Congestion Control in Interconnected Computer Networks

A THESIS
Submitted to the Department of Computer
Engineering and Information Sciences
and the Institute of Engineering and Sciences
of Bilkent University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science

By
Özgür ULUSOY

June, 1988

CONGESTION CONTROL IN INTERCONNECTED COMPUTER NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND

INFORMATION SCIENCES

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Özgür Ulusoy

June 1988

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

M. Baray

Prof.Dr.Mehmet Baray(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof.Dr.Erol Arkun

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist.Prof.Dr.Erdal Arıkan

Approved for the Institute of Engineering and Sciences:

M. Baray

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Sciences

ABSTRACT

CONGESTION CONTROL IN INTERCONNECTED COMPUTER NETWORKS

Özgür Ulusoy
M.S. in Computer Engineering and
Information Sciences
Supervisor: Prof.Dr.Mehmet Baray
June 1988

A computer network has a collection of resources shared by multiple users. The capacity of the resources is limited, and if the user demands exceed the capacity, the network becomes 'congested'. The congestion causes a degradation in system performance. In interconnected networks there are two classes of traffic within a network. One class is the local traffic that is generated and transmitted within the network. The other class is the internetwork traffic transmitted to or from other networks. In this thesis, the effect of internetwork traffic on the performance of a network is investigated. Computer simulation of an interconnected network model is provided in order to evaluate the effectiveness of a window-based congestion control mechanism on preventing congestion in gateways and in attached networks caused by the overload of internetwork traffic. Also two dynamic window congestion control algorithms are provided and studied. These algorithms provide further control to window mechanism by adjusting the window size in accordance with the availability of the network resources at the destination. Dynamic algorithms are evaluated comparing them with static window control.

Keywords: Congestion Control, Computer Networks, Interconnected Computer Networks, Window Control, Performance Evaluation.

ÖZET

BİLGİSAYAR AĞLARINDA AŞIRI YÜK KONTROLÜ

Özgür Ulusoy
Bilgisayar Mühendisliği ve Enformatik Bilimleri Yüksek Lisans
Tez Yöneticisi: Prof.Dr.Mehmet Baray
Haziran 1988

Bilgisayar ağları çok sayıda kullanıcı tarafından paylaşılan kaynakları içermektedir. Bu kaynakların kapasitesi sınırlıdır ve kullanıcı istekleri bu kapasiteyi aşarsa, bilgisayar ağlarında aşırı bir yüklenme görülür. Aşırı yüklenme sistem performansının düşmesine sebep olur. Diğer bilgisayar ağlarıyla bağlantısı olan bir ağda iki grup mesaj trafiği görülür. İlk grup ağ içinde üretilen ve transfer edilen mesajlardan oluşur. Diğer grubu ise başka ağlara transfer edilen veya başka ağlardan gelen mesajlar oluşturmaktadır. Bu tezde ikinci grup trafiğin bir ağın performansı üzerindeki etkisi incelenmektedir. Geliştirilen bir model üzerinde birbirine bağlı bilgisayar ağlarında ve geçitlerde görülen aşırı yüklenmenin önlenmesinde pencere metodunun etkinliği bilgisayar benzetimiyle incelenmektedir. Ayrıca pencere büyüklüğünün dinamik olarak değişebildiği bazı algoritmalar önerilmektedir. Bu algoritmalar pencere büyüklüğünün çeşitli ağ kaynaklarının o andaki yüküne bağlı olarak değişebilmesini sağlayarak pencere metodunun aşırı yüklenme problemine daha etkin bir çözüm getirmesini sağlamaktadır. Dinamik algoritmalar, statik pencere kontrol metodu ile karşılaştırılmalı olarak incelenmektedir.

Anahtar Kelimeler: Aşırı Yük Kontrolü, Bilgisayar Ağları, Birbirine Bağlı Bilgisayar Ağları, Pencere Kontrolü, Performans Ölçümü.

ACKNOWLEDGEMENT

I would like to thank my Thesis advisor, Prof. M.Baray for his guidance and support during the development of this study.

I also appreciate Dr. Erdal Arıkan, Dr. Levent Onur, and Dr. Levent Onural for their valuable discussions and comments.

TABLE OF CONTENTS

APPENDIX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 COMPUTER NETWORKS AND INTERNETWORKING GATEWAYS

'Computer network' is a collection of 'computers' or 'hosts' which provide computing services to users. Communication of these computers are provided by means of special purpose communication processors called 'nodes', connected by some communication medium. The nodes and communication lines together form the 'communication subnet' [39].

There are two types of communication subnet:

- store and forward (point-to-point) subnet,
- broadcast subnet.

In the first one communication lines connect a pair of nodes. If two nodes don't share a channel, they communicate indirectly via other nodes. When a

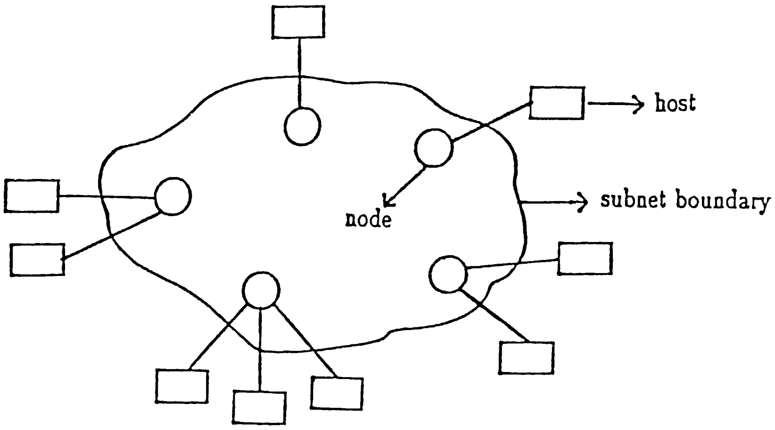
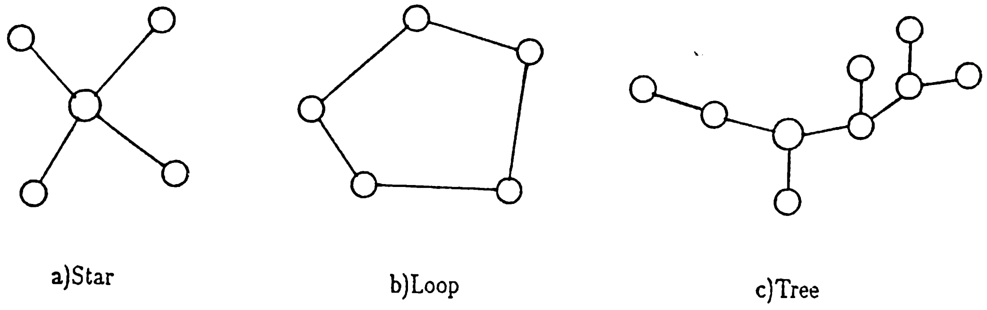


Figure 1.1: Hosts and subnet within a network

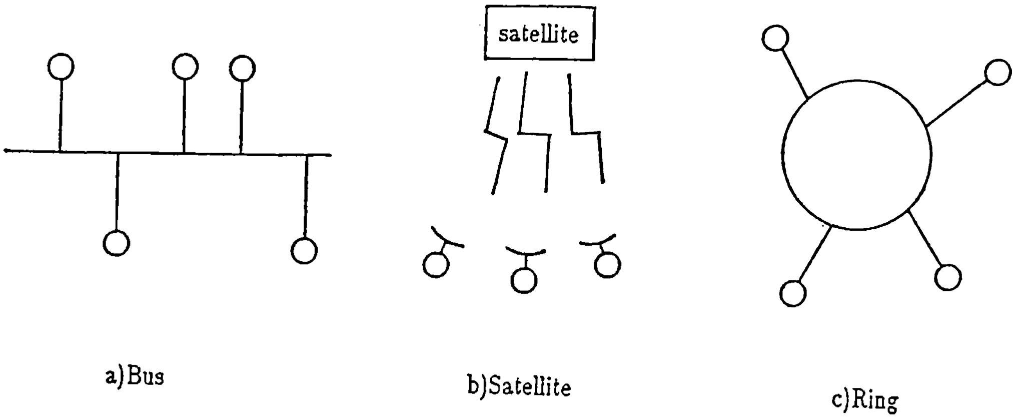


a)Star

b)Loop

c)Tree

Figure 1.2: Some possible point-to-point topological configurations



a)Bus

b)Satellite

c)Ring

Figure 1.3: Broadcast topological configurations

message is sent from one node to another, it is received at each intermediate node, stored until the required line is free, and then forwarded.

For point-to-point channels, node interconnection topology is an important design issue. Examples of different point-to-point network topologies are given in Fig.1.2.

In broadcast subnets, there exists a single communication channel shared by all of the nodes. A message sent by a node is received by all other nodes. The nodes not intended to be a destination, just ignore the message. Some possible topologies for this type of networks are given in Fig.1.3.

The communication among all components of computer networks is regulated by a set of rules called 'protocols'.

In computer networks, there are two general modes of message transfer between nodes. Either 'packet switching (datagram)' or 'virtual circuit ' technique is used during the communication of network components.

In packet switching networks, a message is divided into packets. Each packet has a maximum length and is independently routed through the network. Its routing and transmission through the network is done on the basis of the destination node identification contained in the packet header.

Since each packet is routed independently, the packets injected into the network in a given sequence can arrive at the destination out of sequence. Therefore packet switching networks are non-order preserving.

Another characterization of datagram networks is loss of packets due to discarding them when there exists a shortage of network resources. It is also possible to have duplicate copies of a packet in transit. These duplicates are because of a channel or node failure, and two copies of a packet can arrive at the destination.

In virtual circuit networks, a path is established at the beginning of the communication to be used during the transmission between the sender and receiver. The packets contain virtual circuit number in stead of full receiver address. The packet sequencing is maintained within the virtual circuit.

'Gateways' are used to connect computer networks. Typically, a gateway is a computer system which switches data between networks. An 'interconnected network' ('internetwork' or 'internet' for short) is a collection of networks whose protocols may be different. The function of the gateway is to convert packets from one protocol to another. If the connected networks use different length packets, gateways also perform the function of packet length conversion by fragmentation and reassembly of packets [39].

Gateway applications differ mainly in the types of networks interconnected. The simplest level of internetworking gateway is a 'bridge' connecting similar types of networks. Here no protocol conversion is required. The bridge receives packets from one network, stores them and then retransmit them to the other network.

With the evolving technology of internetworking, communication among different types of networks has become possible. In the connection of various types of networks, gateway application levels change based on the level of protocol layers the networks agree with. This type of gateways perform certain level of protocol conversion [2]. The packets received through one protocol structure, passed across a protocol translator, and transmitted through the other protocol structure.

We can give two examples of internetworking, which are radically different approaches based on the type of service offered. The CCITT internet protocol, called X.75, is based on the virtual circuit model. An internetwork connection is built up by concatenating a series of intranetwork virtual circuits. During the flow of packets along a path, each gateway converts packet formats and virtual circuit numbers as needed.

The alternative internetwork model to CCITT's is the datagram model. This model doesn't require all packets belonging to one connection to traverse the same sequence of gateways. Packets from a source to a destination take a variety of different routes through the internetwork. It is not guaranteed that the packets arrive at the destination in order, assuming that they arrive at all.

1.2 CONGESTION CONTROL

A computer network cannot accept all the data traffic that is offered to it beyond the capacity of its resources. The network may become subject to congestion with increasing traffic if an effective control mechanism does not exist. The congestion gives rise to a degradation in throughput (number of messages handled successfully per unit time) and an increase in average delay of messages. 'Congestion Control' corresponds to a mechanism that controls the traffic to reduce the overload on the network. The objective of congestion control is to prevent or minimize the adverse effects of congestion on the network performance.

Congestion control is one of the most important factors in determining the performance of a computer network. Total throughput of the network, response time of the messages, and the utilization of network resources are highly dependent on the congestion control policy used. All computer networks use some form of congestion control for efficient and fair usage of their resources.

In interconnected networks, if the internetwork traffic constitutes the main portion of the total traffic within a network, then the performance of the network is affected heavily by this class of traffic. Congestion control in interconnected networks is intended to prevent the congestion at gateways along the path of the messages and at the destination network.

1.3 PURPOSE AND SCOPE OF THE THESIS

In this thesis an internetwork model is provided and some congestion control mechanisms are proposed and evaluated in preventing congestion in gateways and in attached networks caused by the overload of internetwork traffic.

In chapter 2, a survey is provided for the most representative congestion control techniques that have been proposed or implemented. It presents the definition, functions, and different methods of congestion control concept in computer networks. The implementation of various congestion control mechanisms in some major operational networks is also discussed in that chapter.

In chapter 3, the internetwork model we have studied on and the internetwork message transfer protocol used are presented. The effectiveness of a window-based congestion control mechanism is evaluated. A brief description of the simulation program that was developed on the basis of the internetwork model is provided. The effect of internetwork traffic and various other internetwork parameters on the performance of a network connected to the internet is evaluated. Performance results are presented for two different medium access protocols: CSMA/CD and token ring, adapted for the transmission of messages within the network.

In chapter 4, two dynamic congestion control algorithms are proposed and studied. The algorithms are applied to the same internetwork model and provide further control to window mechanism by adjusting the window size in accordance with the current system load and availability of resources. An evaluation of control algorithms is provided in terms of the network performance. Dynamic algorithms are evaluated as compared to static window control and it is shown that the algorithms have considerable performance advantages over the static window control.

2. A SURVEY OF CONGESTION CONTROL IN COMPUTER NETWORKS

This chapter provides a survey of various congestion control techniques in computer networks and current implementations of these techniques in some operational networks.

2.1 CONGESTION CONTROL IN COMPUTER NETWORKS

A computer network may be thought of as a collection of resources that are shared by competing users [26]. The resources include communication channels, processing power, and buffers in the nodes. The capacity of the resources is limited, and if the user demands exceed this capacity, the system performance will be degraded. The overloaded network is said to be 'congested'. If the exceeded demands are not controlled, the network throughput will continuously decrease (Fig.2.1).

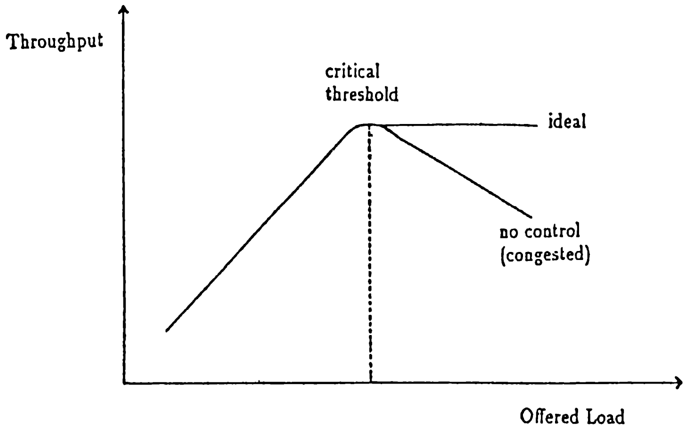


Figure 2.1: Network Congestion

The set of control procedures which are used for eliminating congestion is called 'congestion control' procedures. There are two basic functions of these procedures [27]:

- Detection of congestion, and
- Activation of a suitable control procedure.

Congestion control is necessary for efficient, smooth, and fair transfer of all possible type of information. The quality of the computer network performance is highly dependent on the effectiveness of the congestion control policy used, since it provides the management of the network resources. Fair allocation of the resources between independent users is also intended by these control procedures.

Congestion control procedures include a set of constraints on the network resources and throttles the flow of traffic entering the network. The objective is to prevent throughput degradation and loss of efficiency due to overload. In other words, congestion control is intended to create a loss of traffic sufficient to maintain the level where maximum throughput is achieved.

Congestion control procedures can be implemented both in packet switching and virtual circuit networks [11],[12],[15],[27],[31].

There is some confusion in the literature between the terms 'congestion control' and 'flow control'. Congestion control is primarily concerned with controlling the traffic to reduce the overload on the network [23]. It corresponds to mechanisms that prevent or minimize loss of throughput as the load on the network increases. 'Flow control' is an agreement between a source-destination pair to limit flow of packets without taking into account the load on the network. By this mechanism the sender is controlled by receiver to prevent data from arriving at a faster rate than the receiver can handle it. In other words, the objective of the flow control is to prevent the traffic from entering the network which can not be carried to its destination.

Some authors consider congestion control to be a special case of flow control. But in most cases these two terms are used interchangeably and we will use the term 'congestion control' to mean both.

2.1.1 CONGESTION CONTROL PERFORMANCE MEASURES

Besides the functions mentioned above, there are other objectives of congestion control that are at least as important. These objectives include:

- maximizing throughput,
- minimizing response time of messages.

Throughput is an important measure of congestion control performance. Total throughput (expressed in packets/seconds or bit/seconds) is evaluated as a function of offered load.

Transmission delay of messages may result from the saturation of shared resources. When message delays increase, the current traffic will rise rapidly. In this case some limitation mechanisms will be useful in preventing large delays. A timer can be set for each message travelled (timeout mechanism), or a threshold can limit the number of unsuccessful attempts in sending messages.

Another common measure is the combined delay and throughput performance. In general, it gives us a more complete description of a system performance than the throughput behaviour alone.

2.2 CONGESTION CONTROL TECHNIQUES

Various levels of controlling the congestion in computer networks have been suggested and implemented [11],[31]. We can categorize these levels in three groups:

- End-to-end congestion control,
- Network access congestion control,
- Node-to-node (stepwise) congestion control.

2.2.1 END-TO-END CONGESTION CONTROL

This category of congestion control is applicable to source-destination pair. End-to-end congestion control is the set of mechanisms where destination maintains the sender traffic within the limits compatible with the amount of resources available at the destination. An important function of such control is synchronization of source input rate to sink acceptance rate [27].

End-to-end congestion control methods work by limiting the number of packets permitted from source to destination. Most of them are based on some 'window' mechanism that allows only up to a certain number of packets to be sent by sender before receiving an acknowledgement.

The 'window size' is an important parameter in all of the window control mechanisms and this value may be either static or dynamically changeable depending on the protocol.

Performance analyses of computer networks with window congestion control was studied by Reiser [33], Thomasian [40], and Ilyas [18]. Reiser uses a closed queuing network model to analyze window control applied to virtual circuits of a computer network. An iterative scheme based on Mean Value Analysis (MVA) is presented to solve the network.

In [40] a solution procedure for obtaining individual virtual circuit and network performance measures is presented. The model allows both fixed rate and variable rate traffic sources. Using the model, measures of network performance are computed for several network configurations. These results are used to study the relationship between virtual circuit path length and the optimal virtual circuit window sizes which maximize overall network performance.

Ilyas and Mouftah [18] present the analysis of an end-to-end window congestion controlled network using a hybrid switching technique.

Gerla [13] proposes an algorithm for the optimization of window sizes in a window controlled network for congestion control and fairness requirements.

2.2.2 NETWORK ACCESS CONGESTION CONTROL

Another category of congestion control is network access level congestion control.

The congestion condition is determined at or is reported to the network access points and it is used to regulate the access of external traffic into the network. External traffic is throttled to prevent overall internal buffer congestion. The measurements of internal network congestion may be local due to buffer occupancy in the source node or global due to total buffer occupancy in the entire network.

The most popular implementation of network access congestion control is the 'isarithmic scheme'. It was first proposed by Davies [9]. In this scheme the number of packets in the network is kept below a certain threshold which can be considered as the maximum network load. There exists permits circulating about within the network. When a node wants to send a packet, it first captures a permit. When the destination node removes the packet from the network, it regenerates the permit [39].

An analysis of computer networks with isarithmic congestion control is presented by Wong and Unsoy [47]. They analyze networks with two levels of congestion control. The first level controls the congestion on a global basis by limiting the total number of messages in the network over all message classes (i.e. isarithmic control), while the second level of control operates at the virtual circuit level by limiting the maximum number of messages that each virtual circuit can have (i.e. virtual circuit window size).

The critical parameters of the isarithmic scheme are the total number of permits and the maximum number of permits that can be held at a node. Although this method guarantees that the subnet as a whole will never become congested, there could appear locally congested areas because of unbalanced traffic. Another problem here is that permits can get destroyed for some reason and there is no easy way to find out how many permits exist in the network.

Ilyas [19],[20] proposes several schemes for distribution of permits and effects of these schemes on the performance of computer networks.

'Input buffer limit' control method differentiates between input traffic (i.e. traffic from external sources) and transit traffic. The input traffic is throttled at the source node based on buffer occupancy. It monitors local congestion at the source, rather than global congestion as does the isarithmic method.

The function of this control mechanism is to block input traffic if certain buffer limits are reached in the entry node. This approach favors transit traffic over input traffic. This is a desirable property since a number of network resources have already been invested in transit traffic [15],[27].

An implementation of input buffer limit was suggested by Kamoun [24], in which an input packet is discarded if the total number of packets in the entry node exceeds a given limit. Transit packets can freely claim all the buffers. This scheme is called drop-and-throttle congestion control policy since under heavy traffic conditions it reduces the rate of input traffic (throttling mechanism), and transit traffic that arrives at a congested node is dropped from the network (dropping mechanism).

'Choke packet' control scheme is another network access control mechanism. It is based on the notion of link and path congestion. A link is defined to be congested if its utilization exceeds a certain threshold. A path is congested if any of its links is congested. When a node receives a packet directed to a destination whose path is congested, it drops the packet if it is an input packet. On the other hand, if the packet is a transit packet, it is forwarded but a 'choke' packet is sent back to the source informing it about the congested path and instructing it to block subsequent input packets to this destination. The path to the destination is unblocked if no choke packet is received during a specified time interval [11],[39].

