of the intelligent discard gateways when compared with drop-tail gateways.
6. Conclusions

6.1 Overview

This thesis has concentrated on the development of a methodology for improving the performance of IP gateways used to interconnecting LANs. The main areas covered in this respect were:

- Development of a comprehensive TCP/IP LAN queuing model which implements the protocol stacks at the users and the gateway level.

- Use of the queuing network model to analyse the performance of Random Early Discard gateways


Interconnection of LANs by gateways, was described in Chapter 3 and it was shown that speed differences in the packet arrival rate into a gateway and its service rate can give rise to a bottle-neck. This leads to congestion and a degradation in throughput and quality of service. It was shown that the most common and effective interconnection of LANs happens at the network layer. LAN interconnection at the network level means the gateway has visibility of the network layer header. This also means the gateway takes active
part in routing the traffic effectively through the network. The most common network layer protocol in use within LANs is the Internet Protocol (IP). As a result, IP gateways were chosen for further study. The translation of interconnected LANs and gateways into a queuing network suitable for study was also described.

An example of a practical LAN (AWANET 100) in a WAN environment was also presented. The reason for the selection of AWANET 100 was that development of data transmission protocols for use by AWANET 100 was carried out as part of this Masters thesis program. The developed Frame Relay design as per the Q.922 specifications is attached in Appendix C.

Random early packet discard methodology for the LAN IP gateways was also studied. The RED scheme assumes a gateway is headed for congestion when the average buffer occupancy exceeds a set threshold. In Chapter 4, it was shown that this method does not give any significant improvement in performance.

It was shown that changing the RED algorithm onset thresholds and varying the sliding window size for determining the average gateway occupancy had no positive effect on performance. This was found to be true for both the non-retransmitting and re-transmitting sources. It was also found that for a gateway with idle server time (gateway having no packets to serve for some of the time) the RED algorithm actually resulted in decreased throughput as the number of packets dropped randomly increased.

Having shown that the RED scheme does not give rise to any performance improvement, the intelligent packet discard gateway was proposed in Chapter 5. These gateways work on the principle that TCP retransmits the whole packets it passes down to the IP layer even if only part of the packet is lost during transmission. This can happen when IP fragments the data into smaller data segments. The conditions under which IP performs data fragmentation were
Conclusions

described in Chapter 5. The TCP congestion control and the retransmission algorithms are briefly described in Appendix A.

Gateway performance was shown to improve when all following fragments of a packet are discarded once a single fragment belonging to the same packet had been previously discarded by the gateway. This has the effect of freeing up buffer space for the other inter LAN traffic giving rise to improved throughput. It was shown that this scheme provides improved throughput when compared with drop-tail gateways at higher gateway utilisation. It was also shown that the percentage of packets discarded by the intelligent packet discard gateway is lower when compared with a normal drop-tail gateway. The final result presented showed that as the size of the transport layer PDU increases, the performance of the intelligent discard gateway increases as well.

6.2 Future Work

In this thesis, the intelligent packet discard scheme has been studied for TCP/IP traffic. It was mentioned in Chapter 5 that the proposed scheme only selectively discards newly arriving IP fragments at the gateway and does not discard fragments which are already queued up in the buffer. This decision was taken because discarding already buffered fragments would impart an extra overhead on the gateway, which would need to keep track of each and every data segment in it's buffer. As an extension to this thesis, it is suggested that the exact impact of discarding already buffered fragments on gateway performance be studied in a quantitative manner.

Also, the intelligent packet discard gateway has been studied for use by TCP/IP traffic. With a slight modification, the same scheme can be used in ATM routers. ATM also segments higher layer data into smaller cells and if one of these cells gets discarded by the gateway, the higher layer retransmits the whole packet. The use of intelligent
packet discard scheme would be of special interest in the case of ATM carrying TCP traffic.