A variation on this congestion algorithm is to use queue lengths instead of link utilization in the determination of choke packet generation.

2.2.3 NODE-TO-NODE CONGESTION CONTROL

Throughput degradation and deadlocks are two unpleasant consequences of congestion in store-and-forward networks. To eliminate these effects, node-to-node congestion control procedures monitor buffer occupancies at each node and reject the traffic arriving at the node when some predetermined limits are exceeded.

The first example to node-to-node control is 'channel queue limit' congestion control. Each node distinguishes the incoming messages based on the output queue they must be placed into. The number of message classes is equal to the number of output queues. There are some predefined limits on the number of buffers for each output queue; packets beyond this limit are discarded.

In the channel queue limit control method minimum and maximum limits on the number of buffers of a node can be set for each class. In different versions of the method, buffers on a node can be completely reserved to different classes or can be shared among all classes. It is possible to change buffer parameters dynamically in time. Basic buffer strategies are given in [11],[22],[48].

Another node-to-node control technique is 'structured buffer pool congestion control' which was proposed by Raubold and Haenle [32]. This scheme distinguishes incoming packets based on the 'hop count' (i.e. the number of network links so far travelled). A number of buffers are allocated to each class. Class i+1 can access all the buffers available to each class i plus one additional buffer. If the buffers at level ≤i are full, then incoming packets with class ≤i are discarded.

An external packet from a host can only be admitted to the subnet if buffer for class 0 at the source node is available.

The channel queue limit and structured buffer pool control mechanisms are applicable to both datagram and virtual circuit networks. In addition, selective control can be applied to virtual circuit networks in node-to-node level. In this case, the congestion control method distinguishes packets according to the virtual circuit they belong to , if a virtual circuit architecture is used. A maximum limit is set on the number of packets for each virtual circuit stream that can be in transit at each intermediate node.

The limit may be fixed at virtual circuit setup time or may be dynamically adjusted based on the current load. The advantage of this scheme is to provide a more efficient recovery from congestion by selectively slowing down the virtual circuits directly feeding into the congested area. Although various buffer sharing policies can be proposed, most of the implementations employ dynamic buffer sharing.

Schwartz and Saad provide a survey of congestion control modeling and analysis techniques in [35]. Some quantitative methods are developed for evaluating the relative performance of various congestion control techniques.

2.3 CONGESTION CONTROL IN INTERNETWORKING GATEWAYS

Since the gateway resources are not unlimited, congestion problem also exists in gateways. When the internetwork traffic transmitted to a network via its adjacent gateway increases without control, the network and its gateway will be subject to congestion. The congestion control in gateways is intended to prevent congestion in gateways and in attached networks caused by internetwork traffic.

It is possible to call this level of congestion control, the gateway-to-gateway level [11]. The level should be designed to prevent the congestion of gateways along the path, and should be supported by explicit gateway-to-gateway protocols for the exchange of status information. The status information should include buffer occupancy at the gateway, and load conditions in adjacent networks.

The actual implementation of the internetwork congestion control will be dependent on the internet protocol used. If the CCITT X.75 protocol is adopted, internet congestion control will be virtual circuit oriented, and will be exercised on a connection-by-connection basis. Alternatively, datagram oriented internetwork congestion control schemes can also be implemented. It is possible to say that, many congestion control methods implemented in computer networks are also applicable at gateway level.

The design of efficient gateway congestion schemes requires a consistency between the gateway level and all other levels implemented in each individual network as well as a consistency across the various networks on the internet path. Internetwork congestion control must be able to balance loads between diverse networks environments.

Some examples of the performance studies of congestion control policies in the interconnected networks can be found in [4],[16],[30],[46].

2.4 EXAMPLES OF CONGESTION CONTROL IN VARIOUS NETWORKS

2.4.1 CONGESTION CONTROL IN THE ARPANET

The ARPANET [28] is the creation of DARPA, the Defense Advanced Research Projects Agency of the U.S. The ARPANET communication subnet uses datagrams inside but provides virtual circuit service to the hosts.

For end-to-end congestion control in ARPANET, all messages transmitted from a source host to a destination host are carried on the same logical 'pipe'. Each pipe is controlled by a window mechanism. Correctly received messages are acknowledged with end-to-end control messages, called RFNM (ready for next message). The source node of the pipe advances its transmission window after receiving an RFNM. RFNM's are also used for retransmission purposes. If an RFNM is not received within a specified timeout, the source sends a control message to the destination inquiring about the possibility of message lost. The destination can request a retransmission after an incomplete transmission.

Messages flowing on a pipe are numbered sequentially. Message numbers are checked at the destination and the messages arriving unordered are discarded to prevent resequence deadlocks.

If multipacket messages are transmitted between a pair of nodes, the destination node must reassemble packets into messages. In the ARPANET possible reassembly deadlocks are prevented by requiring a reassembly buffer reservation for each multipacket message entering the network.

DARPA INTERNET

The DARPA Internet architecture is defined by protocols TCP (transmission control protocol) and IP (internet protocol). The IP defines a datagram based service that allows a host on one network to send datagrams through one or more gateways to a host on another network. Datagram delivery is not guaranteed. The TCP uses acknowledgement, retransmission, duplicate filtering, and other mechanisms to provide virtual circuit services built on top of the datagram machinery. The TCP will compensate for datagrams that are lost [17].

The implementation of virtual circuit service differentiates the DARPA interconnection approach and others, such as the X.25/X.75 scheme. In the X.25/X.75 approach, each network implements a virtual circuit service, and X.75 gateways connect circuits together. In the DARPA Internet each underlying network has to provide reasonable datagram delivery service but need not implement virtual circuits.

CONGESTION CONTROL IN TCP/IP INTERNETWORKS

In heavily loaded datagram networks with end-to-end retransmission, if switching nodes become congested, both end-to-end delay and the total number of datagrams in transit will increase. When retransmission of datagrams begins, there exists a danger of congestion.

Host TCP retransmits packets several times at increasing time intervals until some upper limit on the retransmit interval is reached. Under normal load, this mechanism is enough to prevent serious congestion problems.

Another congestion control mechanism is sending a control message to the source, when a gateway finds itself becoming short of resources. Senders are throttled before they overload switching nodes and gateways. In general, the control message is sent when about half of the buffer space is exhausted. There exists other gateway implementations that generate the control message to the sender only after one or more packets have been discarded [29].

2.4.2 CONGESTION CONTROL IN GMDNET

GMDNET is a virtual circuit network. The route of a message stream is determined at connection setup time and remains fixed until its clearing. Another design characteristics of this network is that the communication protocols are structured recursively. This means that the protocol controlling the message transfer of a virtual circuit between two adjacent nodes is equal to the end-to-end protocol being applied between source and destination node of a virtual circuit [15].

In GMDNET congestion control is provided individually on each virtual circuit. Control is performed between two adjacent nodes and between source and destination nodes according to the recursive communication protocol mentioned above. Both control principles are identical and based on a dynamically controlled packet window.

The main purpose of the source to destination (end-to-end) congestion control is to prevent the overflow at the destination node. Dynamic window size can be reduced if the destination node is slow in accepting packets.

The congestion control between adjacent nodes on the virtual circuit is also exercised independently. While end-to-end window is controlled by destination buffer occupancy, node-to-node window is controlled by intermediate node congestion.

The experiments showed that the individual virtual circuit congestion control is not enough to prevent throughput degradation. So, the buffer space for the input packets at each node was restricted by introducing limits. However to only apply control by input buffer limits was not sufficient to guarantee high network performance. In order to get an effective congestion control, not only input class but also all other message classes should have been applied some limits. GMD network group developed input buffer limit congestion control technique for this purpose.

Fixed path routing in GMDNET provides packets arrive at their destinations in sequence. This property eliminates the need for reassembly buffer allocation which is implemented in ARPANET.

2.4.3 CONGESTION CONTROL IN SNA

SNA [8] is a network architecture developed by IBM to allow its customers to construct their networks and provide distributed communication and distributed processing capabilities between these systems.

SNA is a virtual circuit network, such that at the beginning of each user session an available route is associated among several possible virtual routes. Several sessions may be multiplexed on the same virtual route.

IBM SNA networks use 'virtual route pacing control' mechanism in controlling congestion. This mechanism is an example of an end-to-end window control mechanism. For each virtual route, a fixed window is initially established and a pacing count at the source is set at this value. When a message enters the virtual route, the pacing count is decremented. The first message of the window generates an acknowledgement to the source when it arrives at the destination. The signal arriving at the source node causes the current pacing count to be incremented by the window size [36].

The window size of a virtual route gives the number of messages permitted by the destination to be sent by the source. This can be dynamically adjusted by the destination node and also intermediate nodes along the path on the basis of the buffer availability.

In [36] the performance of the IBM SNA virtual route pacing control mechanism and two variations are analyzed. Simulation is used to ascertain the accuracy of the analysis.

2.4.4 CONGESTION CONTROL IN TYMNET

TYMNET [41] is a commercial network originally developed to interface low speed terminals to time-sharing computers. It is probably the earliest virtual circuit network. TYMNET is a character oriented network.

Each link between two adjacent nodes is divided into channels, and a virtual circuit passing over that link is assigned to a channel. Data from various channels may be combined or multiplexed into one physical packet to share the overhead of checksums and headers among several low speed channels. A high speed channel may use the whole physical packet by itself.

In the congestion control of virtual circuits, for each channel on a link between two nodes, there is quota of bytes which can be transmitted. This quota is assigned at the virtual circuits setup time and varies with the load of the circuit. When a node has exhausted the quota for a given channel, it can not send any more on that channel until the quota is refreshed from the other node. The destination node sends back permission to refresh the quota when it doesn't have much data buffered and the quota is low or exhausted. Doing nothing is the way to throttle the data at the source. This backpressure scheme provides that if a node is overloaded, one effect of the overload is to reduce the load [42].

2.4.5 CONGESTION CONTROL IN DATAPAC NETWORK

DATAPAC [5] is the Canadian public data network, providing virtual circuit service built on a datagram subnetwork. The main communication layers (i.e. virtual circuit and datagram subnet layers) of the DATAPAC network apply different congestion control policies.

DATAGRAM CONGESTION CONTROL

Datagram congestion may be due to a variety of factors including insufficient buffer space, insufficient channel capacity, or a network overload. Regardless of the cause, the congestion will always cause a depletion of communication memory buffers.

When congestion of common memory occurs, the basic mechanism to relieve congestion in DATAPAC networks is to discard subnet datagrams. This event backs up the congestion to the virtual circuit layer of communication (i.e. the virtual circuit layer holds a copy of the packet at the source up to T seconds while waiting for an acknowledgement of transmission across the subnet. If the timer T expires, retransmission of the packet will occur). The virtual circuit layer will control the external rate of packet arrivals according to the rate at which they can be successfully transmitted through the subnet. This feedback mechanism will clear the congestion situation by throttling additional input into the network.

If a packet doesn't get through the subnet after four retransmission attempts by the virtual circuit, then the problem is considered serious enough to clear the virtual circuit call.

The policy for determining the datagrams to be discarded affects the robustness and stability of the system. The policy implemented in DATAPAC gives precedence in access to buffers to datagrams already within the subnet over datagrams just entering the subnet. A threshold is set on available memory buffers, below which no additional datagrams are admitted to the network [38].

VIRTUAL CIRCUIT CONGESTION CONTROL

The basic communication service offered in the DATAPAC network is a virtual circuit service. The virtual circuits make use of the datagram subnet for internode communications. No physical path is actually assigned to virtual circuit as in SNA and GMDNET.

The virtual circuit is implemented as the concatenation of three protocol segments: a protocol from the source device to entry node, a protocol from entry node to exit node, and a protocol from exit node to destination device. Once a virtual circuit is established, congestion control is maintained by a set of three windows for each direction of transmission: a local access window, a subnet access window, and a remote access window. Window sizes are statically assigned for each access line according to the type of service supported (e.g. large windows are assigned if high virtual circuit throughput is required).

The virtual circuit congestion control is very much as described in previous section, that is, subnet congestion backs up to the virtual circuit layer; and the virtual circuit applies control on the external packet arrivals.

GATEWAY CONGESTION CONTROL

DATAPAC internetworking is realized through interface based on CCITT Recommendation X.75 between adjacent networks. Virtual circuits fixed to X.75 gateways will be subject to the same subnet and virtual circuit congestion controls as outlined before. However some additional control is required to prevent a large number of virtual circuits exceeding the capacity of the gateway links. The control is provided by limiting the number of incoming calls and the number of outgoing calls that may be simultaneously established across each gateway link.

3. WINDOW BASED CONGESTION CONTROL IN INTERCONNECTED NETWORKS

This chapter provides a model for the evaluation of congestion control in interconnected networks. Internetwork message transfer is controlled by a window mechanism to prevent message overload at the gateways and connected networks. The window mechanism provides the control by restricting the number of messages in transmission between a source-destination network pair.

3.1 THE MODEL

Fig.3.1 presents the internetwork structure we study on where the individual local networks are connected to the system via their gateways.

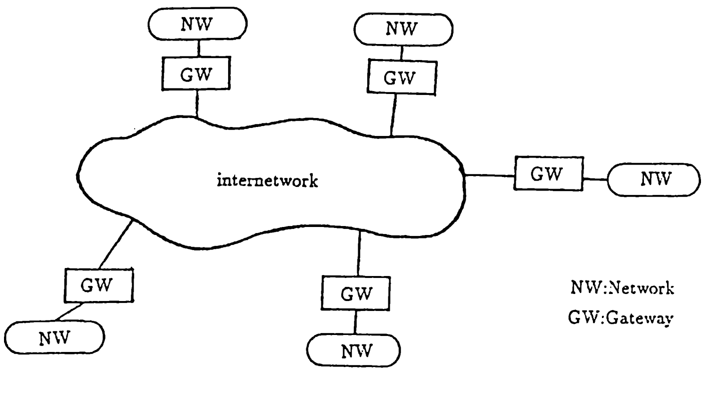


Figure 3.1: Internetwork structure

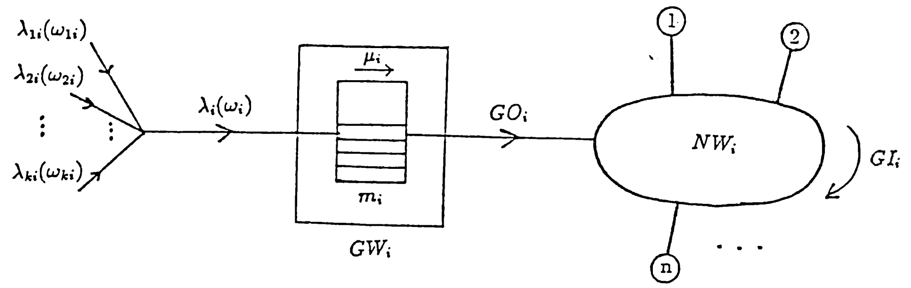

$\lambda_{1i}(\omega_{1i})$
$\lambda_{2i}(\omega_{2i})$
$\lambda_{ki}(\omega_{ki})$
$\lambda_i(\omega_i)$
μ_i
m_i
GW_i
GO_i
NW_i
GI_i
1
2
n

Figure 3.2: Model for a destination network and adjacent gateway

Our study is concerned primarily with investigating the effect of internetwork traffic on the performance of connected networks and providing gateway-to-gateway level congestion control to prevent internet message overload at the gateways and networks. In Fig.3.2 a model is provided for a network and its adjacent gateway within the internet.

Internetwork message transfer is controlled by a window mechanism. Each network is allowed to send up to a fixed number of messages to another network without getting acknowledgement. The fixed number gives the 'window size' between a network pair.

Let GW_i (gateway i) be the representative gateway model and NW_i (network i) be its connected network (Fig.3.2). It is assumed that there are k networks connected in the system. The effect of the internetwork traffic transmitted to NW_i will be investigated. The following parameters can be given for $1 \leq j \leq k$:

$W_{ji}(packets)$: Window size for internet messages from NW_j to NW_i where $j \neq i$. The messages in the system will consist of one packet; so the window size can be given in packets.

$\gamma_{ji}(packets/second)$: Generation rate of internet messages at NW_j to be destined to NW_i where $j \neq i$. The arrival process of internet messages is poisson.

$\tau(seconds)$: Timeout period for the internet messages. If an internet message is not acknowledged within the period τ, it is retransmitted by the source network.

$\omega_{ji}(packets)$: Number of packets (from NW_j to NW_i) currently in the system.

$\omega_i = (\omega_{1i}, \omega_{2i}, ..., \omega_{ki})$: Collection of currently existing packets destined to NW_i.

$\lambda_{ji}(\omega_{ji})(packets/second)$: Effective arrival rate of internet messages from NW_j to NW_i.

$$\lambda_{ji}(\omega_{ji}) = \begin{cases} \gamma_{ji} & if \quad \omega_{ji} \leq W_{ji} \\ \\ 0 & \text{otherwise} \end{cases}$$

$\lambda_i(\omega_i)$: Total effective arrival rate of internet messages to GW_i.

$$\lambda_i(\omega_i) = \sum_{j=1}^{k} \lambda_{ji}(\omega_{ji})$$

$\mu_i(packets/second)$: Message processing rate at GW_i. Service time distribution is exponential.

$m_i(packets)$: Buffer capacity of GW_i. The packets residing on the buffers are serviced in a FCFS (First Come First Served) order.

n_i : number of nodes within NW_i

$GI_i(packets/second)$: Total intranet traffic rate within NW_i. It is the sum of the rate of internal messages generated by all of the nodes. Arrival of internal messages is also assumed to be a poisson process.

$C_i(bits/second)$: Speed (capacity) of the NW_i.

$l(bits/packet)$: Average message(packet) length (both internet and intranet). Message lengths are exponentially distributed with expectation l.

$GO_i(packets/second)$: Rate of internet packets processed at GW_i to its destination (i.e.NW_i). GO_i is a function of parameters $\lambda_i(\omega_i), \mu_i, m_i$. An analytical derivation of this parameter is presented in Appendix A.

The throughput and delay parameters of the messages within NW_i can be given as:

$SO_i(packets/second$ or $bits/second)$: Throughput of internet messages.

$SI_i(packets/second$ or $bits/second)$: Throughput of intranet messages.

$TO_i(seconds)$: End to end delay of internet messages.

$TI_i(seconds)$: End to end delay of intranet messages.

3.2 INTERNET MESSAGE TRANSFER PROTOCOL

The internet message transfer protocol used in this system makes the source network stop sending packets to a destination network if the number of unacknowledged messages sent by the source network to the destination network reaches to the window size limit of the corresponding network pair. Further arrivals of messages have to be blocked at the source. For each packet successfully received, an acknowledgement is generated by the destination. Each packet sent by the source network causes a one unit incrementation of current window, and each acknowledgement from the destination decrements the window by one unit.

The resource limitations for internetwork messages arriving at the destination gateway are gateway buffer size, gateway message service rate, and the link capacity of the network connected to the gateway. After processed by the gateway, an internetwork message will be ready to be transmitted to its destination node within the network. It will then compete with the other internet and intranet messages for the network link capacity.

Some of the internet messages can be discarded at the destination gateway because of unavailable buffer space. A copy of each packet is kept at its source until the acknowledgement of that packet returns from the destination. If the acknowledgement doesn't come within a prespecified period of time (i.e. timeout period), the packet will be retransmitted.

The next section presents the description of CSMA/CD and token ring access protocols which are adapted to the network model under consideration for the transmission of both internet and intranet messages.

3.3 NETWORK ACCESS PROTOCOLS

3.3.1 NONPERSISTENT CSMA/CD

The first protocol we have used for the transmission of messages accessing our network is nonpersistent Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol.

CSMA protocols are used in random access networks where a common channel is shared by the nodes and each node operates independently of the other nodes. There is no central control or channel-controlled access mechanism for random access networks. Each node is free to transmit its packet at a time determined by itself [14],[39].

CSMA protocols are refinements of ALOHA protocols [1], with the refinements through sensing the transmissions of other stations. A node of the CSMA network listens to the channel before transmitting a packet. If the channel is sensed to be busy, the station can defer its transmission until the channel is sensed to be idle.

If two or more users decide to transmit at almost the same time, they can sense the channel as idle and try to transmit. Their packets will overlap on the channel. Such an overlap of packets is called a 'collision'. When a node learns that its transmission is unsuccessful, it reschedules the transmission of the packet to a later time by using some specified backoff algorithm. After the backoff, the node again senses the channel to send its packet.

There are two general classes of CSMA protocols, nonpersistent and persistent, based on the usage of carrier-sense information. In nonpersistent CSMA protocol, when a node becomes ready, the channel is sensed to see if anyone else is transmitting. If the channel is sensed idle, the packet is transmitted; if the channel is sensed busy, the node reschedules the packet to a later time. After that random period of time, the channel is sensed again, and the algorithm is repeated.

In persistent CSMA protocol, when a node has data to send, it first listens to the channel. If no one is sending, it transmits the packet. Otherwise, if the channel is sensed busy, the node keeps on sensing the channel until the channel goes idle, and then it transmits. In other words, the channel is continuously sensed for the purpose of seizing it immediately upon detecting

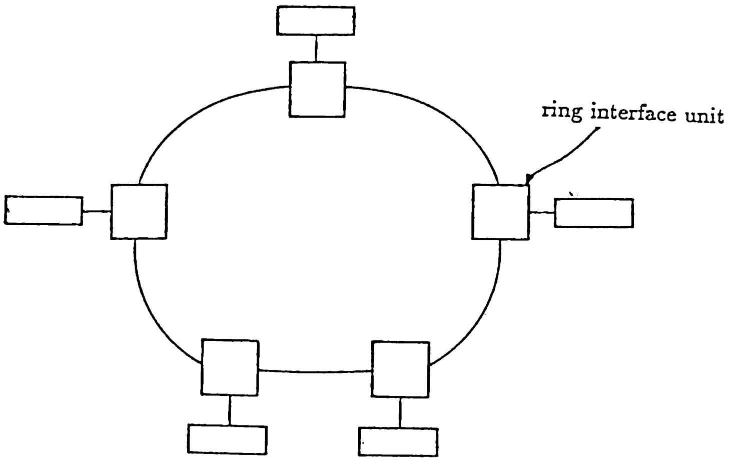


Figure 3.3: A typical ring structure

the end of the previous transmission.

A further improvement on random access networks is possible by the use of carrier sense multiple access with collision detection (CSMA/CD). Collision detection, also called 'listen while transmit', makes it possible to detect a collision shortly after it occurs and terminate transmission immediately, minimizing the channel time occupied by unsuccessful transmissions. To implement the additional feature of collision detection, the transmitter must include hardware not only for transmitting and monitoring, but, in addition, monitoring while transmitting.

In random access networks, the minimum time to detect the collision is the time it takes the signal to propagate from one station to the other. This time interval is called propagation delay and it depends on the physical length of the channel. For carrier sensing to be effective, propagation delays must be less than packet transmission times. The ratio of propagation delay to packet transmission time is an important parameter in performance studies of CSMA.

3.3.2 TOKEN RING PROTOCOL

The second protocol adapted to the local network of our model is a popular kind of ring networks called token ring. The organization of ring networks is different from carrier sense networks. A ring network can be characterized as a sequence of point-to-point links between consecutive nodes (Fig.3.3) [10],[14],[39].

All messages travel over a fixed route from node to node around the loop, passing through network interfaces at each node. Each node is active in the sense that it regenerates the message and can identify addresses, but the interface unit does not usually store messages, as in a store-and-forward network. Each node passes the message on after a short delay of few bit times.

A token ring has the basic structure of all ring networks. Bits from the ring enter the interface in one direction in a serial fashion, are read in the interface, and then after a delay of several bits, are retransmitted over the ring either unchanged or after some modification.

Access to the ring for transmission is controlled by a token, which is a bit structure that can be in one of two possible states: busy or idle. An idle token circulates around the ring whenever all nodes are idle. When a node has data to transmit reads the idle token and changes it to busy state before transmitting it. The busy token then becomes a part of the header of data transmitted on the ring. Thus other nodes on the ring can read the header, note the busy token, and refrain from transmitting. Since there is only one token, there is never any contention as in CSMA networks.

Typically, the token can be a dedicated pattern of several bits. It must be ensured that the bit pattern for a token does not occur in the data. This is accomplished by monitoring the data and using 'bit stuffing', a procedure that breaks-up any data pattern that duplicates the token by adding or 'stuffing' extra bits. When stuffing has been used, the receiver must be able to identify the stuffed bits so that they can be removed before error detection.

There are two operating modes of ring interfaces, listen and transmit. In listen mode, the inputs are simply copied to output. In transmit mode, a node can transmit its data after changing the state of the idle token.

When the data bits that have been propagated around the ring come back, they are removed from the ring by the source node. After finishing the transmission of the last bit of data, the source regenerates the token and switches the interface back into listen mode.

In the listen mode, each node passes on the packet received at its input after a delay referred to as the 'node latency'. The token circulates around the ring in a time equal to the sum of propagation delays between nodes plus the sum of the node latencies. This composite time is called 'ring latency'.

Once a node has captured a free token and thus gained access to use of the

mesno
arrtime
time
mestype
source
next

Table 3.1: Structure of an event list element

ring, there are two types of operation: exhaustive service and nonexhaustive service. For exhaustive service, the node retains use of the ring until it has transmitted all the data stored in its buffer. For nonexhaustive service, the node is allowed to transmit only a specified number of packets.

3.4 SIMULATION PROGRAM

A simulation program has been developed on the basis of the model discussed in the previous sections. The program simulates a network and its adjacent gateway that provides the connection of the network to the internet. End-to-end message transfer in the internet is controlled by the source-destination gateway pairs. Gateways provide the control by restricting the number of internet messages in transit between networks. This restriction is aimed for the fair transmission of internet messages. In other words, internet message overload on the gateways and networks is prevented by assigning proper 'window sizes' between networks.

The simulation is event driven, that is the simulation clock is advanced after simulation of an event to the time of the next event to simulate its action. The event list is stored in a linked list structure ordered by event times. Each event list element is a message generated in the representative network or a message destined to the network from other networks. Each list element consists of time fields, data fields, and a pointer to the next event in the list (Table 3.1).

'Mesno' field of message is used to identify the message in the network. 'Arrtime' field stores the generation(arrival) time of a message. In the 'time' field the time of the current event for the message is stored. Possible events include the message generation, message transmission, retransmission, and gateway service for the internet messages. The message list is kept sorted by this field.

'Mestype' field specifies the type of message, whether it is an internetwork or intranetwork message. 'Source' field stores the source network of an internet message or the source node of a message generated within the network. The pointer 'next' points at the next message in the list. If the CSMA/CD protocol is used, a 'collcount' field is added to each message specifying the number of collisions of the message with the other messages in the network. This information is used during the rescheduling process.

At the beginning of the simulation process, internet and intranet messages are generated and inserted into the message list. Internet messages are generated exponentially with the specified average internet arrival rate. Each source network generates the messages independently from the others and networks can have different generation rates.

Intranet messages are generated within the representative network model. The nodes of the network are the sources of messages and the arrival of intranet messages is exponential with a mean of a specified intranet arrival rate. The gateway can be considered as another node of the network trying to transmit internet messages within the network.

A copy of an internet message is stored in a list at its source when it begins to be transmitted. The time field of the message copy includes the end of timeout period for retransmission. The gateway maintains a list of messages that reside in its buffers.

When the transmission of a message is completed, the message is deleted from the message list. The copy of it is also deleted from the list kept at its source.

The assumptions for the internetwork simulation model are as follows.

- The arrival process of internet and intranet messages follows a poisson process.
- No multipacket message exists in the system, that is all messages consists of only one packet.
- Message lengths are exponentially distributed with the average length of l for both internet and intranet messages. With this assumption, the service time distribution for the internet messages at the gateway and the message transmission time distribution for both types of messages within the network are exponential.

- The gateway has a finite number of buffers and each buffer can store only one packet.
- The transmission error rate is negligible.

The nonpersistent CSMA/CD network protocol adapted to the network model has the following properties:

- The nodes and the gateway compete for channel access.
- The arrival process to a particular node is deactivated until the transmission of a packet already at the node is successfully completed. In the same way, the gateway tries to transmit the first message in its queue.
- After a collision, binary backoff algorithm is used for rescheduling the collided packets. After a successful transmission, the nodes compete for the channel. If there is a collision, all colliding nodes set their local collision count parameter to 2, and they distribute their transmission time over 2 message delay period. After each collision, the node involved in a collision doubles the value of its local parameter and retransmission time interval.

The nonpersistent CSMA/CD protocol applied to our network can be described with the flowchart of Fig.3.4.

The token ring network access protocol is used in the network with the following properties:

- Exhaustive service operation is used in the transmission of packets (i.e. when a node receives permission to send, it empties itself of all queued packets).
- When a node finishes its transmission and regenerates the token, the next node downstream will remove the token if it has data to send. In this manner the permission to send rotates smoothly around the ring in one direction.

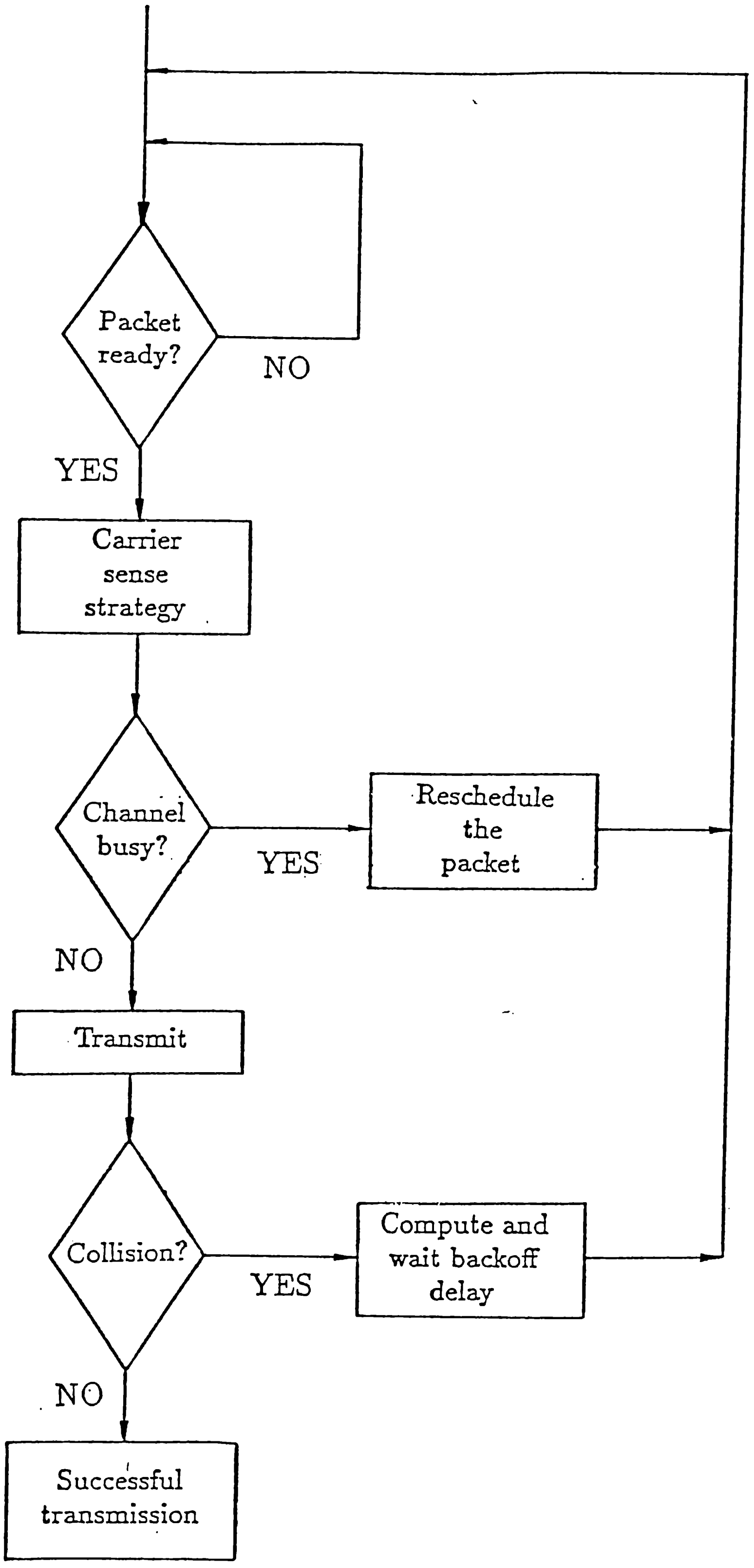


Figure 3.4: Flow diagram for nonpersistent CSMA/CD

$NW = 5$	$n = 10$
$\tau = 0.1 sec$	$GI = 300 pac/sec$
$\mu = 750 pac/sec$	$C = 10^6 bit/sec$
$m = 10$	$l = 1000 bit/pac$
propagation delay=0.00025sec (for CSMA protocol)	
node latency=0.0001sec, propagation delay between nodes=2^{-5} sec (for token ring protocol)	

Table 3.2: System parameters in simulation model

The exhaustive token ring protocol can be described with the flowchart of Fig.3.5.

In the calculation of delay of packets, the ring latency is added to the channel transmission time and the waiting time of packets in the queue.

In the simulation of our model, 'independent replication' method was used to obtain more reliable results. For each configuration simulated, the experiment was repeated many times and the average of the results was used as final estimate. At each run, random number generator was initialized to a different number.

The simulation program was written in Turbo Pascal to be executed on IBM PC. Later a UNIX Pascal version of it was produced for faster execution. The source list in Appendix B simulates CSMA/CD protocol for the transmission of messages in the network. For token ring access protocol the program is similar except the intranetwork message transfer procedure.

3.5 PERFORMANCE MEASURES

In this section, two basic performance measures, throughput and delay, are discussed. The effect of some system parameters (i.e. internetwork message load, internetwork window size, destination gateway buffer size) on the throughput and delay performance of both internet and intranet messages is investigated.

The constant system parameters used in the following measurements are given in Table 3.2.

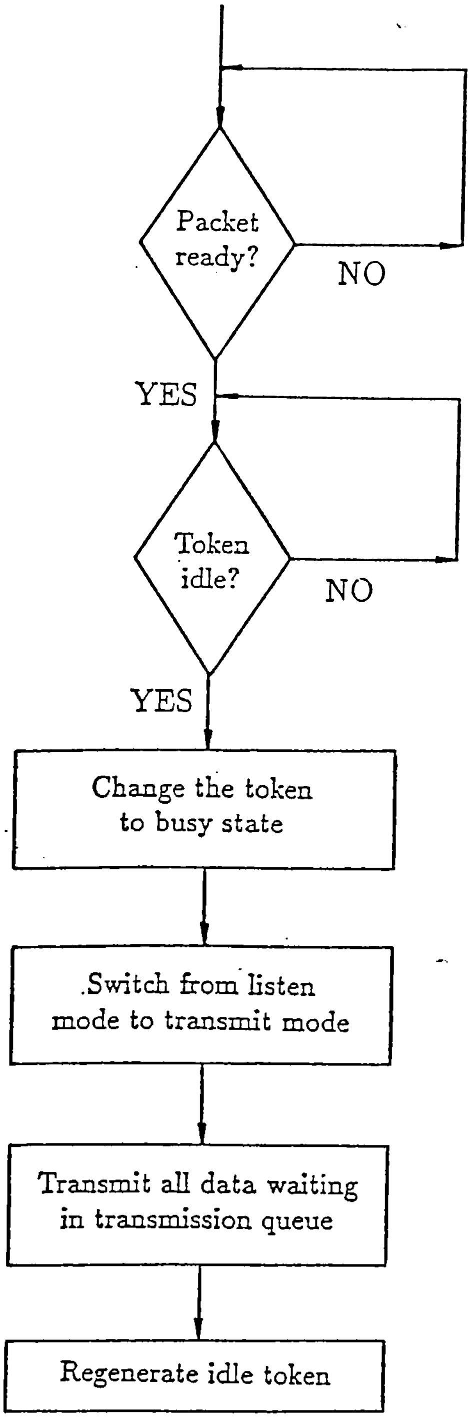

Packet ready?
NO
YES
Token idle?
NO
YES
Change the token to busy state
.Switch from listen mode to transmit mode
Transmit all data waiting in transmission queue
Regenerate idle token

Figure 3.5: Flow diagram for token ring access protocol

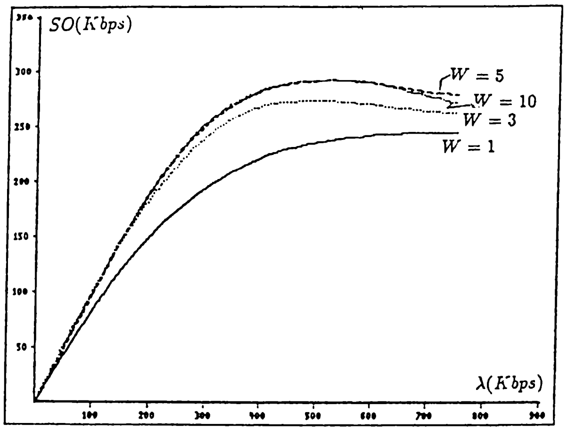


Figure 3.6: Internetwork throughput versus internetwork load in the CSMA/CD network

In the figures, subscripts of the system parameters have been deleted for simplicity.

3.5.1 CSMA/CD PROTOCOL PERFORMANCE

Fig.3.6 shows the throughput of internet messages as a function of internetwork traffic rate for different window size values. W is used to denote the window size between a source and destination network. The nodes of the network are assumed to generate a constant rate of intranetwork traffic. The gateway buffer size is 10.

When the offered internet data rate is increased from zero, internet throughput initially increases linearly. For moderate load values, because of the limiting capacity of gateway and network resources (i.e. gateway buffer size, gateway service rate, network channel capacity) the increase rate of internet throughput will become less and less as the internet load increases. For the high values of offered load, no increase is possible for the throughput of internet messages. It can also be seen the small decrease in throughput values for heavy load with large window size.

For small window size values ($W < 5$), an increase in window size results in an improvement in internetwork throughput. This can be explained by the limiting factor of window size on the number of messages in transmission. Less messages exist in the system for less window size; as a result, the network resources are not fully utilized. Better throughput can be obtained by increasing window size.

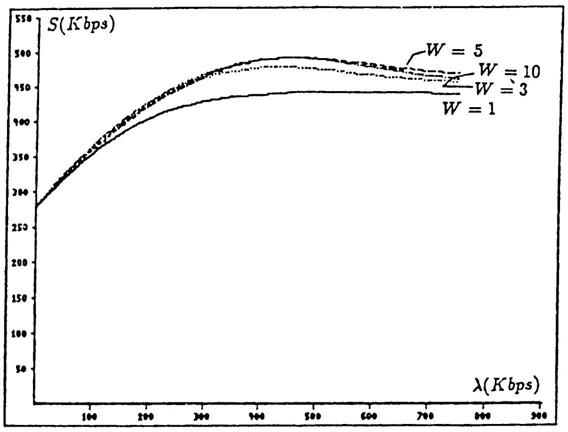


Figure 3.7: Total throughput versus internetwork load in the CSMA/CD network

For larger window size values ($W \geq 5$), destination gateway is busy almost all the time and the resources are almost fully utilized. In this case, increasing the number of messages in transmission will not improve the throughput of messages. In our system, when the internetwork window size is 5, further improvement is not possible by increasing window size. Under heavy load, more congestion is observed at the destination if the window size is large. The congestion results in a decrease in throughput of messages.

Fig.3.7 shows the total throughput of both internet and intranet messages versus internet data rate. Similar to internet throughput, improvement by increasing window size is not possible after window size becomes 5. For small and moderate load values, the rate of increase in total throughput is observed to be less than that in internet throughput. This results from the decrease in intranet throughput by increasing internet load. For large values of offered internet load the total throughput characteristics is similar to internet throughput; no increase is observed in throughput with increasing load.

Fig.3.8 shows the throughput from internet and intranet traffic and the total throughput as a function of offered internet load for the window size of 5. As internet load increases from relatively small values, the throughput of the traffic generated inside the network decreases while the throughput of the traffic from outside increases. However, when the internet load becomes larger, no further increase in internet throughput and total throughput is observed because of congestion at various network resources.

Fig.3.9 shows the average internet delay as a function of the internet load.

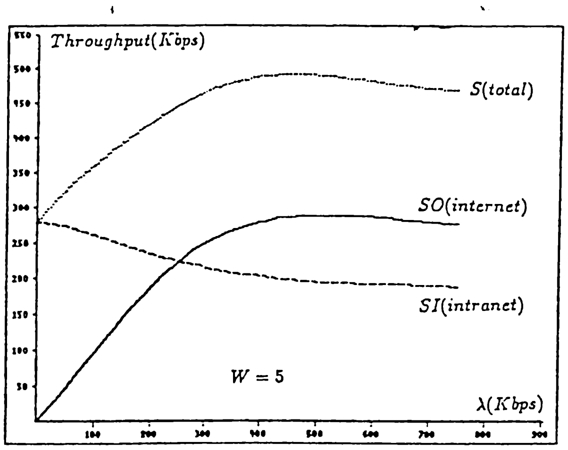


Figure 3.8: Throughput versus internetwork load

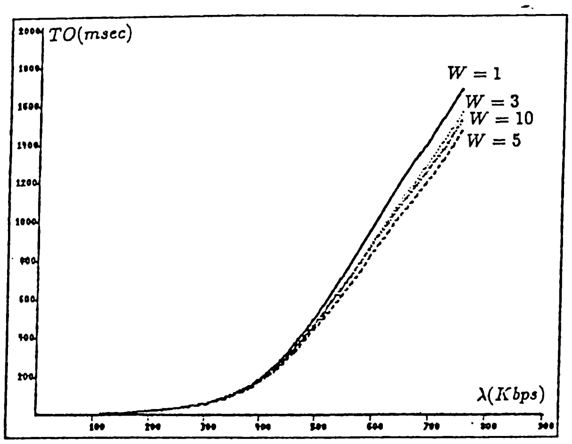


Figure 3.9: Internetwork delay versus internetwork load in the CSMA/CD network

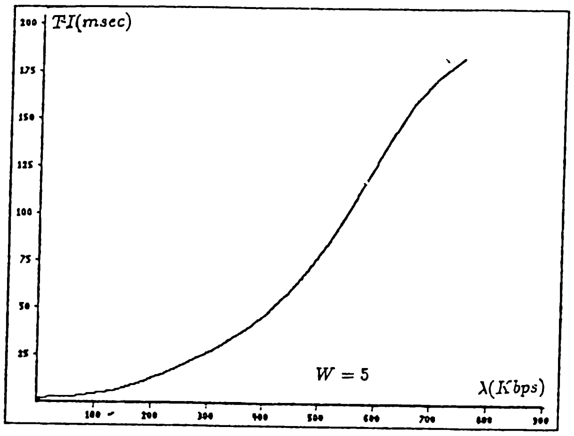


Figure 3.10: Intranetwork delay versus internetwork load

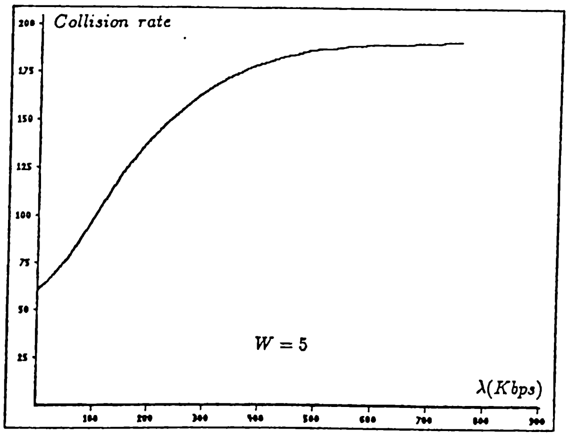


Figure 3.11: Collision rate within the network

It can be seen that as the offered internet message load increases, the delay of those messages also increases because of the contention for the gateway and network resources among the messages. The limitations on the capacity of the resources cause a steep increase in the message delay for high load values.