Finally, it is recommend that the intelligent packet discard scheme be implemented in an actual IP gateway and performance measurements obtained under real operating conditions.
References


Appendix A. TCP Congestion Control Mechanism

A.1 Outline

This Appendix describes the congestion control mechanism of TCP. This mechanism consists of "slow start" and "congestion avoidance" phase. It causes the TCP sources to slam shut their transmission windows when a packet is dropped by the network and can cause traffic synchronisation. This is of relevance in chapters 4 and 5 of this thesis.

A.2 TCP Congestion Control

In order to be able to improve the performance of a system handling TCP traffic, understanding of how the TCP handles congestion in the network is essential. In order for the transport layer to provide reliable data communications, each source acknowledges the packets it receives from the other end. However, both data packets and the acknowledgments can get lost and these are handled by the source keeping track of a number of timers such as:

- Persist timer: this keeps the window size information flowing even if the remote end closes it's receive window.
• Keep-alive timer: This detects when the other end of an idle connection dies.

• Retransmission timer: This is used to detect lost data and acknowledgments.

The re-transmission timer

Each source in the model described above basis it's retransmission timer on the round trip time (RTT). As this can change in a dynamic network with multiple sources, the sources keep track of the changes in RTT. The sources measure the RTT at frequent intervals and smooth it out using a low pass filter. The equations can be written as:

\[ R = \alpha R + (1 - \alpha)M \]  
(Eqn A.1)

Where \( R \) is the smoothed RTT, \( \alpha \) is a smoothing factor set to 0.9 and \( M \) is the measured round trip time. This results in the new value having 10 percent content in the calculation of the RTT. Each source, having calculated the RTT, now sets the timeout value at:

\[ \text{Timeout Value} = 2R \]  
(Eqn A.2)

Whenever a packet is transmitted, the source waits for an ACK to come back from the destination until the RTT expires. If an ACK has not yet arrived, the packet retransmitted and the source assumes that the link has become congested.

Transmission Windows

Each source also has a sliding window which it uses for flow control during data transfer. The sources in the simulation model have the transmission window size set at 64K. The sources increases there
transmission window size in accordance with the slow start scheme described earlier.

![Diagram of Transmission Window](image)

Figure A.1 Transmission window of source

The sliding window moves to the right as the receiver acknowledges the received packets with the size of the window at any one time depending on the acknowledgments received. The window closes as the left edge advances when data is sent and acknowledged. It opens when the right side moves to the right when the receiving source reads acknowledged data and frees up space in its receive buffer. Note from Figure 5.5 that the size of the receive buffer has been set to 8K. The left side of the window can not move to the left as it depends upon the acknowledged packets received from the receiving source. If an ACK arrives asking the window to be moved to the left, it is taken to be a duplicate ACK and ignored. A zero window condition is reached when the left edge reaches the right edge and the source then stops transmission of data until further ACKs are received.

**Congestion Avoidance**

The sources implement congestion avoidance using the following main points:

- Two windows, congestion window (cwnd) and a slow start threshold size (sthresh) are used to determine the transmission window size. This also has an impact in the throughput of the gateway.
• At the start of data transmission, the cwnd is set to unity while the sthresh is set to 64K bytes.

• The transmitting source never sends more than the minimum of cwnd and the receivers advertised window.

• When congestion is detected, sthresh is set to half the current window size and cwnd is reduced to one. This means that the source now transmits with a window size of one (minimum of the cwnd and sthresh values).

• Upon each successful transmission of a packet, the source increases the window size exponentially in accordance with slow start as long as the window size is less than sthresh. When sthresh is exceeded, the window size starts increasing in an additive rather than exponential manner resulting in a more cautious increase in window sizes.