For small window size values ($W < 5$), the message delay is higher for smaller window sizes. The reason for this is the time periods during which network cannot transmit packets because the window is closed. For large window sizes ($W \geq 5$) it can be seen from figure that the effect of increasing the window size is just the opposite. The larger window sizes cause more delay of messages because of rejections at the gateway. As mentioned before, in our internet protocol, if the gateway buffer becomes full of messages, the newly arrived packets are rejected to be retransmitted later. The retransmission delay has an important effect on the overall internet message delay for large window size values.

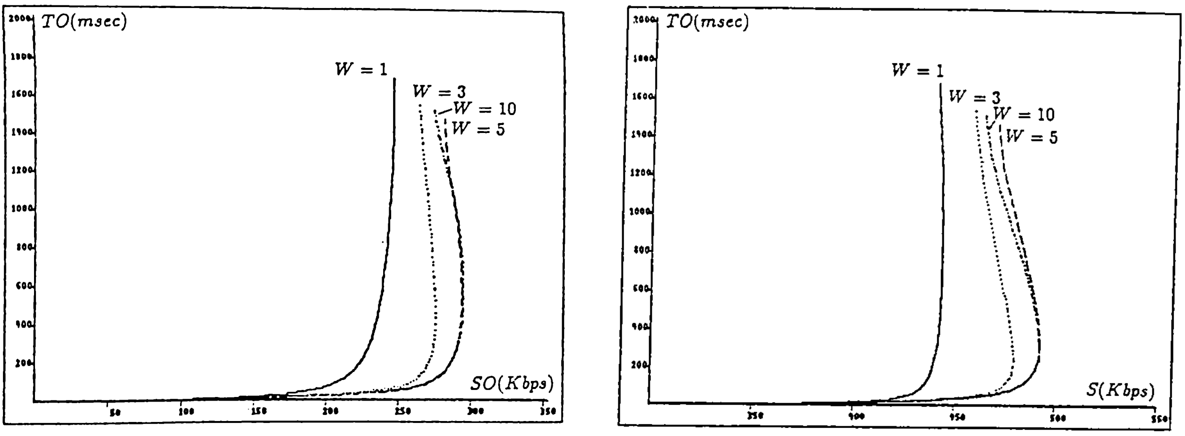


Figure 3.12: Internetwork delay versus internetwork and total throughput

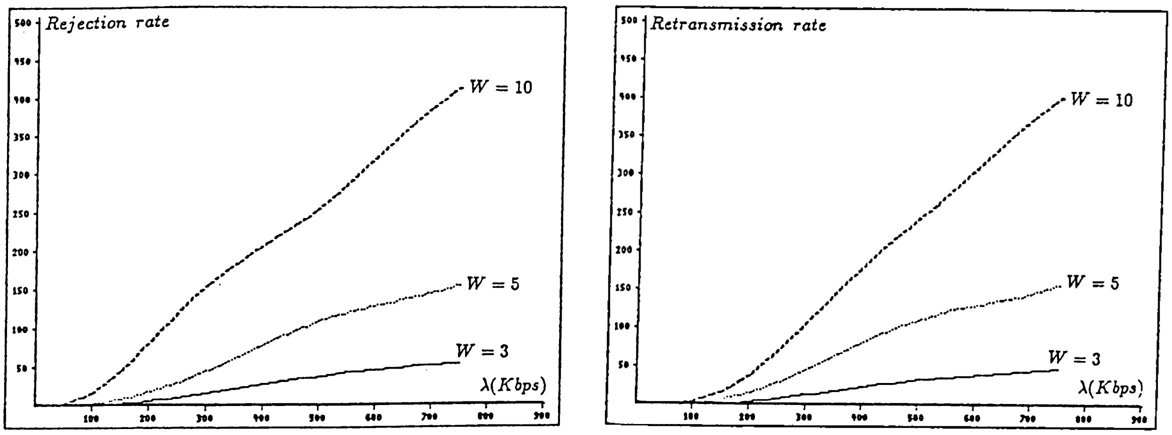


Figure 3.13: Rejection and retransmission rates of internet messages

Increasing internet message load also affects the delay of messages generated within the network. As it can be seen from the Fig.3.10, the intranet message delay increases with increasing internet load because of the contention for the network channel. More collisions in the network are observed with the increasing load (Fig.3.11). The collided packets are rescheduled for the transmission, and thus their delay increase.

Fig.3.12 exhibits the behaviour of the average internet delay with respect to internet throughput and total throughput. Delay increases with increasing throughput because of the contention at various gateway and network resources. When the throughput reaches the maximum value attainable, the increase of delay becomes very steep. After that point, further increase in the offered internet load may lead to both a decrease in throughput and an increase in delay.

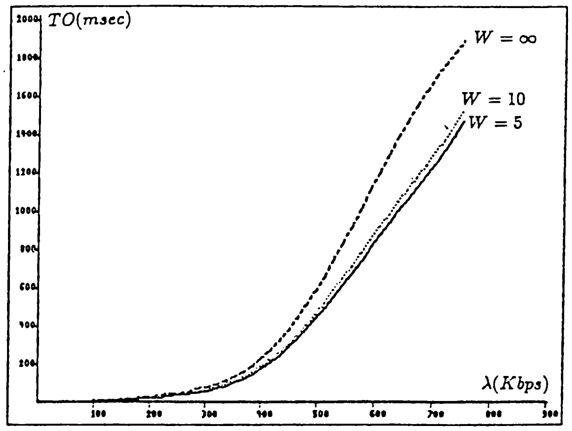


Figure 3.14: Internetwork delay versus internetwork load with unlimited window

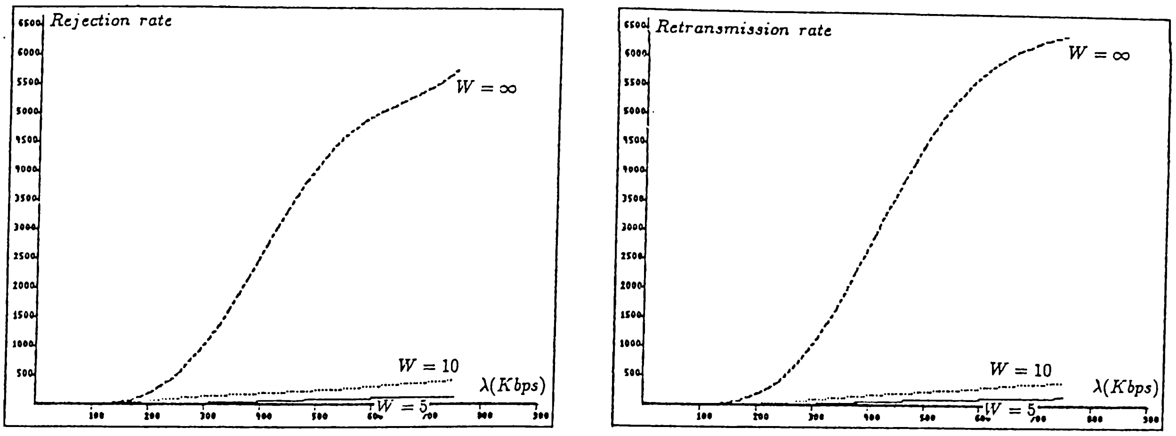


Figure 3.15: Rejection and retransmission rates with unlimited window

Fig.3.13 presents the rejection and retransmission rates of internet messages at gateway as a function of internet message load. The gateway buffersize is 10. Since no rejection occurs for a windowsize less than 3 with these data, the results are given for larger window size values. Noticeable increase in the number of rejections and retransmissions is observed with increasing window sizes.

Fig.3.14 shows the delay behaviour of internet messages without window control (W=∞). It can be seen that the message delay is much higher when the end to end control of internet messages does not exist. This result is due to the increasing rate of rejections and retransmissions of the messages with unlimited window (Fig.3.15).

Our discussion so far has been limited to a fixed value of gateway buffersize m=10. The effect of buffersize is observed to be important for large window size values, since the rate of rejections of messages at the gateway is highly

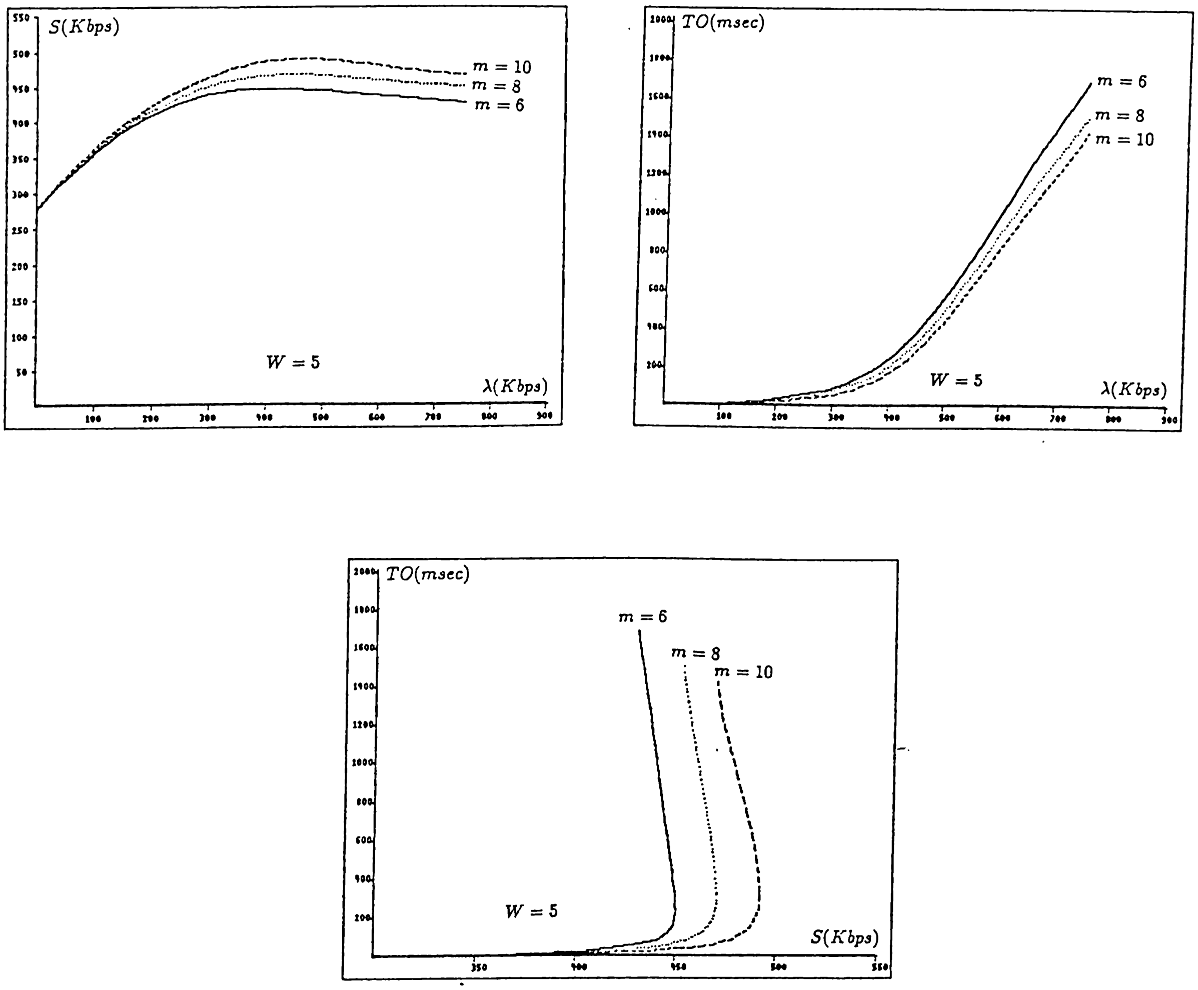


Figure 3.16: Internetwork message characteristics for different buffer sizes

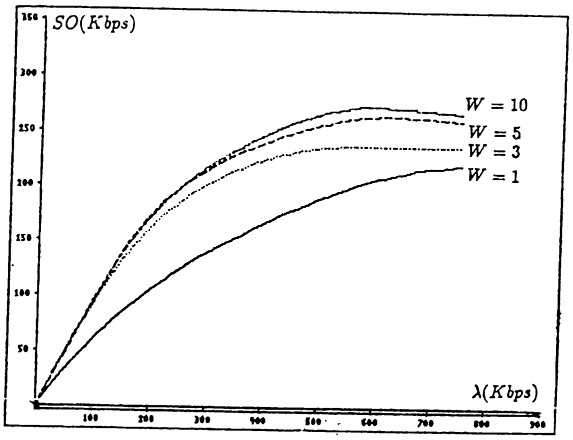


Figure 3.17: Internetwork throughput versus internetwork load in the token ring

dependent on the buffersize. The results for the window size of 5 are shown in Fig.3.16. To investigate the effect of buffersize on network performance, we allowed the value of buffersize to vary between 6 and 20. But since after window size value of 10, no noticeable difference is observed, the results for buffer size values of 6,8 and 10 are presented. The figures show that for a fixed internet traffic rate, as the gateway buffer size increases, the throughput increases and the delay decreases for internet messages. However, we have to note that after a certain buffersize value, further performance improvement by larger buffer size is not possible. The reason is that the input traffic to the gateway approaches output capabilities of the gateway and larger buffer size can not increase the utilization of gateway processors.

3.5.2 TOKEN RING PROTOCOL PERFORMANCE

In the second part of the simulation study, the access protocol of the network has been changed to token ring. By using the same system parameters given in Table 3.2, the following results were obtained.

The throughput characteristics obtained here have some differences from the results of the previous network access protocol. In this case, improvement in internet throughput by increasing window size is possible up to window size of 10 (Fig.3.17). The reason of this fact is the exhaustive characteristic of the network access protocol. The gateway whose buffer becomes full of messages because of a large window size now has the chance to empty its buffer when it gets the right to access the network channel. We observe the same tendencies for the total throughput of the system as a function of

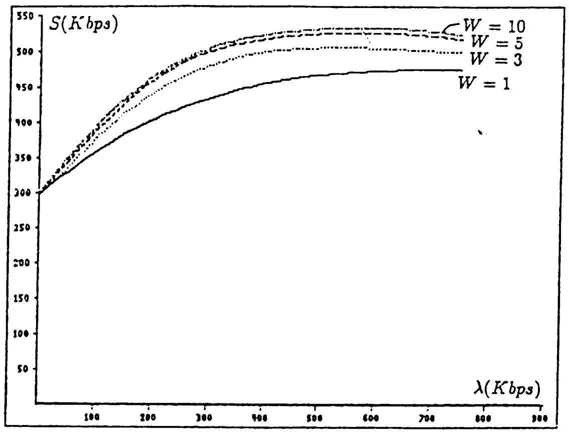


Figure 3.18: Total throughput versus internetwork load in the token ring

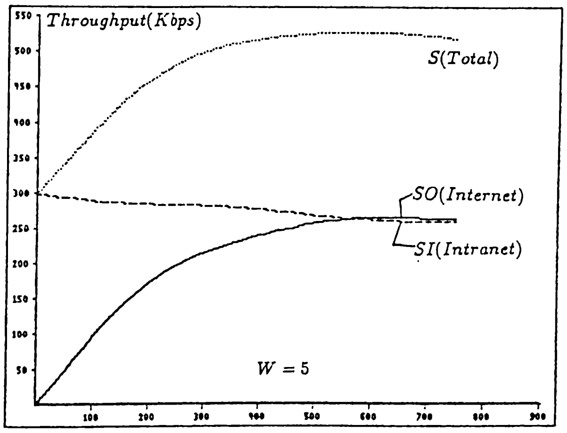


Figure 3.19: Throughput versus internetwork load

internetwork load. Fig.3.18 shows the total throughput characteristics for different window sizes.

If the throughput characteristics of messages are compared with that for the CSMA/CD protocol case, the slight increase in throughput of the token ring method can be seen. This observation can be explained by the lack of contention in token ring network.

Fig.3.19 presents the throughput of internet and intranet messages, and the total throughput as a function of internet load for window size value $W = 5$. As can be seen from the comparison of Figs.3.8 and 3.19, with increasing internet load the rate of decrease in intranet throughput is less in token ring protocol than that in CSMA/CD protocol. It results from the fact that since no contention occurs for the network channel, the effect of internet load on the internal traffic is less for the token ring compared to CSMA/CD protocol.

Figs. 3.20 and 3.21 exhibit the delay performance of the internet messages

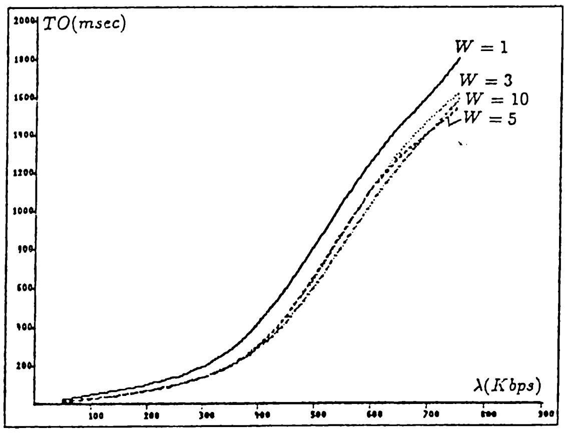


Figure 3.20: Internetwork delay versus internetwork load in the token ring

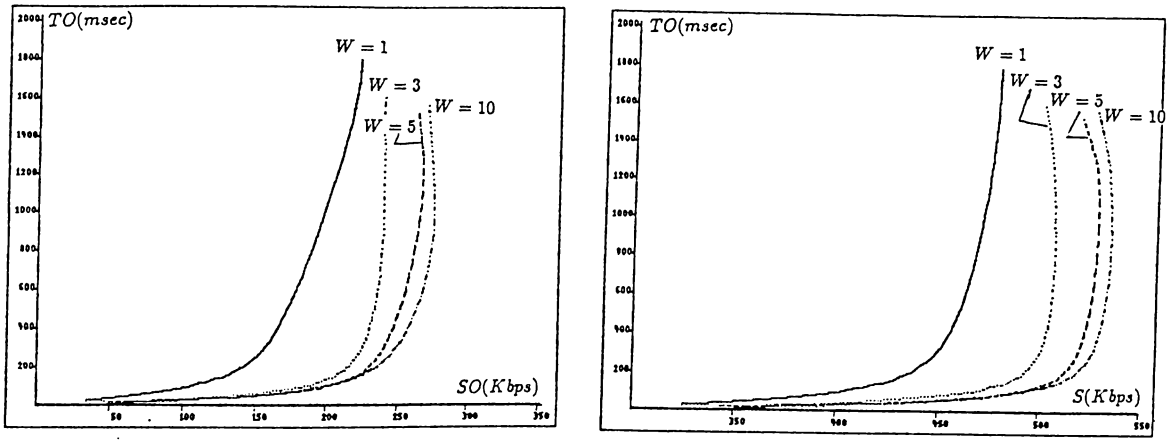


Figure 3.21: Internetwork delay versus internetwork and total throughput

versus offered load and throughput values. As can be seen from the comparison of these figures and the internet delay figures of CSMA/CD protocol, for low load or low throughput values the delay of internet messages messages is slightly higher in token ring network. This result is due to the access delay of the ring until the free token arrives.

It can also be observed from the figures that for small window size values, noticeable decrease in the delay is possible by increasing window sizes. This is because of the exhaustive nature of the network access protocol. All packets queued at the gateway are transmitted together when the token is captured by the gateway. However, for large window sizes ($W \geq 5$) further improvement in the delay under heavy load conditions is not possible by increasing window size due to more rejections at the gateway and the need for retransmission of the rejected messages.

We can see the rejection and retransmission rates of internet messages for various window size values in Fig.3.22.

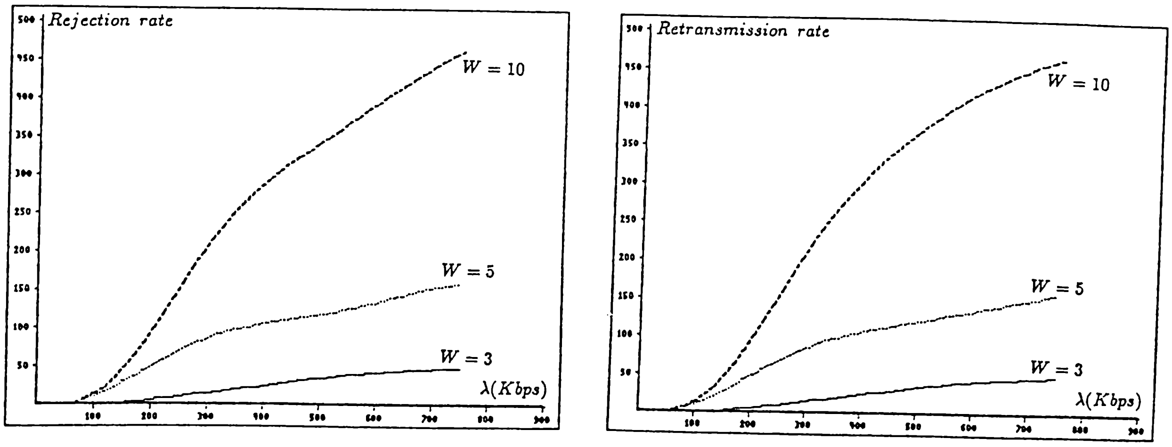


Figure 3.22: Rejection and retransmission rates of internet messages

Another important result of the experiment is that the delay of intranet messages is not affected noticeably by the internetwork parameters such as window size and internet message load. So intranet message delay versus internet load curve is not presented for the token ring protocol.