Figure A.2 Changing source transmission window size

The above plot shows the size of the source transmission window after congestion has occurred when the ssthresh was at 32. Note the additive increase in window size starts when the ssthresh is at
half the previous value, i.e. 16, once congestion has occurred. The system goes into congestion if the service capacity of the gateway is exceeded. The capacity of the data transmission pipeline is determined by the delay-bandwidth product defined as:

\[ \text{Capacity} = \text{Bandwidth} \times \text{RoundTripTime} \]

It can be seen from the equation above that either the bandwidth or the delay can change the capacity of the pipeline allowing the window size to increase and thus having more unacknowledged packets in the data stream. This can effect the congestion of the system described in the simulation due to the fact that if a single packet is dropped by the gateway there are other packets in the pipeline which can avail the space made available in the buffer. Congestion can occur at the gateway when data on multiple input streams arrives at a rate greater than the output capacity of the gateway. During congestion, the gateway buffer overflows and incoming data is rejected.

As will be seen in the following section, if the TCP segment is further fragmented by the IP layer, then it only makes sense to drop the whole packet once the gateway has rejected one fragment. This will free up space in the gateway buffer which can then be used by other packets either in the pipeline of the same source or by other sources connected to the gateway. This forms the basis of the intelligent discard gateways being proposed in this chapter. It will be shown that the intelligent packet discard scheme working in conjunction with the normal TCP congestion control mechanism, results in improved throughput performance of the gateway.
Appendix B. Frame Relay Development: Overview

B.1 Outline

This Appendix gives an overview of the frame relay software developed as part of this masters program. The software is used for the interconnection of LANs by gateways and can encapsulate the IP packets within it. This has the effect of converting the connectionless IP protocol into the connection-oriented frame relay.

B.2 Frame Relay

All IP datagrams that are to be transported over the Wide Area Network arrive at the gateway which attaches a specific Data Link Channel Identifier (DLCI) depending upon the virtual circuit that it is to be transported on. The format of a frame relay encapsulated IP datagram is shown below.
The virtual circuit is not set for each individual user but rather for the whole AWANET-100 LAN connection to another AWANET-100 LAN. Another virtual circuit could be set up if another AWANET-100 is to be included in the WAN. The virtual circuit assignment is done using the IP addresses.

The IP header of each datagram contains addressing information. Each address is made up of two parts, the "netid" and the "hostid". Each AWANET 100 LAN is identified by an IP network address (netid) that corresponds to a DLCI entry in the frame relay call connection table. It should be noted that IP datagrams are designed specifically for connection-less transmission whereas frame relay provides virtual circuits. This method of transporting the IP packets is referred to as "tunnelling". Practical results for similar networks have demonstrated that this procedure enhances the reliability of the IP packet transmission. Once the IP packet has been encapsulated in a frame relay frame, it is transported through the frame relay network and the header removed from it at the receiving AWANET 100 LAN gateway.

### B.2.1 PVC Based Frame Relay Core

The core part of the Q.922 protocol that involves attachment of Digital Logical Channel Identifiers (DLCI) based on IP addressing has
been developed. The DL-CORE software written accepts packet streams coming from the Ethernet and the AMAC backplane.

Due to static allocation of memory, the management layer is not allowed to add more virtual circuits. In the another version also developed, dynamic memory allocation is used and the management layer has the ability to add new virtual circuits while the program is running. Interface to the other working modules of the software interacting with Frame relay software is provided by the DL-CORE-SAP. The functions used by the management layer to add new virtual circuits, as specified in the standard, have been provided in the DL-CORE-SAP. For testing purposes, the test packet generator passes the test packets on to the DL-CORE software via the DL-CORE-SAP that attaches the DLCI to it and places it in the transmit buffer.

B.2.2 Integration with Data Link Control.

As the Q.922 upper sub-layer is based on the LAPD protocol, the incorporation of the core with the upper layer is necessary. The software integration modules have been written.