If no window control exists between the source and destination networks in the interconnected network, the number of rejections at destination gateway and thus the number of retransmissions will increase without bound as can be seen in Fig.3.23. The effect of the large number of rejections and retransmissions on the delay of internet messages is also presented in the figure.

Fig.3.24 shows us how the buffersize of destination gateway affects the network performance. The window size is 5 and other system parameters are same as previous evaluations. For this configuration, better throughput and delay characteristics can be obtained with larger gateway buffer size up to the buffer size value of 12. No further improvement is observed after that value. In CSMA/CD network the improvement was possible up to buffer size m=10. In token ring network, the increase in the buffer size limit for better performance can be explained by exhaustive characteristic of this access protocol.

3.5.3 RESULTS

The key results of the performance measurements of the network with two different medium access protocols can be summarized as follows.

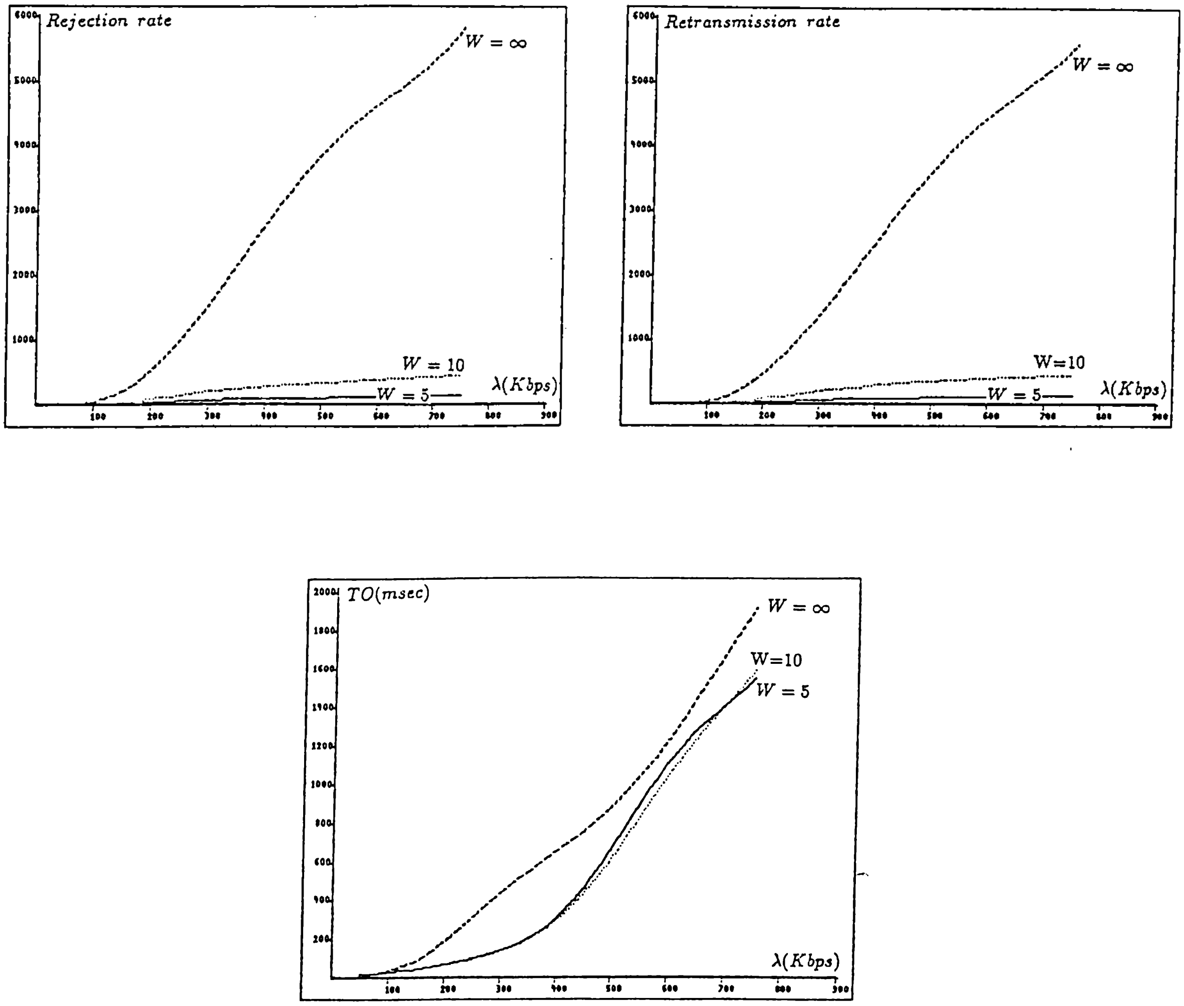


Figure 3.23: Internetwork message characteristics with unlimited window

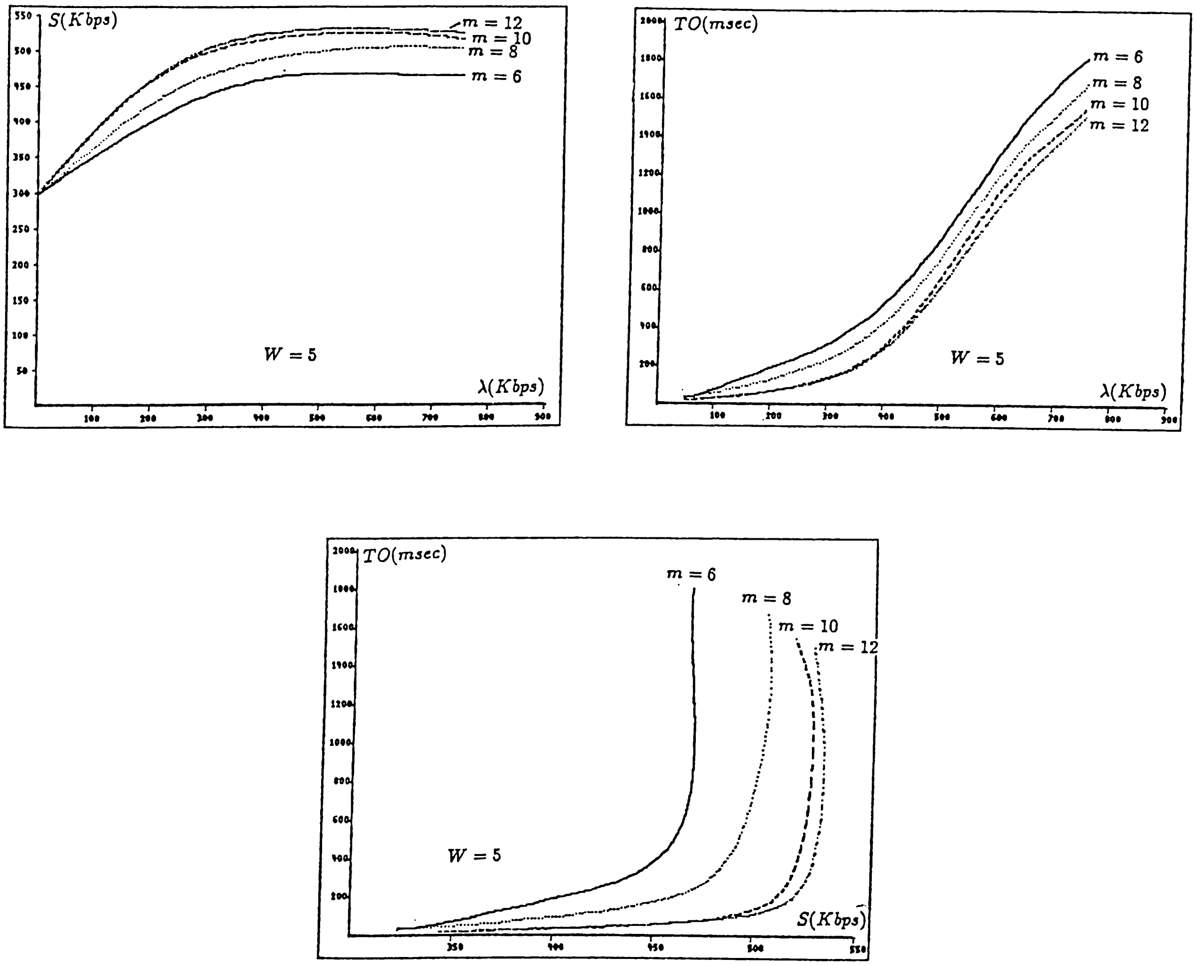


Figure 3.24: Internetwork message characteristics for different buffer sizes

- For low and medium internet load values, as the internet load increases, the internet throughput inside the network increases while the intranet throughput decreases. Total throughput of internet and intranet messages also increases with increasing load. For larger values of internet load, no further increase in the internet throughput and total throughput is possible with increasing load due to limitations of network resources. For the case of internet traffic overload, a slight decrease in throughput values is observed.

- For small internetwork window size values, improvements in both internet and total throughputs are possible by increasing the window size. After a certain window size value, throughput improvement with larger window size is not possible since the full utilization of resources is already attained.

- Increasing internet traffic rate causes an increase in the average delay of internet and intranet messages due to limited capacity of various resources.

- The effect of internetwork traffic on the performance of intranet messages has been observed to be more in CSMA/CD protocol than that of token ring protocol. This effect is due to the contention characteristic of the CSMA/CD method.

- When the internetwork window size is very small or large, the delay of internet messages is larger compared to that for the medium window size values. The reason for the delay in the first case (i.e. small window size) is the time periods during which networks can not transmit packets because the window is closed. For the large window size values, the large message delay is due to rejections at the destination because of limited capacity of network resources. Each rejection will cause a retransmission of message and thus increase in message delay.

- If no congestion control exists between the source and destination network pairs (i.e. no window mechanism), the congestion of the gateway buffer and other network resources increases rapidly and the performance characteristics of messages become worse compared to that with the window control.

- For large window size values, better internetwork throughput and delay characteristics can be obtained by increasing gateway buffer size. However after a certain buffer size value no improvement is possible in the performance.

In the evaluated congestion control mechanism, the internetwork window size parameter is not dynamically adjusted to the internetwork load. The results indicate that it is not possible to set a proper window size between networks that performs well under different internetwork load conditions. For large values of offered internetwork load, a degradation in the throughput and delay performance of the messages is observed due to the increasing number of rejections at the destination gateway. If there exists a further control to reduce the window size in the case of a congestion at the destination, then fewer number of rejections occur resulting in a drop in the average delay of internetwork messages. This is the idea behind the dynamic congestion control algorithms proposed in the next chapter.

4. PROPOSED DYNAMIC CONGESTION CONTROL ALGORITHMS

In this chapter we propose two control algorithms, namely Algorithm1 and Algorithm2, that can adjust internetwork window size dynamically according to the availability of network resources. The main difference between two algorithms is the location of the control of internetwork traffic. In the first algorithm internetwork traffic is regulated at its destination while in the second one source gateway controls the traffic flow.

The proposed control algorithms are intended to operate the system below the critical internetwork load that gives rise to congestion at gateways and connected networks. Dynamic control is provided on the basis of estimated load values determined from the measurements of some control variables. Threshold values are specified for the control variables of the algorithms to place an upper bound on the offered internetwork load.

4.1 PROPOSED ALGORITHMS

In the first algorithm,the internetwork window size between network pairs is adjusted by the destination gateway based on the utilization rates of its resources by internetwork messages. In case of a possible congestion, the destination gateway tries to reduce internetwork traffic. If no congestion is observed the message transmission is operated under normal conditions. The control parameters used in sensing a possible congestion are the rejection rate of messages because of full buffer and the utilization of buffers at the destination gateway.

Let r_i denote the rejection rate of messages and u_i the utilization rate of buffers at the gateway adjacent to network i. The threshold values of the rejection and utilization rates are denoted by R and U respectively. The algorithm enforces the system to operate without exceeding the threshold values in order to prevent the congestion and its degrading effect on system performance.

Initially the interconnected network operates with an initial window size of W_{init} between pairs. We can start at the maximum window size value (W_{max}) allowed by the destination network. The lower limit on window size is one (W_{min}=1).

The following algorithm is executed periodically at destination gateway i.

```
Algorithm1
    If ($r_i \geq R$) or ($u_i \geq U$) then
        If ( $W_{ji} > W_{min}$) then
            Send a control message to source networks
            to decrease the window size by 1
            (For each source network $j$ $W_{ji} = W_{ji} - 1$ )
    Otherwise
        If ($W_{ji} < W_{max}$) then
            Send a control message to source networks to
            increment the window size
            (For each source network $j$ $W_{ji} = W_{ji} + 1$ )
```

W_{ji} denotes the current value of window size between the source network j and destination network i. It can change dynamically between the values W_{min} and W_{max}.

Algorithm2 is a different version of Algorithm1 in the sense that in this case the control is provided by the gateway next to the source network and applied to each of its destination network independently. When a source gateway senses the congestion at a destination, it limits the number of packets destined to that network by dynamically adjusting its window size. The control parameters used in the algorithm are the retransmission rate and response time of messages transmitted by the source network.

For the source network j and the destination network i assume that n_{ji} and t_{ji} denote the retransmission rate of messages and average response time of messages respectively. The threshold values of the control parameters n_{ji} and t_{ji} are denoted by N and T.

The following algorithm is executed periodically by the gateway adjacent to network j.

```
Algorithm2
    For each destination network i do

        If $(n_{ji} \geq N)$ or $(t_{ji} \geq T)$ then

            If $(W_{ji} > W_{min})$ then

                Decrease the window size by 1
                ( $W_{ji} = W_{ji} - 1$ )

        Otherwise

            If $(W_{ji} < W_{max})$ then

                Increase the window size by 1
                ( $W_{ji} = W_{ji} + 1$ )
```

Again we can start at maximum window size value ($W_{init} = W_{max}$) and the window size W_{ji} changes dynamically between the values W_{min} and W_{max} as in the first algorithm.

The difference between the control algorithms can be summarized as follows:

- In Algorithm2 internet traffic is regulated at its sending sources while in Algorithm1 control is provided by the destination gateways.

- The control parameters in Algorithm1 are the rejection rate of messages

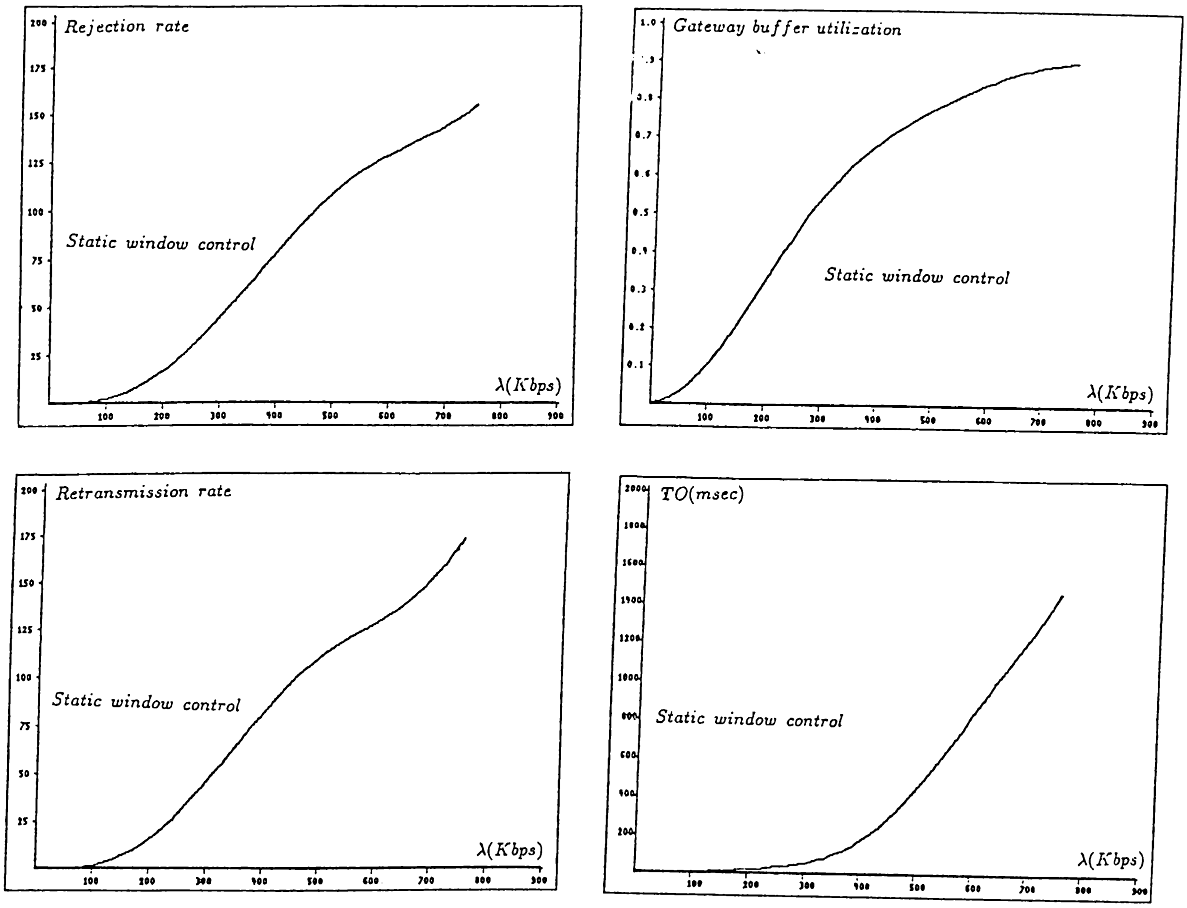


Figure 4.1: Various network characteristics for static window control

and the utilization rate of buffers at the gateway connected to the destination network. In Algorithm2 retransmission rate and response time of messages are used as control parameters by a source network; control for each destination network is provided independently based on the characteristics of the message class belong to that destination.

4.2 PERFORMANCE MEASURES

For the performance evaluation of the algorithms, the simulation program developed for static window congestion control has been enhanced to include the dynamic control algorithms. Three versions of the program, one for the static control and other two for the dynamic algorithms, have been executed to obtain comparative results.

In this section throughput and delay performance of the messages are presented comparatively for static and dynamic window control. The constant

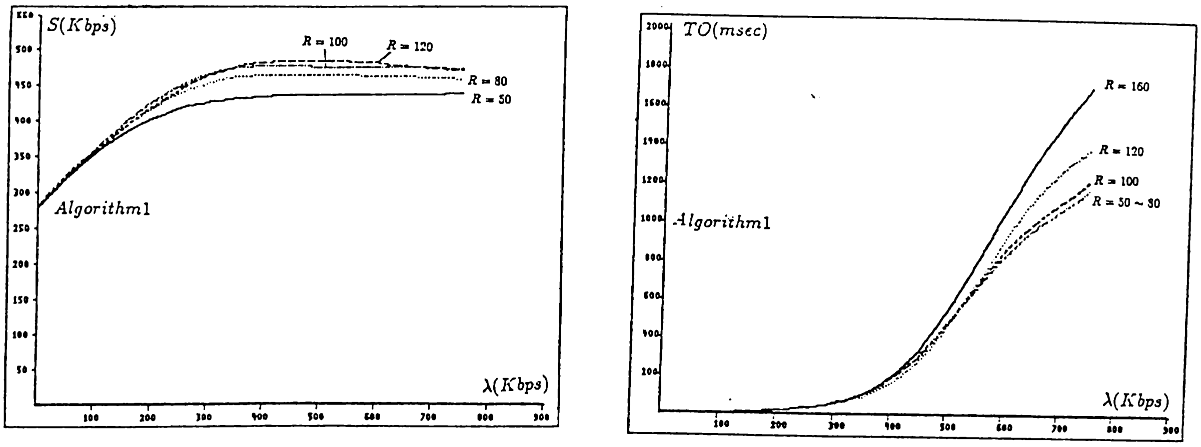


Figure 4.2: Total throughput and internet delay characteristics with Algorithm1 for the different threshold values of rejection rate

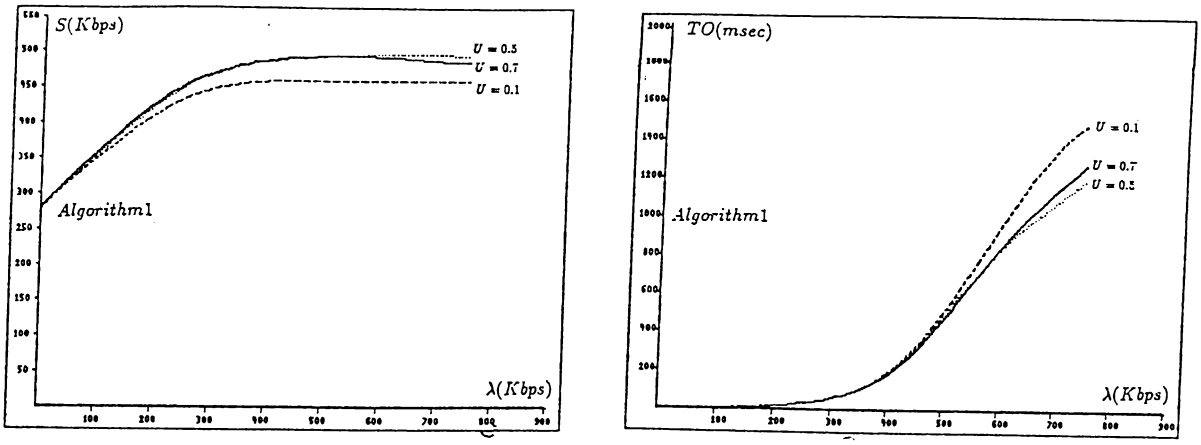


Figure 4.3: Total throughput and internet delay characteristics with Algorithm1 for the different threshold values of buffer utilization rate

system parameters used in the measurements are same as the ones used in previous evaluations (Table 3.2).

Fig.4.1 shows the behaviour of messages under static window control with respect to the internetwork message load from outside to the network under consideration. The static internetwork window size is 5. The figures present the behaviour of various system measures that are used as control parameters by the dynamic algorithms.

Before the implementation of the algorithms, we have to choose the appropriate threshold values for the control parameters. This choice will affect the responsiveness of algorithms to changing conditions. The threshold values are interpreted by the control algorithms as an indication of congestion at the destination network. Figs.4.2-4.5 provide an evaluation of the dynamic

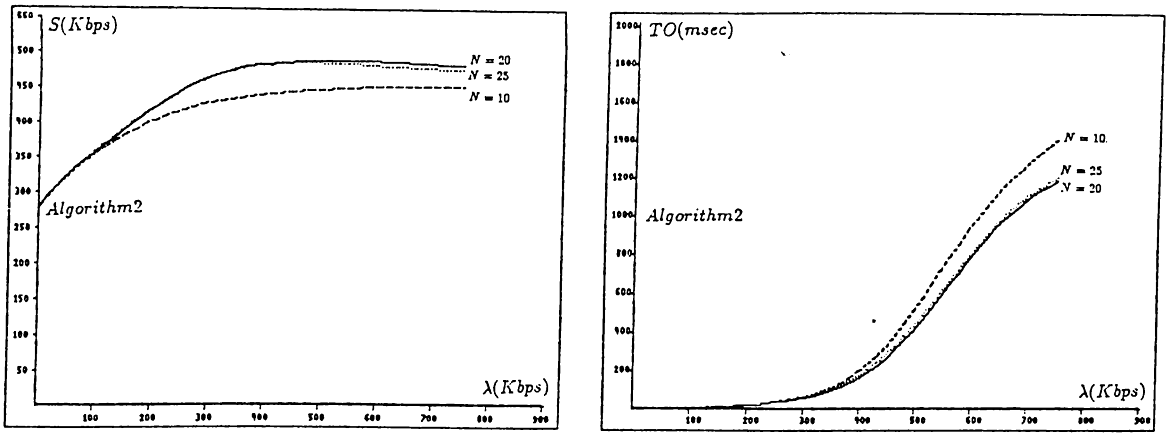


Figure 4.4: Total throughput and internet delay characteristics with Algorithm2 for the different threshold values of retransmission rate

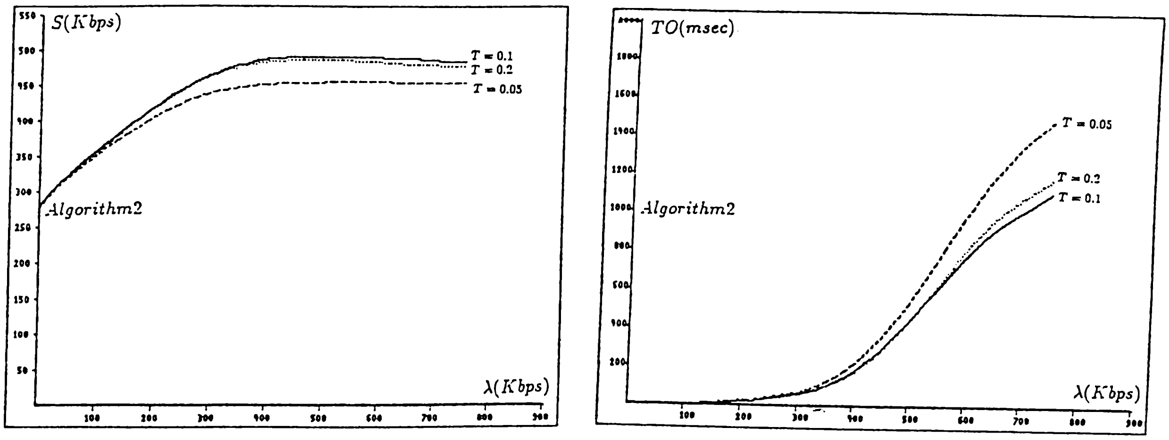


Figure 4.5: Total throughput and internet delay characteristics with Algorithm2 for the different threshold values of message response time

algorithms in terms of the total network throughput and end-to-end internet delay as a function of the total internetwork load for different threshold values of control parameters. The maximum window size (W_{max}) was chosen equal to 5. Appropriate threshold values can be obtained for each control variable to get the highest possible efficiency from the algorithms. If we return back to Fig.4.1, we can see that the proper threshold values of the control parameters correspond to the critical value of internetwork load where the congestion of resources begins. Higher values of internetwork load causes a rapid increase in the rejection and delay of messages without improvement in the throughput. Operating the system without exceeding threshold values, it is aimed to maintain the level where maximum throughput is achieved.

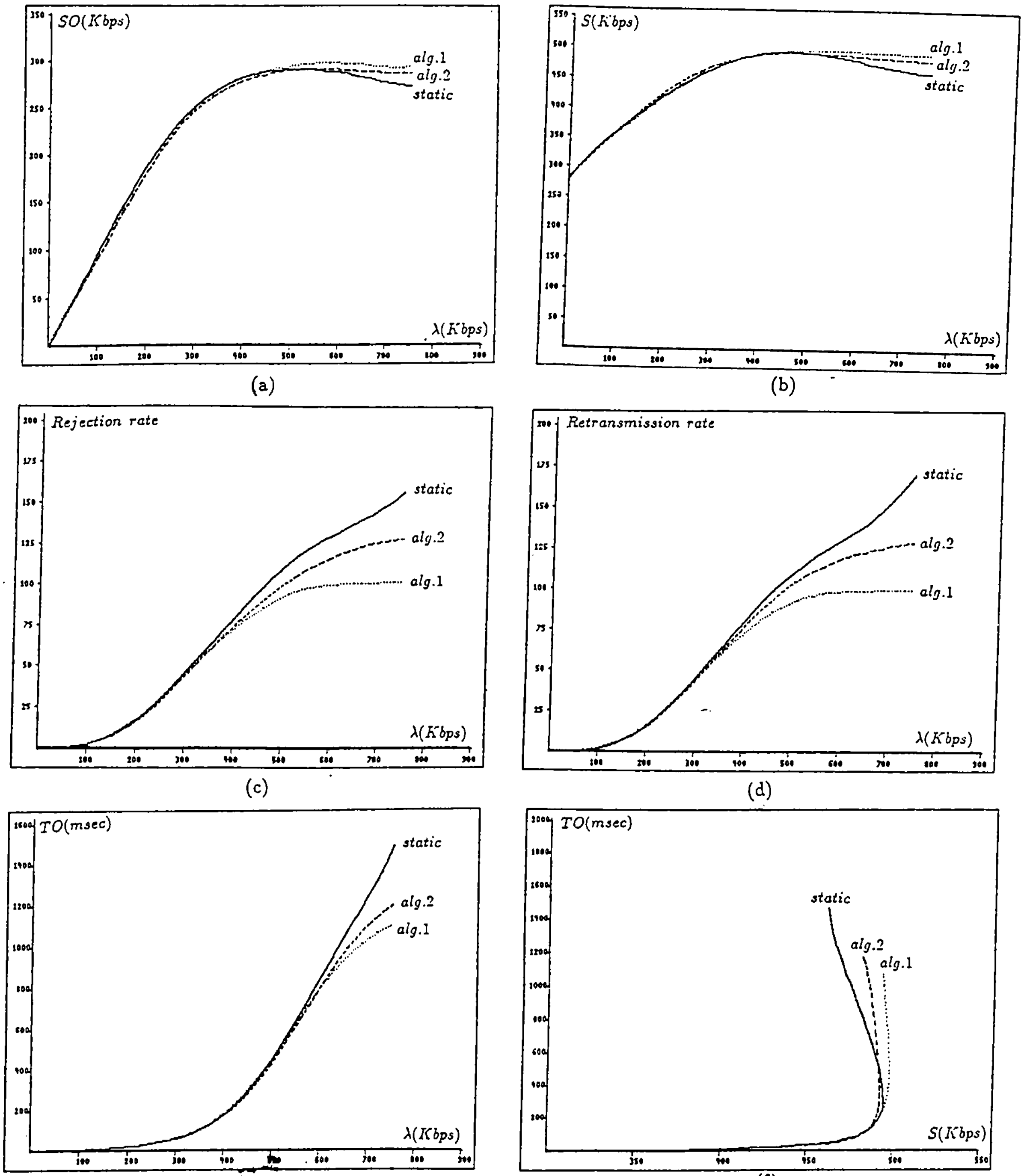
We have to note that the choice of threshold values depend on the internetwork configuration. For our configuration, the critical load value for internetwork messages changes between 400-500 Kbps. We can use the corresponding values of control parameters , i.e. buffer utilization, rejection rate, retransmission rate, and response time of messages; as the threshold values for the control algorithms, Algorithm1 and Algorithm2.

Fig.4.6 demonstrates performance aspects of the dynamic control algorithms for the messages directed to a local network where CSMA/CD medium access protocol is employed. Fig.4.6a presents the throughput of internet messages as a function of internet traffic load for the static and dynamic window control. At light loads there is no difference in the throughput values since no congestion is observed and the dynamic algorithms do not intend to reduce the load. At the high values of offered load, increasing the load under the static window control causes a drop in throughput because of the congestion of resources. The throttling effect of the dynamic control algorithms prevents the congestion and thus the decrease in the throughput value while the internet load is increasing. The same behaviour can be observed for the total throughput of both internet and intranet messages within the network. Fig.4.6b shows the total throughput characteristics for the static and dynamic window control comparatively. The conclusion that can be drawn from these two figures is that although there does not exist much difference in the throughput values, the dynamic algorithms are effective in preventing the decrease in throughput for large internetwork loads.

Figs. 4.6c and 4.6d give the rejection and retransmission rates of internet messages comparatively for the static and dynamic window control. In both figures the rapid increase in the number of rejections and retransmissions can easily be seen for the static control with the increase in offered load. The dynamic algorithms try to reduce this increase by adjusting the window size value to the current system load. The effectiveness of the control algorithms in preventing congestion can also be seen in delay characteristics of the messages. As it can be observed from the Fig.4.6e, the steep increase in the delay of internet messages because of the large number of rejections and retransmissions can be prevented by applying the control algorithms.

Fig.4.6f exhibits the behaviour of average internetwork message delay with respect to total network throughput. When the throughput reaches its maximum value, further increase in the offered internetwork load leads to both a decrease in throughput and an increase in delay. Dynamic algorithms are effective in preventing the sudden rise of delay.

Under the same assumptions, Fig.4.7 shows the performance measures



(a) (b)

(c) (d)

(e) (f)

Figure 4.6: Different performance characteristics of a network for static and dynamic window control with CSMA/CD medium access method within the network

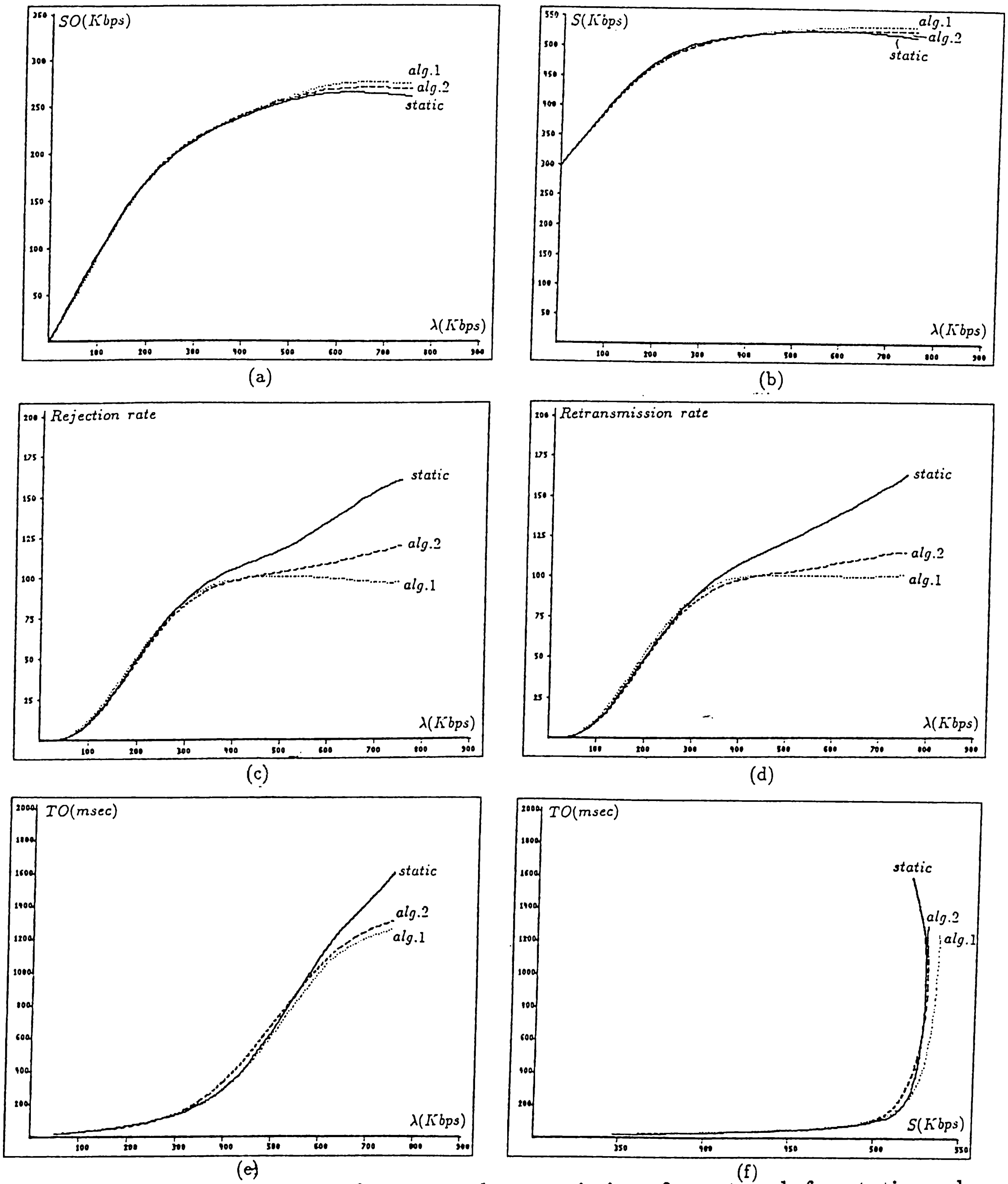


Figure 4.7: Different performance characteristics of a network for static and dynamic window control with token ring medium access method within the network

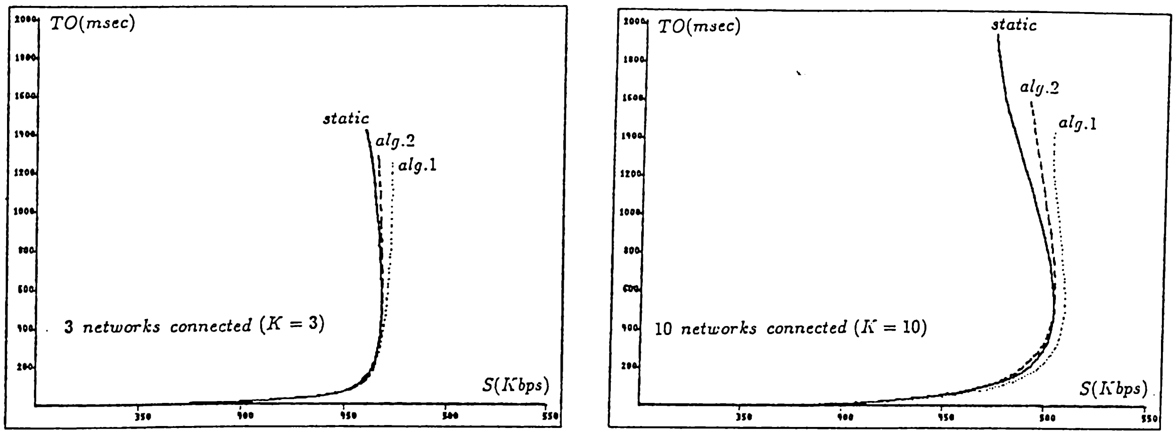


Figure 4.8: Average internet delay versus total throughput for different number of networks connected to internetwork

when the token ring access protocol is applied within the destination network. The performance results obtained for the CSMA/CD protocol are also valid for the token ring protocol. When the system is overloaded, the throttling effect of the algorithms prevents the drop in throughput values (Figs.4.7a,4.7b), and causes a decrease in the number of rejections and retransmission of internetwork messages (Figs.4.7c,4.7d). The improvement in the message delay by the application of dynamic control can be observed from the Figs.4.7e,4.7f.

The next point to evaluate is the fairness of the dynamic congestion control algorithms under different networking conditions. For this purpose, the control algorithms were simulated with many different possible configurations. In the evaluation of different configurations, various system parameters, (i.e. number of networks connected, destination gateway buffersize, gateway message service rate), were employed with changing values. In the following figures, the results of different configurations are presented by average internetwork delay versus total throughput curves.

Fig.4.8 presents the effect of the dynamic control algorithms on the throughput and delay performance of messages for different number of networks connected to internet.

The performance results for different number of connected networks are obtained under the same conditions as the previous examples. The local network executes CSMA/CD method for the access of messages. The threshold values used for the control parameters of the algorithms are obtained from the corresponding values of critical internetwork load under the static window control.

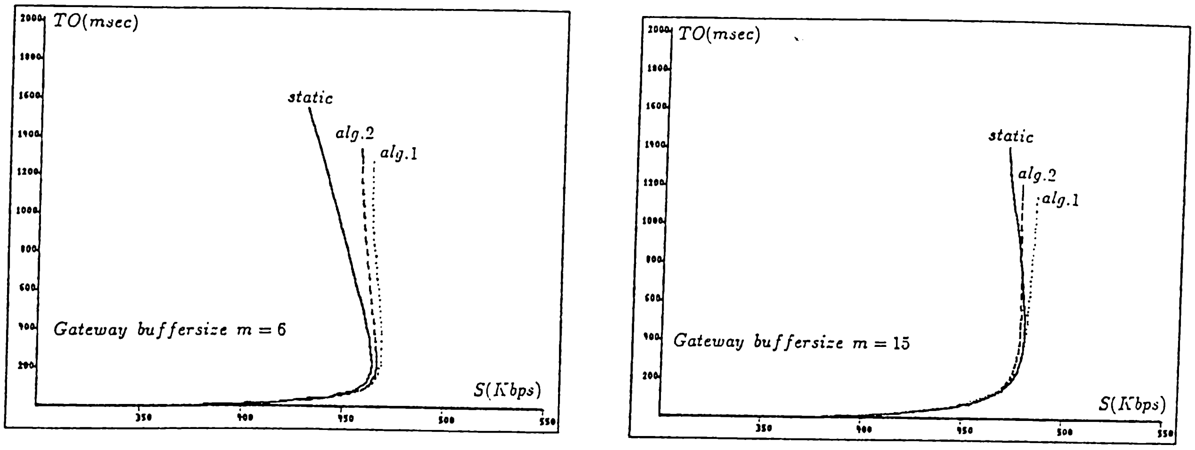


Figure 4.9: Average internet delay versus total throughput for different buffersizes of destination gateway

We have already evaluated the effectiveness of the dynamic control algorithms for 5 networks connected to the internetwork model. In this case, the performance results are obtained for relatively small and large number of networks. Equally good performance results are yielded by the dynamic algorithms for the new parameters. Fig.4.8 exhibits that for the moderate and high load values the improvement in the throughput and delay of the messages is possible for both small and large number of connected networks. Just like the previous example, the drop in the throughput values can be prevented together with less message delay by the application of dynamic control algorithms.

In Fig.4.9 we compare the throughput and delay characteristics of the dynamic and static window congestion control mechanisms for two different gateway buffersizes (m=6 and m=15). As it can be observed from the figures, better throughput and delay characteristics are obtained for larger buffersize since for high values of the applied load, the small buffersize becomes the system bottleneck. For the large values of offered load a stable throughput behaviour is yielded by the dynamic algorithms for both small and large buffersize values. We can also note the internet delay efficiency of the algorithms just like in the previous example with m=10.