The Q922_SAP function DATA_LINK_UL_INDICATION is used to pass the layer three frames down to the DL_CORE part of the software. This means that the DL_CORE part of the software relies on the upper sub-layer to set up the connection and once the connection has been established, no sequence number checking and error correction is performed at the data-link layer. This is in accordance with the standards defining IP encapsulation in Frame Relay. Software testing of the testing of the combined DL_CORE and DL_CONTROL layers making up the Q922 has been be performed by using test IP packets and a dummy packet driver.
B.2.3 Interaction with other layers

The primitives for the interaction between different layers are defined below:

DL-CORE-SAP: The DL-CORE provides services to the user DL-CONTROL at the DL-CORE-SAP.

DL-SAP: This is the point where the DL-CONTROL layer interacts with layer three in the user plane.

M2N-ASSIGN request: Is used to pass down the DLCI to be used from layer three to the management entity at layer two.

M2N-REMOVE request: Is used by layer 3 to indicate to layer 2 that the connection is to be removed.

MDL-ASSIGN request: Is used to establish the connection between the DL-CORE-CEI and the DL-CEI in the DL-CONTROL layer (The DL-CEI remains set for a circuit but the DLCI might change, hence the need to set up the association).

MDL-REMOVE request: Is used by the layer 2 management to remove a mapping between a DL-CEI and DL-CORE-CEI.

MC-ASSIGN request: Is used by the link management to signal to the DL-CORE sublayer that:

• A DL-CORE connection has been established and conveys the agreed DLCI between DL-CORE entities in support of that connection.

• Convey the associated DL-CORE CEI to be used to identify the DL-CORE connection.

• Convey the CEI used to support the DL-CORE connection.

MC-REMOVE request: Is used by the layer management to signal to the DL-CORE sublayer that a DLCI has been released.
An SNMP module has also been written. This module provides statistics on how many packets have been transmitted and received on each VCI as well as how many packets have been rejected due to virtual circuit errors. The functionality of this module is given below.

- **sysDescr**: ASCII string describing the SNMP proxy agent. Can be up to 255 characters long.
- **sysName**: Domain name of the SNMP proxy agent.
- **sysLocation**: Location of the SNMP proxy agent.

The "interface table" (ifTable) contains information on the managed resources. In accordance with the guidelines defined by the frame relay services MIB working group, the frame count per virtual connection are not covered. Instead, the information relating to the frame relay service interface is managed. Some typical entries in the ifTable are:

- **ifType**: The standard value for the frame relay service is 44.
- **ifSpeed**: Peak bandwidth in bits per second available for use.
- **ifOperStatus**: The current operational status of the Frame relay service.
- **IfInOctets**: The number of received octets. This stretches from the beginning of the frame relay header field to the end of user data.
- **IfInDiscards**: The number of received frames discarded. Includes frames discarded due to policing and congestion.
- **IfInErrors**: The number of frames discarded due to an error, such as invalid DLCIs, incorrect length etc.
- **IfOutOctets**: The number of transmitted octets.
- **IfOutUcastPkts**: The number of frames transmitted.
• IfOutDiscards: The number of egress (outward bound) frames discarded due to congestion etc.

• IfOutErrors: The number of frames discarded in egress direction due to transmission problems.
Appendix C. Outline of ATM Driver Software for SARA-S and SARA-R Chips

C.1 Outline

Chapter 3 outlined how AWANET 100 LANs use ATM for transmission of data to the LAN gateway. This Appendix describes the functionality of the Transwitch segmentation and reassembly chips. The software used to drive the SARA chips was developed as part of this thesis project but has not been included as part of the thesis due to proprietary constraints.

This Appendix described how to setup the SARA chips at system start up to enable them to receive and transmit cells as well as perform segmentation and reassembly. These chips are powerful independent chips that work in pairs and perform all the functions required for the segmentation and re-assembly of data packets with minimal load on the central microprocessor of the interface card.

The software runs on the MC68302 microprocessor and use the SARA-S and SARA-R chips. These chips are memory mapped to the microprocessor and the mode of operation of each chips is set by programming the MODE registers contained within the chips. The
interface to the PACKET and CONTROL memories is via appropriate chip. The SARA chips interface with the memory in the non-parity checking mode.