Fig.4.10 shows the performance results for different service rates of messages at the destination gateway. We have already seen the effects of the dynamic algorithms on system performance for the gateway service rate of $\mu = 750$ mes/sec. In this case the results are obtained for two values of service rate, namely $\mu = 250$ mes/sec. and $\mu = 1500$ mes/sec., corresponding to relatively slow and fast processors at the gateway. From the figures we

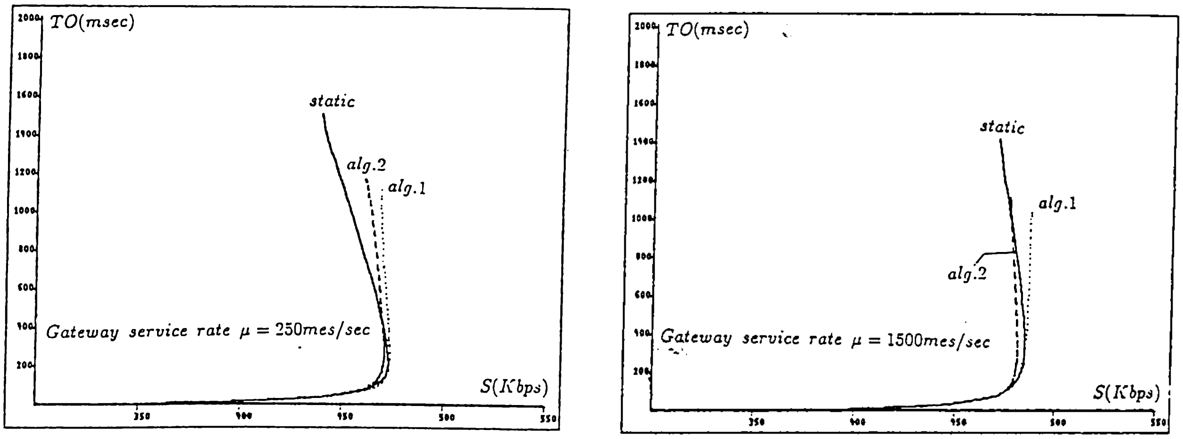


Figure 4.10: Average internet delay versus total throughput for different gateway service rates

can note the negative effect of the slow gateway processors on system performance. It results from the fact that, much more rejection and retransmission of the internet messages are observed with the slow message processing rate. The dynamic algorithms help to improve the performance for that case. The algorithms also performs well for the case of fast gateway processing rate. It can be observed from this and previous examples that, when the system resources are much limited, the dynamic algorithms begin to affect system performance at lower values of offered load.

4.2.1 RESULTS

From the above discussion, the following points can be concluded for the efficiency of the dynamic control algorithms:

- The performance of a network comprising an internet is affected heavily by the internetwork messages. For the large values of offered internetwork load a degradation in the throughput and delay of the messages is observed due to the increasing number of rejections at the destination. The dynamic algorithms provide a solution to the overload case by adjusting the internetwork traffic rate to the current system load.
- The dynamic algorithms are effective for moderate and large internetwork load values. For the light values of offered load, since no congestion is observed the message transmission operates under normal conditions. It is guaranteed by the algorithms that when the system

is lightly loaded, networks operate at maximum window size allowed which is a desired property to make full use of the available resources.

- The dynamic control algorithms provide a stable throughput behaviour, in the sense that for the large values of internetwork load throughput does not decrease with increasing load value.

- In static window control the number of rejections at the destination and thus the number of retransmissions of the internet messages can increase without bound with the increasing load due to the limited network resources. This will cause a steep increase in the delay of messages when the system is overloaded. The dynamic control algorithms reduce the number of rejections by limiting the number of messages in transmission. As a result large message delays are prevented. Dynamic control algorithms are superior to static window control in end-to-end delay of internet messages.

- Dynamic control algorithms yield satisfactory network performance under different load conditions, and different network patterns. Various system parameters (i.e. number of networks within internet, gateway buffersize, gateway service rate) have been used in the evaluation of fairness of the algorithms. It has been observed that network performance is not sensitive to the values of different system parameters when the dynamic control algorithms are applied.

- The comparison of two dynamic control algorithms under various conditions shows that, in order to prevent the overload of destination resources, it is better to control the external message traffic at the destination rather than at the source of messages. Better performance results are obtained with the application of Algorithm1 under heavy load conditions. This results from the fact that Algorithm1 is more effective in restricting the number of messages in transmission for the large values of internet load. In case of a possible congestion, a destination network can throttle all source networks that send messages to it. In the 2nd algorithm source networks try to prevent congestion at the destinations of their messages by estimating the current load at the destinations.

5. CONCLUSION

We have evaluated some window-based congestion control mechanisms in interconnected networks. In the first part of the thesis, an internetwork simulation model has been provided to investigate the effect of internetwork traffic on the gateways and connected networks. Internetwork message transfer has been controlled by a window mechanism. The performance of a network connected to internet, has been evaluated under different internet load conditions. It has been observed that it is not possible to set a proper window size value that performs well under all internetwork load conditions. For the light values of offered internet load it is desirable to operate the system with large window size values to make full use of the resources. For the large values of internet load, more rejection of messages at the destination has been observed due to the limited capacity of the resources. This has resulted in an increase in average response time of internet messages. In this case it is better to reduce window size for fewer number of rejections. As a result, it has been determined that it is desirable to adjust the window size in accordance with the availability of the network resources and with the possibility of congestion at the destination.

In the second part of the thesis work, two dynamic window congestion control algorithms have been proposed and compared with static window control. The dynamic algorithms have been evaluated as effective and fair in controlling the congestion in the proposed internetwork model. It has also been concluded that control of the internetwork messages at the destination gateway yields better performance results than the control at the source gateway of the message.

REFERENCES

[1] Abramson, N. "The Throughput of Packet Broadcasting Channels", IEEE Trans. Comm., Jan. 1977, pp.117-128.

[2] Benhamou, E., Estrin, J. "Multilevel Internetworking Gateways: Architecture and Applications", IEEE Computer, Sept. 1983, pp.27-34.

[3] Bux, W. "Performance Issues in Local Area Networks", IBM Systems Journal, vol.23, 1984, pp. 351-374.

[4] Bux, W., Grillo, D. "Flow Control in Local Area Networks of Interconnected Token Rings", IEEE Trans. Comm., Oct.1985, pp.1058-1066.

[5] Cashin, P. M. "Datapac Network Protocols", Proc. 3rd. Int. Conf. Comput. Commun., 1976, pp.150-155.

[6] Chang, J., Liu, C. "A Two Level Input Buffer Limit Congestion Control", Proc. of IEEE INFOCOM, April 1984, pp. 81-86.

[7] Chou, W. "Computer Communications, Systems and Applications", Prentice-Hall, 1985.

[8] Cypser, R.J. "Communications Architecture for Distributed Systems", Addison-Wesley, 1978.

[9] Davies, D. "The Control of Congestion in Packet-Switching Networks", IEEE Trans. Comm., June 1972, pp.546-550.

[10] Dixon, R.C. "A Token Ring Network for Local Data Communications", IBM Systems Journal, vol. 22, 1983, pp. 47-61.

[11] Gerla, M., Kleinrock, L. "Flow Control: A Comparative Survey", IEEE Trans. Comm., April 1981, pp.553-574.

[12] Gerla, M. "Routing and Flow Control in Virtual Circuit Networks", Advances in Comm., Reidel Publishing Company, 1980, pp. 386-394.

[13] Gerla, M., Chan, W. "Window Selection in Flow Controlled Networks", Proc. Data Comm. Symp., Sept. 1985, pp.84-92.

[14] Hammond, J., O'Reilly, P. "Performance Analysis of Local Computer Networks", Addison Wesley, 1986.

[15] Hanle, J. "Congestion Control in Packet Networks", Advances in Comm., 1980, Reidel Publ. Company, pp.367-376.

[16] Harrison, P. "An Analytical Model for Flow Control Schemes in Communication Network Nodes", IEEE Trans. Comm., Sept.1984,pp.1013-1019.

[17] Hinden, R., Haverty, J., Sheltzer, A. "The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways", Computer, Sepr. 1983, pp. 38-48.

[18] Ilyas, M., Mouftah, T. "End-to-end Flow Control with a New Hybrid Switching in Computer Networks", Conf. Proc. IEEE GLOBECOM, Dec.1982, pp.366-370

[19] Ilyas, M. "On Isarithmic Flow Control in Computer Networks", Conf. Proc. IEEE Southeastcon, 1985, pp.212-216.

[20] Ilyas, M., Mouftah, H., Bhatia, S. "Performance Evaluation of Isarithmic Congestion Avoidance in Computer Networks", IEEE Global Tel. Conf., 1986, pp.155-159.

[21] Ilyas, M., Bhatia, S. "Flow Control in Local Area Networks", IEEE Southeastcon,1987, pp.153-155.

[22] Irland, M. "Buffer Management in a Packet Switch", IEEE Trans. Comm., Mar. 1978, pp.328-337.

[23] Jain, R. "A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks", IEEE Journal on Sel. Areas in Comm., 1986, pp.1162-1167.

[24] Kamoun, F. "A Drop and Throttle Flow Control Policy for Computer Networks", IEEE Trans. Comm., April 1981, pp.444-452.

[25] Kermani, P., Kleinrock, L. "Dynamic Flow Control in Store and Forward Computer Networks", IEEE Trans. Comm., Feb. 1980, pp.263-271.

[26] Kleinrock, L., Kermani, P. "Static Flow Control in Store and Forward Computer Networks", IEEE Trans. Comm., Feb. 1980, pp.271-279.

[27] Lam, S. "Congestion Control Techniques for Packet Networks", Advances in Comm., 1980, Reidel Publ. Company, pp.377-385.

[28] McQuillan, J., Walden, D. "The ARPA Network Design Decisions", Computer Networks, 1977, pp.243-289.

[29] Nagle, J. "Congestion Control in IP/TCP Internetworks", Computer Comm. Review, June 1984, pp. 11-17.

[30] Nassehi, M., Tobagi, F. "Performance of End-to-end and Gateway-to-gateway Flow Control Procedures in Internet Environments", Proc. IEEE Conf.on Decision and Control, Dec.1982, pp.112-119.

[31] Pouzin, L. "Methods, Tools, and Observations on Flow Control in Packet Switched Data Networks", IEEE Trans. Comm., April 1981, pp.413-426.

[32] Raubold, E., Haenle, J. "A Method of Deadlock Free Resource Allocation and Flow Control in Packet Networks", Proc. Intern. Conf. on Computer Commun., Aug.1976.

[33] Reiser, M. "A Queuing Network Analysis of Computer Communication Networks with Window Flow Control", IEEE Trans. Comm., Aug. 1979, pp. 1199-1209.

[34] Sauer, C.H., Chandy, K.M. "Computer Systems Performance Modeling", Prentice Hall, 1981.

[35] Schwartz, M., Saad, S. "Analysis of Congestion Control Techniques in Computer Communication Networks", Febr. 1979, pp.113-130.

[36] Schwartz, M. "Performance Analysis of the SNA Virtual Route Pacing Control", IEEE Trans. Comm., Jan. 1982, pp. 172-184.

[37] Schwartz, M. "Telecommunication Networks", Addison-Wesley Publishing Company, 1987.

[38] Sproule, D., Mellor, F. "Routing, Flow and Congestion Control in the DATAPAC Network", IEEE Trans. Comm., April 1981, pp.386-391.

[39] Tanenbaum, A. "Computer Networks", Prentice Hall, 1981.

[40] Thomasian, A., Bay, P. "Performance Analysis of Window Flow Control for Multiple Virtual Routes", Proc. of IEEE INFOCOM, April 1984, pp.69-80.

[41] Tymes, L. "TYMNET, A Terminal Oriented Communication Network", Proc. AFIPS, Spring Joint Comput. Conf., 1971, pp. 211-216.

[42] Tymes, L. "Routing and Flow Control in TYMNET", IEEE Trans. Comm., April 1981, pp.392-398.

[43] Ulusoy, Ö., Baray, M. "A Survey on Congestion Control in Computer Networks", Technical Report, Bilkent University, SERC, Dec.1987.

[44] Ulusoy, Ö., Baray, M. "An Evaluation of Interconnected Network Congestion Control Based on a Window Mechanism",Technical Report, Bilkent University, SERC, 1988.

[45] Ulusoy, Ö., Baray, M. "Dynamic Window Congestion Control Algorithms in Interconnected Computer Networks", Technical Report, Bilkent University, SERC, 1988.

[46] Varakulsiripunth, R., Shiratory, N., Nogushi, S. "A Congestion Control Policy on the Internetwork Gateway", Computer Networks and ISDN Systems 11, 1986, pp.43-58.

[47] Wong, W., Unsoy, M. "Analysis of Flow Control on Switched Data Networks", Proc. IFIP Congress, Aug. 1977, pp.315-320.

[48] Yum, P., Dou, C. "Design Algorithms for Buffer Allocation Strategies in a Computer Network Node", Conf. Proc. IEEE GLOBECOM, Dec. 1982, pp. 371-375.

A. DERIVATION OF AN EXPRESSION FOR THE RATE OF INTERNET PACKETS PROCESSED AT DESTINATION GATEWAY

In this appendix an expression is derived for the rate of internetwork messages processed at GW_i (gateway i) to its destination network NW_i.

Assuming that there are k networks connected to the internet, the following parameters can be defined for $1 \leq j \leq k$:

$W_{ji}(packets)$: Window size for internet messages from NW_j to NW_i where $j \neq i$.

$\omega_{ji}(packets)$: Number of packets (from NW_j to NW_i) currently in the system.

$\omega_i = (\omega_{1i}, \omega_{2i}, ..., \omega_{ki})$: Collection of currently existing packets destined to NW_i.

$\gamma_{ji}(packets/second)$: Generation rate of internet messages at NW_j to be destined to NW_i where $j \neq i$. The arrival process of internet messages is poisson.

$\lambda_{ji}(\omega_{ji})(packets/second)$: Effective arrival rate of internet messages from NW_j to NW_i.

$$\lambda_{ji}(\omega_{ji}) = \begin{cases} \gamma_{ji} & \text{if} \quad \omega_{ji} \leq W_{ji} \\ \\ 0 & \text{otherwise} \end{cases}$$

Total effective arrival rate of internet messages to GW_i:

$$\lambda_i(\omega_i) = \sum_{j=1}^{k} \lambda_{ji}(\omega_{ji})$$

$\mu_i(packets/second)$: Message processing rate at GW_i. Service time distribution is exponential.

$m_i(packets)$: Buffer capacity of GW_i. The packets residing on the buffers are serviced by FCFS (First Come First Served) queuing discipline.

$GO_i(packets/second)$: Rate of internet packets processed at GW_i to its destination (i.e.NW_i). We will derive an expression for this parameter.

Let's define state n as the number of packets in the buffer pool of GW_i $(0 \leq n \leq m_i)$, and probability functions:

$P(n)$: Probability that the system is in state n.

$P_L(n)$: Probability of leaving state n.

$P_E(n)$: Probability of entering in state n.

$$P_L(n) = (\delta_1 \lambda_i(\omega_i) + \delta_2 \mu_i)P(n))$$

since an arrival $(\lambda_i(\omega_i))$ or departure (μ_i) changes the state.

In the above equation δ_1 and δ_2 are two Kronecker deltas defined as:

$$\delta_1 = \begin{cases} 1 & \text{if} \quad 0 \leq n < m_i \\ \\ 0 & \text{if} \quad n = m_i \end{cases}$$

(i.e. if the buffer is full, new arrivals will be rejected)

$$\delta_2 = \begin{cases} 1 & \text{if} \quad 0 < n \leq m_i \\ \\ 0 & \text{if} \quad n = 0 \end{cases}$$

(i.e. packet processing is not possible if buffer is empty)

$$P_E(n) = \lambda_i(\omega_i)P(n-1) + \mu_i P(n+1)$$

(note that $P(-1) = P(n > m_i) = 0$)

In equilibrium, the number of messages entering into and leaving out of

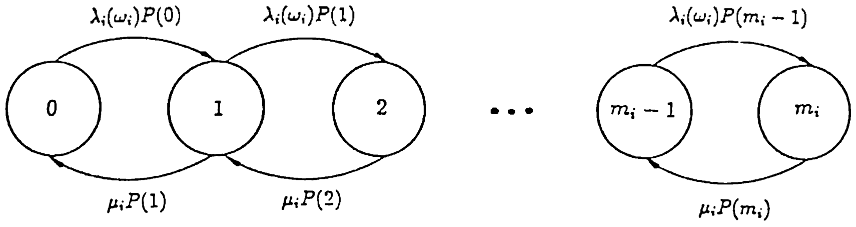

$\lambda_i(\omega_i)P(0)$
$\lambda_i(\omega_i)P(1)$
$\lambda_i(\omega_i)P(m_i-1)$
0
1
2
m_i-1
m_i
$\mu_i P(1)$
$\mu_i P(2)$
$\mu_i P(m_i)$

Figure A.1: State diagram of the gateway queue

a state will be equal. So, by equating $P_L(n)$ and $P_E(n)$ we get:

$$(\delta_1\lambda_i(\omega_i)+\delta_2\mu_i)P(n)) = \lambda_i(\omega_i)P(n-1)+\mu_i P(n+1) \qquad (1)$$

Fig.A.1 gives the states and transition rates between states. We can get an expression for $P(n)$ by solving equation (1) iteratively.

From Fig.A.1:

$$\begin{gathered}\lambda_i(\omega_i)P(0) = \mu_i P(1)\\ \lambda_i(\omega_i)P(1) = \mu_i P(2)\\ .\\ .\\ \lambda_i(\omega_i)P(m_i-1) = \mu_i P(m_i)\end{gathered}$$

Introducing $p = \lambda_i(\omega_i)/\mu_i$, we can write

$$\begin{gathered}P(1) = pP(0)\\ P(2) = p^2 P(0)\\ .\\ .\\ P(m_i) = p^{m_i}P(0)\end{gathered}$$

In general,

$$P(n) = p^n P(0) \qquad 0 \le n \le m_i$$

To eliminate $P(0)$, we use the fact that the probabilities must sum to 1:

$$\sum_{n=0}^{m_i} P(n)=1, \text{ i.e. } \sum_{n=0}^{m_i} p^n P(0)=1$$

Using the well known formula

$$\sum_{n=0}^{m_i} p^n = \frac{1-p^{m_i+1}}{1-p}$$

we can get an expression for P(0):

$$P(0) = \frac{1-p}{1-p^{m_i+1}}$$

and using this expression we get the state probability:

$$P(n) = \frac{p^n(1-p)}{1-p^{m_i+1}}$$

The probability of dropping incoming packets from GW_i (blocking probability) can be given as

$$P(n = m_i) = \frac{p^{m_i}(1-p)}{1-p^{m_i+1}}$$

We can get the expression for the rate of packets processed at GW_i to its destination network NW_i by using the blocking probability:

$$GO_i = \lambda_i(\omega_i)(1 - P(m_i))$$

After final substitutions,

$$GO_i = \lambda_i(\omega_i)(1 - (\lambda_i(\omega_i)/\mu_i)^{m_i})/(1 - (\lambda_i(\omega_i)/\mu_i)^{m_i+1})$$