**C.2 SARA-S Software Interfaces.**

The interface to other software modules is provided by means of an API class. This class provides functions to initialize the driver as well as starting segmentation and reassembly of the passed packet. Status report is also be provided indicating whether the packet transmission was successful or not. The interface of the modules are shown in the Rose Tool design.

**Start Up Initialisation.**

Packet Memory Buffer Allocation.

The maximum size of any packet buffer is 65535 bytes. These buffers are set up at initialisation or when a packet is setup for segmentation. The important thing is to estimate the number of buffer descriptors needed in the control memory.

Buffer Descriptor Allocation and Initialisation.

Each packet buffer has a buffer descriptor associated with it. If the buffers in the packet memory are allocated at packet setup, then the packet start address field in the buffer descriptors can be set as part of packet setup. All other bits in the descriptor are set to zero.

Virtual Circuit Table Allocation.

The entries in the VC table are accessed by the VC index. Since the entries are programmed when a VC is established, this table need not be written to at initialisation. The VC_BASE register however is programmed to hold the value of the base address location of the VC table.
Communications Queue Setup.

Each queue must be large enough to hold all the buffer descriptors. The base location of these queues is programmed in the QUEUE_BASE register and at initialisation, the read and write pointers of both the queues point to the start addresses of the queue. The transmit complete queue be initialised with all the descriptors that are to be used.

Peak-Rate Metering Parameter Setup.

Each of the eight rate queues has a rate counter and an associated rate register. Whenever the rate counter overflows, a cell is transmitted from each of the packets queued on the rate queue. The rate queue register is initialised to realise the peak rate of that queue. The PRESCALER bits in the rate queue register determine the clocking interval of the rate counter. Another eight bits of the register are used to reload the rate counter. The rate queue is enabled by setting the RQ_ENABLE bit of the rate queue register.

Mask Bit Initialisation.

Upon reset, all mask bits in the mask bit register default to 1s. To enable an interrupt from the interrupt register INTR_STATUS_REG, the corresponding mask bit is set to 0.

Mode Register Setup.

The mode registers MODE_REG_0 and MODE_REG_1 are initialised in accordance with the system design. The SARA-S is then be placed on line.

Virtual Circuit Setup.

To establish a virtual circuit, the following fields in the VC table need to be initialised. All reserved fields are set to zero.

- The ATM field header.
• The VC mode bits and the cell quotas for the rate queue to be used,

• The metering parameters and the congestion control characteristics.

**Packet Transmission.**

The free buffer descriptor is obtained from a list maintained by the software (or the transmit complete queue). The following fields are then set in the buffer descriptor before packet segmentation begin.

• Descriptor mode bits with bits 7 to 0 set to zero.

• VC index. 0 if not used.

• Packet byte count which is a multiple of four bytes.

This buffer descriptor is then written to the packet ready queue as described earlier. The SARA-S returns the descriptor number via the transmit complete queue and generates a maskable interrupt.

**OAM Cell Transmission.**

To transmit an OAM frame, the following fields are set in the buffer descriptor.

• Set descriptor mode bit APP_CRC 32 = 0

• Set descriptor mode bit PKT_TYPE = 10/11

• Set the PTI value if PKT_TYPE = 11

• VC index

• Packet byte count = 0x30

**CBR Cell Transmission.**

The CBR cell is first loaded into the memory location pointed to by the CBR_ADDR_HI and CBR_ADDR_LO registers. The CBRXMIT signal is then activated to transmit one CBR cell. The cell can also
be transmitted by setting the SEND_CBR bit in the Mode register 1 of the SARA-S.

C.2.1 Control Memory.

Buffer Descriptor Table.

This table is made up of a set of 32 byte entries. The entries describe how the packets are to be fragmented and their location in the packet memory. A buffer descriptor is initialised for each packet ready for segmentation. Up to 8191 buffer descriptors can be maintained at any one time. Address DESC_BASE stores the upper six bits of the base location of the buffer descriptor table. Each entry contains:

- Descriptor Mode Bits
- VC table Index
- MID
- Packet Byte Count
- Packet memory start address (High)
- Packet memory start address (Low)

The mode bits describe the packet type such as AAL3/4, AAL5 or OAM. The VC table index determines the VC to be associated with the packet. This is used as a pointer to the VC table entry corresponding to the VC circuit. The MID is used by the AAL3/4 layer. The packet byte count is programmed with the length in bytes of the packet. The packet start address is calculated from the high and low start addresses.

Virtual Circuit Table.

This is a set of 16 byte entries and is used by the SARA-S to generate the cell headers. It also contains the average metering and the congestion control functions of each virtual circuit. The upper 11 bits
are stored in the virtual circuit base address register VC_BASE. The location of each entry is determined by concatenating the base address with the VC index value of the buffer descriptor. Each entry contains:

- ATM Header 0
- ATM Header 1
- ATM Header 2
- ATM Header 3
- Virtual Circuit Mode Bits
- Cell Quota

The ATM header 0/1/2/3 is written to conform to the ATM standard as the SARA-S uses these without modification and attaches them to the cell as the first 4 bytes.

The VC mode bits define the FCS and congestion control setting for the VC. The CC defines the number of successful cell transfers that must take place before the cell transmission rate is restored. The mode bits also define which one of the eight rate queues the VC is policed with. The average metering of the queues is based on the 'leaky bucket' algorithm. The rate of transmission on each VC is a fraction of the rate queue it is associated with where the fraction is determined by the Time Interval Quota (TIQ) field in the mode bits and the Cell quota field of the VC table.

**Average Metering Function.**

The desired peak rate of a queue is determined by the corresponding RQ_REG_A(0:3) and RQ_REG_B(0:3). As long as the number of cells transmitted in the averaging interval defined as:

\[(\text{TIQ}+1) \times (\text{CQ} \times 32) \times 424 / \text{Peak-Rate}\]
is less than the maximum credits allowed, the cell is transmitted at the peak rate by the SARA-S chip.

Packet Queue Ready.

This queue holds the buffer descriptors of packets ready for segmentation. It is defined by four registers: PRQ_ST_ADR, PRQ_ED_ADR, PRQ_RD_PTR and PRQ_WR_PTR. These hold the start address, end address, read pointer and end pointer of the packet. The read pointer points to the next location where a buffer descriptor can be found by the SARA-S for segmentation and the write pointer points to the location where the microprocessor places the descriptor of the next packet that becomes available for segmentation.

Transmit Complete Queue.

This is used by SARA-S to return the buffer numbers of packets that have been segmented. The microprocessor reads this queue upon receiving an interrupt. It is defined by four registers: TCQ_ST_ADR, TCQ_ED_ADR, TCQ_RD_PTR and TCQ_WR_PTR. Buffer descriptors starting from the read pointer up to the write pointer is read. This will provide all the descriptor numbers released by the SARA-S since the last time the queue was read.

**C.2.2 Packet Memory.**

Packet Data Buffers.

These hold the packet data that is to be segmented and the starting address is specified in the buffer descriptor of the packet. The data must be 32-bit word-aligned.

OAM (Test) Cells.

This is a special case of a normal AAL5 data type.

CBR Data.
The location of CBR data is determined by the CBR_ADDR_HI and
CBR_ADDR_LOW registers of the SARA-S. The first two bytes of the
data provide the VC index to obtain the VC for CBR data transmis-
sion.

C.3 SARA-R Software Interfaces

Access to the SARA-R driver is provided by means of an API class.
Status reports for the reassembled packets as well as the chip will
also be provided if required.

Initialisation.