B. THE SOURCE PROGRAM SIMULATING A CSMA/CD NETWORK WITHIN AN INTERNETWORK ENVIRONMENT

```
PROGRAM INTERNETSIM;  (*MAIN*)

  TYPE

    TYPELIST = 1..4;
    FLAGLIST = 1..2;
    SOURCELIST = 1..20;
    MESPTR = ^MESSAGE;  (*Structure of a message*)
    MESSAGE = RECORD
      MESNO : INTEGER;
      ARRTIME : REAL;  (*Generation time*)
      TIME : REAL;  (*Time of current event*)
      MESTYPE : TYPELIST;  (*1:internetwork, 2:intranetwork*)
      SOURCE : INTEGER;
      COLLCOUNT : INTEGER; (*Number of collisions(in CSMA/CD protocol)*)
      NEXT : MESPTR;  (*Next message in the list*)
    END;

  VAR
    SIMTIME :REAL;
    BUFFERSIZE : INTEGER;
    INTER_ARR_RATE : REAL;
    INTRA_ARR_RATE : REAL;
    SERVICERATE : REAL;
```

```
GWNUMBER : INTEGER;
NODENUMBER : INTEGER;
WINDOWSIZE : INTEGER;
NWCAPACITY : INTEGER;
MESLENGTH : INTEGER;
MESDELAY : REAL;
PROPDELAY : REAL;
TIMEOUT : REAL;

INTERTHRU : INTEGER;
INTRATHRU : INTEGER;
SUMINTERDELAY : REAL;
SUMINTRADELAY : REAL;
NROFREJECT : INTEGER;
NROFRETRANS : INTEGER;
NROFCOLL : INTEGER;

MESNUMBER : INTEGER;
WINDOW : ARRAY [SOURCELIST] OF INTEGER;

QUEUE : RECORD   (*For collecting statistics at
                   destination gateway queue*)
   LENGTH : INTEGER;
   MEANSERVICE : REAL;
   LASTCLOCK : REAL;
   TOTALTIME : REAL;
   TOTALBUSYTIME : REAL;
   COMPLETIONS : INTEGER;
END;

FIRSTMES, LASTMES, CURRENT : MESPTR;
QUEUEFIRST, QUEUELAST : MESPTR;
GWLIST_FIRST, GWLIST_LAST : ARRAY [SOURCELIST] OF MESPTR;

CLOCK : REAL;
FULLWINDOWLIST : SET OF SOURCELIST;
MIAT : REAL;
GWACTIVE : FLAGLIST; (*If GW is currently trying to send a message*)
TRANSENDTIME : REAL;
```

```
  INTERMES, INTRAMES : INTEGER;
  READYGWMES:INTEGER;
  KEYPR : CHAR;
  FILEIN, FILEOUT : TEXT;

PROCEDURE INSERT (PUT : MESPTR; VAR FRONT, TAIL : MESPTR);
(*Insert a message pointed by 'PUT' to the message list*)

  VAR
    TEMP : MESPTR;

  BEGIN
    IF FRONT=NIL THEN
      BEGIN (*List was empty*)
        FRONT:=PUT;
        TAIL:=PUT;
        PUT^.NEXT:=NIL;
      END
    ELSE IF PUT^.TIME<FRONT^.TIME THEN
      BEGIN (*Insert at beginning of list*)
        PUT^.NEXT:=FRONT;
        FRONT:=PUT;
      END
    ELSE IF PUT^.TIME>=TAIL^.TIME THEN
      BEGIN (*Insert at end of list*)
        TAIL^.NEXT:=PUT;
        TAIL:=PUT;
        PUT^.NEXT:=NIL;
      END
    ELSE
      BEGIN (*Insert into proper place in list*)
        TEMP:=FRONT;
        WHILE PUT^.TIME>=TEMP^.NEXT^.TIME DO
          TEMP:=TEMP^.NEXT;
        PUT^.NEXT:=TEMP^.NEXT;
        TEMP^.NEXT:=PUT;
      END;
END; (*INSERT*)
```

```
  PROCEDURE DELETE(TAKE:MESPTR; VAR FRONT, TAIL : MESPTR);
  (*Delete a message from list pointed by 'TAKE'*)

    VAR
      TEMP : MESPTR;

    BEGIN
      IF TAKE=FRONT THEN
        BEGIN (*Delete first message in list*)
          FRONT:=FRONT^.NEXT;
          IF FRONT=NIL THEN TAIL:=NIL;
        END
      ELSE
        BEGIN
          TEMP:=FRONT;
          WHILE (TEMP^.NEXT<>TAKE) DO
            TEMP:=TEMP^.NEXT;
          TEMP^.NEXT:=TAKE^.NEXT;
          IF TEMP^.NEXT=NIL THEN TAIL:=TEMP;
        END;
   END;(*DELETE*)


 FUNCTION EXPON(A:REAL) : REAL;
 (*Generate a random variable EXPON using cumulative*)
 (*exponential distribution with mean A             *)

   BEGIN
     EXPON:=-A*LN(1-RANDOM);
   END;

PROCEDURE MESGEN(GENTIME : REAL; ORIGIN : INTEGER; MTYPE : TYPELIST);
(*Generate a message*)

  VAR
    NEWMES : MESPTR;

  BEGIN
    NEW(NEWMES);
    WITH NEWMES^ DO
```

```
      BEGIN
        MESNO:=MESNUMBER;
        ARRTIME:=GENTIME;
        TIME:=ARRTIME;
        MESTYPE:=MTYPE;
        SOURCE:=ORIGIN;
        COLLCOUNT:=0;
      END;
     MESNUMBER:=MESNUMBER+1;
    INSERT(NEWMES,FIRSTMES,LASTMES);
  END; (*MESGEN*)



PROCEDURE INITIALIZATION;

  VAR
    I : INTEGER;

  PROCEDURE INIT_MES_GEN;
  (*Initial message generation*)

    VAR
      I : INTEGER;
      TIME,MIAT : REAL;

    BEGIN  (*INIT_MES_GEN*)
      MIAT:=1.0/INTER_ARR_RATE;
      FOR I:=1 TO GWNUMBER DO  (*Generate internet messages*)
        BEGIN
          TIME:=EXPON(MIAT);
          WHILE (TIME<SIMTIME) DO
            BEGIN
              MESGEN(TIME,I,INTERMES);
              TIME:=TIME+EXPON(MIAT);
            END;
        END;
      MIAT:=1.0/INTRA_ARR_RATE;
      FOR I:=1 TO NODENUMBER DO  (*Generate intranet messages*)
      BEGIN
        TIME:=EXPON(MIAT);
        MESGEN(TIME,I,INTRAMES);
      END;
  END; (*INIT_MES_GEN*)



BEGIN  (*INITIALIZATION*)
  ASSIGN(FILEIN,'FILEIN');
  ASSIGN(FILEOUT,'FILEOUT');REWRITE(FILEOUT);
  WRITELN('Enter source for simulation data: ');
  WRITELN('Enter "D" for disk input, "M" for manual input');
  KEYPR:=' ';
  WHILE (UPCASE(KEYPR) IN ['D','M'])=FALSE DO
    READ(KBD,KEYPR);
  IF UPCASE(KEYPR)='M' THEN
    BEGIN
      REWRITE(FILEIN);
      WRITE('Enter number of gateways > '); READLN(GWNUMBER);
        WRITELN(FILEIN,GWNUMBER);
      WRITE('Enter internet mes. arrival rate > ');
      READLN(INTER_ARR_RATE);
        WRITELN(FILEIN,INTER_ARR_RATE);
      WRITE('Enter window size > '); READLN(WINDOWSIZE);
        WRITELN(FILEIN,WINDOWSIZE);
      WRITE('Enter gateway service rate > '); READLN(SERVICERATE);
        WRITELN(FILEIN,SERVICERATE);
      WRITE('Enter gateway buffer size > '); READLN(BUFFERSIZE);
        WRITELN(FILEIN,BUFFERSIZE);
      WRITE('Enter propagation delay > '); READLN(PROPDELAY);
        WRITELN(FILEIN,PROPDELAY);
      WRITE('Enter timeout for internet messages > ');
      READLN(TIMEOUT);
        WRITELN(FILEIN,TIMEOUT);

      WRITE('Enter number of nodes within network > ');
      READLN(NODENUMBER);
        WRITELN(FILEIN,NODENUMBER);
    WRITE('Enter intranet mes. arrival rate > ');
    READLN(INTRA_ARR_RATE);
      WRITELN(FILEIN,INTRA_ARR_RATE);
    WRITE('Enter network link speed > '); READLN(NWCAPACITY);
      WRITELN(FILEIN,NWCAPACITY);
    WRITE('Enter average intranet mes. length  > ');
    READLN(MESLENGTH);
      WRITELN(FILEIN,MESLENGTH);
    WRITE('Enter simulation time > '); READLN(SIMTIME);
      WRITELN(FILEIN,SIMTIME);

  END
ELSE
  BEGIN
    RESET(FILEIN);
    READ(FILEIN,GWNUMBER); READ(FILEIN,INTER_ARR_RATE);
    READ(FILEIN,WINDOWSIZE); READ(FILEIN,SERVICERATE);
    READ(FILEIN,BUFFERSIZE); READ(FILEIN,PROPDELAY);
    READ(FILEIN,TIMEOUT);
    READ(FILEIN,NODENUMBER); READ(FILEIN,INTRA_ARR_RATE);
    READ(FILEIN,NWCAPACITY); READ(FILEIN,MESLENGTH);
    READ(FILEIN,SIMTIME);
  END;
CLOSE(FILEIN);
WRITELN(FILEOUT,' gateways > ',GWNUMBER);
WRITELN(FILEOUT,'gateway service rate > ',SERVICERATE);
WRITELN(FILEOUT,'gateway buffer size > ',BUFFERSIZE);
WRITELN(FILEOUT,'propagation delay > ',PROPDELAY);
WRITELN(FILEOUT,'timeout for internet messages > ',TIMEOUT);
WRITELN(FILEOUT,'number of nodes within network > ',NODENUMBER);
WRITELN(FILEOUT,'intranet mes. arrival rate > ',INTRA_ARR_RATE);
WRITELN(FILEOUT,'network link speed > ',NWCAPACITY);
WRITELN(FILEOUT,'average intranet mes. length  > ',MESLENGTH);

FOR I:=1 TO GWNUMBER DO
  BEGIN
    WINDOW[I]:=0;
    GWLIST_FIRST[I]:=NIL; GWLIST_LAST[I]:=NIL;
  END;
FIRSTMES:=NIL; LASTMES:=NIL;
```

```
    WITH QUEUE DO  (*Initialization for queue statistics*)
      BEGIN
        LENGTH:=0;
        MEANSERVICE:=1.0/(SERVICERATE*MESLENGTH);
        LASTCLOCK:=0.0;
        TOTALTIME:=0.0;
        TOTALBUSYTIME:=0.0;
        COMPLETIONS:=0;
      END;
    MESNUMBER:=1;
    NROFREJECT:=0;
    NROFRETRANS:=0;
    NROFCOLL:=0;
    QUEUEFIRST:=NIL; QUEUELAST:=NIL;
    INTERMES:=1; INTRAMES:=2; (*Message types*)
    GWACTIVE:=0;
    FULLWINDOWLIST:=[];
    TRANSENDTIME:=0;
    MESDELAY:=MESLENGTH/NWCAPACITY;
    INTERTHRU:=0; INTRATHRU:=0;
    SUMINTERDELAY:=0; SUMINTRADELAY:=0;
    INIT_MES_GEN;
  END; (*INITIALIZATION*)




PROCEDURE INTERMESTRANS(VAR MES : MESPTR);
(*An internet message is transmitted to its destination network*)
(*through the adjacent gateway of the network                  *)

  VAR
    I : INTEGER;
    GWMES,NEXTMES : MESPTR;

  BEGIN (*INTERMESTRANS*)
    I:=MES^.SOURCE;
    IF (I IN FULLWINDOWLIST) AND (MES^.MESTYPE=1) THEN
      BEGIN  (*If the window of the first network is closed,
               do not send the message*)
    NEXTMES:=MES^.NEXT;
    DELETE(MES,FIRSTMES,LASTMES);
    DISPOSE(MES);
    MES:=NEXTMES;
    IF MES<>NIL THEN CLOCK:=MES^.TIME;
  END
ELSE
 BEGIN
   IF MES^.MESTYPE=1 THEN  (*If it is not a retransmitted
                             message*)
     BEGIN
       WINDOW[I]:=WINDOW[I]+1;
       IF WINDOW[I]=WINDOWSIZE THEN
         FULLWINDOWLIST:=FULLWINDOWLIST+[I];
       DELETE(MES,FIRSTMES,LASTMES);
     END;
   NEW(GWMES); (*Generate a copy of message to
                 store at its source*)
   GWMES^:=MES^;
   INSERT(GWMES,GWLIST_FIRST[I],GWLIST_LAST[I]);
   IF QUEUE.LENGTH<BUFFERSIZE THEN
     BEGIN
       IF GWACTIVE=0 THEN  (*If gateway have no
                             message to send*)
         BEGIN
           MES^.TIME:=MES^.TIME+
                      (QUEUE.MEANSERVICE*EXPON(MESLENGTH));
                         (*Service rate is exponential*)
           MES^.MESTYPE:=4;  (*Make it ready to be
                          transmitted within the network*)
           READYGWMES:=MES^.MESNO;
           INSERT(MES,FIRSTMES,LASTMES);
           GWACTIVE:=1;
        END
      ELSE INSERT(MES,QUEUEFIRST,QUEUELAST);
      WITH QUEUE DO
        BEGIN
          TOTALTIME:=TOTALTIME+(CLOCK-LASTCLOCK)*LENGTH;
          LENGTH:=LENGTH+1;
          IF LENGTH<>1 THEN
```

```
                  TOTALBUSYTIME:=TOTALBUSYTIME+(CLOCK-LASTCLOCK);
                LASTCLOCK:=CLOCK;
              END;
          END
        ELSE
          BEGIN  (*Reject it*)
            DISPOSE(MES);
            NROFREJECT:=NROFREJECT+1;
          END;
        IF MES^.MESTYPE<>3 THEN
          BEGIN
            MES:=FIRSTMES;
            IF MES<>NIL THEN
              CLOCK:=MES^.TIME;
          END;
     END;
  END; (*INTERMESTRANS*)


PROCEDURE INTRAMESTRANS (VAR MES:MESPTR);
(*A message is transmitted within the network using*)
(*CSMA/CD link access protocol                     *)

  VAR
    TEMP, QUEUEMES : MESPTR;
    TIMEDIFF, DELAY, COLLINTERVAL,GTIME : REAL;


  PROCEDURE RESCHEDULE  (VAR MESSAGE:MESPTR);
  (*Reschedule the collided message*)

    VAR
      DELAY:REAL;
      STORE:MESPTR;

    BEGIN
      STORE:=MESSAGE^.NEXT;
      WITH MESSAGE^ DO
        BEGIN
          DELETE(MESSAGE,FIRSTMES,LASTMES);
```

```
         IF COLLCOUNT=0 THEN COLLCOUNT:=2
         ELSE COLLCOUNT:=COLLCOUNT*2;
         DELAY:=MESDELAY*COLLCOUNT;
         TIME:=TIME+RANDOM*DELAY;
       END;
     INSERT(MESSAGE,FIRSTMES,LASTMES);
     MESSAGE:=STORE;
   END; (*RESCHEDULE*)
```

```
 PROCEDURE POSTPONE(VAR MESSAGE:MESPTR);
 (*Postpone the transmission because of busy link*)

   VAR
     NEXTMES:MESPTR;

   BEGIN
     NEXTMES:=MESSAGE^.NEXT;
     DELETE(MESSAGE,FIRSTMES,LASTMES);
     MESSAGE^.TIME:=MESSAGE^.TIME+2*MESDELAY*RANDOM;
     INSERT(MESSAGE,FIRSTMES,LASTMES);
     MESSAGE:=NEXTMES;
   END;
```

```
PROCEDURE CHECKFORRESCHEDULE(PERIOD:REAL; SCHTYPE:FLAGLIST);

  VAR
    FLAG:FLAGLIST;

  BEGIN
    FLAG:=0;
    IF TEMP<>NIL THEN FLAG:=1;
    WHILE (FLAG=1) DO
      BEGIN
        FLAG:=0;
        WHILE ((TEMP<>NIL) AND (TEMP^.MESTYPE=1)) DO
```

```
             TEMP:=TEMP^.NEXT;
         IF TEMP<>NIL THEN
           BEGIN
             TIMEDIFF:=TEMP^.TIME-MES^.TIME;
             IF TIMEDIFF<PERIOD THEN
               BEGIN
                 FLAG:=1;
                 IF SCHTYPE=1 THEN
                   RESCHEDULE(TEMP)
                 ELSE
                   POSTPONE(TEMP);
                 COLLINTERVAL:=TIMEDIFF;
               END;
           END;
      END;
  END; (*CHECKFORRESCHEDULE*)


PROCEDURE SENDMES;
  (*Transmit the message within the network*)

  VAR
    I : INTEGER;

  BEGIN
    TRANSENDTIME:=CLOCK+EXPON(MESDELAY)+PROPDELAY;
    IF MES^.MESTYPE=4 THEN  (*If the message is transmitted from
                              the gateway*)
      BEGIN
        WITH QUEUE DO
          BEGIN
            TOTALTIME:=TOTALTIME+(CLOCK-LASTCLOCK)*LENGTH;
            LENGTH:=LENGTH-1;
            TOTALBUSYTIME:=TOTALBUSYTIME+(CLOCK-LASTCLOCK);
            COMPLETIONS:=COMPLETIONS+1;
            LASTCLOCK:=CLOCK;
          END;
        QUEUEMES:=QUEUEFIRST;
        IF QUEUEMES<>NIL THEN
          BEGIN
```

```
            DELETE(QUEUEMES,QUEUEFIRST,QUEUELAST);
            QUEUEMES^.MESTYPE:=4;
            READYGWMES:=QUEUEMES^.MESNO;
            QUEUEMES^.TIME:=QUEUEMES^.TIME+QUEUE.MEANSERVICE;
            IF QUEUEMES^.TIME<CLOCK THEN
              QUEUEMES^.TIME:=CLOCK;
            INSERT(QUEUEMES,FIRSTMES,LASTMES);
          END
        ELSE GWACTIVE:=0;
        I:=MES^.SOURCE;
        QUEUEMES:=GWLIST_FIRST[I];
        WHILE (QUEUEMES^.MESNO<>MES^.MESNO)  DO
          QUEUEMES:=QUEUEMES^.NEXT;
        DELETE(QUEUEMES,GWLIST_FIRST[I],GWLIST_LAST[I]);
        (*Delete the copy of the message from queue of
            the source network*)
        DISPOSE(QUEUEMES);
        WINDOW[I]:=WINDOW[I]-1;
        IF I IN FULLWINDOWLIST THEN
        FULLWINDOWLIST:=FULLWINDOWLIST-[I];
     END;
   IF (CLOCK+MESDELAY)<=SIMTIME THEN
     BEGIN
       DELAY:=CLOCK+MESDELAY-MES^.ARRTIME;
       IF MES^.MESTYPE=4 THEN  (*Internet message*)
         BEGIN
           INTERTHRU:=INTERTHRU+1;
           SUMINTERDELAY:=SUMINTERDELAY+DELAY;

         END
       ELSE  (*Intranet message*)
         BEGIN
           INTRATHRU:=INTRATHRU+1;
           SUMINTRADELAY:=SUMINTRADELAY+DELAY;
           MIAT:=1.0/INTRA_ARR_RATE;
           GTIME:=CLOCK+EXPON(MIAT);
           MESGEN(GTIME,MES^.SOURCE,INTRAMES);
         END;
    END;
END; (*SENDMES*)
```

```
BEGIN (*INTRAMES*)
  IF MES^.TIME<TRANSENDTIME THEN  (*If link is busy*)
    POSTPONE(MES)
  ELSE
    BEGIN (*Try to send the message*)
      TEMP:=MES^.NEXT;
      WHILE ((TEMP<>NIL) AND (TEMP^.MESTYPE=1)) DO
        TEMP:=TEMP^.NEXT;
      IF TEMP=NIL THEN
        BEGIN
          SENDMES; (*Send message since no other intranet message*)
          DELETE(MES,FIRSTMES,LASTMES);
          DISPOSE(MES);
        END
      ELSE
        BEGIN
          TIMEDIFF:=TEMP^.TIME-MES^.TIME;
          IF TIMEDIFF>PROPDELAY THEN
            BEGIN (*Success - no collision, message is sent*)
              SENDMES;
              DELETE(MES,FIRSTMES,LASTMES);
              DISPOSE(MES);
            END
            ELSE (*collision*)
              BEGIN
                TEMP:=MES^.NEXT;
                COLLINTERVAL:=0.0;
                CHECKFORRESCHEDULE(PROPDELAY,1);
                DELAY:=COLLINTERVAL+PROPDELAY;
                CHECKFORRESCHEDULE(DELAY,0);
                RESCHEDULE(MES);
                NROFCOLL:=NROFCOLL+1;
              END;
      END;
  END;
  MES:=FIRSTMES;
  IF MES<>NIL THEN
    CLOCK:=MES^.TIME;
  END; (*INTRAMESTRANS*)


PROCEDURE CHECKTIMEOUT;
(*Check the queue of each gateway to see if there is any message*)
(*whose timeout has expired. Retransmit such messages.          *)

  VAR
    I : INTEGER;
    PTR, NEXTPTR, MESLISTPTR : MESPTR;
    FLAG:FLAGLIST;
    QPTR:MESPTR;

  BEGIN
    FOR I:=1 TO GWNUMBER DO
      BEGIN
        PTR:=GWLIST_FIRST[I];
        WHILE (PTR<>NIL) DO
          BEGIN
            NEXTPTR:=PTR^.NEXT;
            IF (PTR^.TIME+TIMEOUT)<=CLOCK THEN
              BEGIN
                NROFRETRANS:=NROFRETRANS+1;
                DELETE(PTR,GWLIST_FIRST[I],GWLIST_LAST[I]);
                PTR^.TIME:=CLOCK;
                FLAG:=0; QPTR:=QUEUEFIRST;
                WHILE ((FLAG=0) AND (QPTR<>NIL)) do
                  BEGIN
                    IF PTR^.MESNO=QPTR^.MESNO THEN FLAG:=1;
                    QPTR:=QPTR^.NEXT;
                  END;
                IF (FLAG<>1) AND (PTR^.MESNO<>READYGWMES) THEN
                   (*If message is not in queue*)
                  BEGIN
                    PTR^.MESTYPE:=3;  (*It is a rejected message*)
                    INTERMESTRANS(PTR);  (*Send it again*)
                  END
                ELSE INSERT(PTR,GWLIST_FIRST[I],GWLIST_LAST[I]);
              END;
            PTR:=NEXTPTR;
          END;
      END;
  END; (*CHECKTIMEOUT*)



BEGIN (*MAIN*)

  INITIALIZATION;
  CURRENT:=FIRSTMES;
  CLOCK:=CURRENT^.TIME;
  WHILE ((CLOCK<=SIMTIME) AND (CURRENT<>NIL)) DO
    BEGIN
      IF CURRENT^.MESTYPE=1 THEN
        INTERMESTRANS(CURRENT)
      ELSE
        INTRAMESTRANS(CURRENT);
      CHECKTIMEOUT;
    END;

 WRITELN(FILEOUT,'INTERTHRU:',INTERTHRU/SIMTIME,' INTRATHRU:',
   INTRATHRU/SIMTIME,' TOTAL:',(INTERTHRU+INTRATHRU)/SIMTIME);
 IF INTERTHRU<>0 THEN
   WRITELN(FILEOUT,'AVERAGE INTERNET DELAY:',SUMINTERDELAY/INTERTHRU)
 IF INTRATHRU<>0 then
 WRITELN(FILEOUT,'AVERAGE INTRANET DELAY:',SUMINTRADELAY/INTRATHRU);
 WRITELN(FILEOUT,'NR OF REJECTIONS PER SEC:',NROFREJECT/SIMTIME,
 'NR OF RETRANS PER SEC:',NROFRETRANS/SIMTIME);
 WRITELN(FILEOUT,'NR OF COLL PER SEC:',NROFCOLL/SIMTIME);
   WITH QUEUE DO
     BEGIN
       TOTALTIME:=TOTALTIME+(SIMTIME-LASTCLOCK)*LENGTH;
       IF LENGTH<>0 THEN
         TOTALBUSYTIME:=TOTALBUSYTIME+(SIMTIME-LASTCLOCK);
       WRITELN(FILEOUT,'...QUEUE STATISTICS...');
       WRITELN(FILEOUT,' QUEUE THRUPUT:',COMPLETIONS,
         ' MEAN QUEUE LENGTH:',TOTALTIME/SIMTIME,
         ' BUSYTIME:',TOTALBUSYTIME);
    END;
  CLOSE(FILEOUT);

END. (*MAIN*)
```