Packet Memory Buffer Allocation: The small and large buffers are
initialised at start up in the packet memory.

Buffer Descriptor Allocation and Initialisation: A Buffer Descriptor is
allotted for each packet buffer

Virtual Path Table Allocation: The entries in this table point to the
appropriate location in the reassembly table.

Virtual Circuit Table Allocation: All entries is initialised to the appro-
priate reassembly table. Each entry is indexed by the VC index of
the received cell.

Reassembly Table Allocation: This will always be a multiple of 256
entries.

Virtual Circuit Table Setup.

Virtual Path Table Setup.

Reassembly Table Setup.

All AAL3/4 entries are initialised to 0, all AAL5 is initialised to
2000H, all CBR entries is initialised to C000h and all raw cell cir-
cuits is initialised to E000H.
Communications Queue Setup: This is similar to the SARA-S queue.

Mask Bit Initialisation: To enable an interrupt from the SARA-R, the corresponding mask register bit must be set to 0.

Mode Register Setup.

The mode registers is initialised and SARA-S brought on line.

Receiving a Cell.

VPI field is read from the incoming cell and the appropriate entry in the VPI/VCI table checked. For valid cells, the appropriate entry in the reassembly table is read and the cell processed as indicated by the entry.

Receiving a Packet.

A buffer descriptor is assigned for each packet received. The arrival of a new packet is indicated by the BOM field. All subsequent cell arrivals are processed as above. The SARA-R then reassembles the packet, writing the VC index into the VC index field of the Buffer Descriptor. Upon reassembly the Buffer Descriptor pointer is placed in the packet received queue.

Receiving CBR traffic.

The destination of the CBR cells can be a real FIFO.

C.3.1 Control Memory

Buffer Descriptor Table.

This table is made up of a set of 32 byte entries with each entry having the following fields:

- Descriptor mode bits
- VC index
- MID/VP
• Packet byte count
• Packet memory start address high
• Packet memory start address low
• DMA address high
• DMA address low
• Residual CRC upper
• Residual CRC low
• Packet timeout

The mode bits hold the error status of the reassembled packet. The packet byte count is loaded from the BOM cell and counted down to detect the end of the packet. The DMA field contains the pointer to a memory location immediately following the end of packet. The packet time out field is used to maintain the age of the packet.

Reassembly Table Pointer: To generate the descriptor table address the following steps is followed:

• Determine weather VPI/VCI or VCI/MID fields are used for address generation.
• Generate control memory address for VPI/VCI table entry.
• Extract reassembly pointer from the VPI/VCI table entry. Generate reassembly table address.
• Extract descriptor number from reassembly table entry and generate descriptor table address.

Virtual Circuit Table.

With the bit 0 of this entry set to zero, the 16 bits in the entry point to the reassembly table location. If the bit is set to 1, then the lower
3 bits are MID bits. If the lower 3 bits are set to 111, then the cells arriving on this VC is discarded.

Virtual Path Table: Similar to the VC table.

Reassembly Table: This is made of up 2 byte entries containing information related to each VC as well as the buffer descriptor address.

Small/Large Buffer Descriptor Queues

These hold the incoming packets depending on the size of the packet. AAl5 packets will always use the large queue. Each queue has a start, end, read and write pointer. The mode of operation is similar to the SARA-S.

Packet Complete Queue.

Similar to the SARA-S. All the descriptors from the packets complete queue is read and then the function exited.

Exception Queue.

This transfers the error status not associated with any packets. This too will have four queue pointer.

C.3.2 Packet Memory

Packet Data for Reassembly: The data is pointed to by the address in the buffer descriptors.

CBR Data: All the cells are stored in memory location pointed to by a circular queue with four pointers. i.e. the cells will not reassembled into a packet, rather just stored as cells.

Raw Cell Queue: This store the OAM cells, congestion notification and raw cells. It's operation is identical to the CBR queue.