SCHEDULING AND MANAGEMENT OF REAL-TIME COMMUNICATION IN POINT-TO-POINT WIDE AREA NETWORKS

By

Cheryl Lynn Pope

2002

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ADELAIDE

2002

Abstract

Applications with timing requirements, such as multimedia and live multi-user interaction, are becoming more prevalent in wide area networks. The desire to provide more predictable performance for such applications in packet switched wide area networks is evident in the channel management provided by Asynchronous Transfer Mode (ATM) networks and in the extensions to the Internet protocols proposed by the Internet Engineering Task Force (IETF) working groups on integrated and differentiated service. The ability to provide guarantees on the performance of traffic flows, such as packet delay and loss characteristics, relies on an accurate model of the traffic arrival and service at each node in the network.

This thesis surveys the work in bounding packet delay based on various proposed queuing disciplines and proposes a method for more accurately defining the traffic arrival and worst case backlog experienced by packets. The methods are applied to the first in first out (FIFO) queuing discipline to define equations for determining the worst case backlog and queuing delay in multihop networks. Simulation results show a significant improvement in the accuracy of the delay bounds over existing bounds published in the literature. An improvement of two orders of magnitude can be realised for a ten hop path and the improvement increases exponentially with the length of the path for variable rate network traffic. The equations derived in the thesis also take into consideration the effect of jitter on delay, thereby removing the requirement for rate controllers or traffic shaping within the network.

In addition to providing more accurate delay bounds, the problem of providing fault tolerance to channels with guaranteed quality of service (QoS) is also explored. This

thesis introduces a method for interleaving resource requirements of backup channels to reduce the overall resource reservations that are required to provide guaranteed fault recovery with the same QoS as the original failed channel. An algorithm for selecting recovery paths that can meet a channel's QoS requirements during recovery is also introduced.

Declaration

This is to certify that this thesis contains no material which has previously been accepted for the award of any degree or diploma in any University. To the best of my knowledge and belief it contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

If this thesis is accepted for the award of the degree, permission is granted for it to be made available for loan and photocopying.

Cheryl Pope

2002

Acknowledgements

I may never have undertaken this PhD without the enthusiastic encouragement of the person who became my advisor, Dr. Jay Yantchev. He often steered me in the right direction when I wasn't certain which path to investigate and I acknowledge his significant contribution to my PhD studies.

Prof. Chris Barter deserves special mention, not only for providing excellent facilities as Head of the Computer Science department; but also for providing the opportunity for me to spend part of my early candidature at Oxford University. It was during this time that I was able to visit the University of York's real-time computing group and define my research problem.

Spouses are often acknowledged for their significant support and tolerance during PhDs. Few people are as fortunate as I am, however, to have a spouse who has offered not only emotional support but intellectual support as well. For his infinite patience, system administration and LaTex skills, giving me priority on our home computer, being a sounding board for my ideas, not being afraid to actually read and understand the "hairy equations", as well as giving me encouragement when I needed it, I acknowledge my husband, Dr. Michael Pope. I'm very lucky to have him.

Finally I would like to acknowledge my parents for encouraging my interest in maths and science and for the many sacrifices they have made for my studies. I would also like to thank Erika for having some long naps while I was making amendments to this thesis. This thesis is dedicated to her. May she find the world a peaceful place full of discoveries to be made.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Significant changes have occurred during the last decade in the users and the applications that communicate on wide area data networks. The most extensive wide area data network, the Internet, has changed from a primarily research/academic network to include wide spread use by businesses, K-12 education, and home users. Much of this change can be attributed to the introduction of the World Wide Web (WWW) [83], which made the Internet more accessible to those without specialist knowledge of computers. WWW traffic contributed less than one percent of Internet traffic before 1990, but now comprises approximately seventy percent of all Internet traffic. This dominance of WWW traffic continues to increase, and is significantly changing the pattern of the Internet traffic [49], prompting new research in traffic modeling and protocols [65] [97]. The WWW has had such a large effect partially because it enables not just one, but many applications. The initial web browsers that displayed text and graphics web pages have developed to support a wide range of applications including e-commerce, multi-dimensional modeling, and video/sound playback, all of which can occur within a single web page. In addition to the WWW, there is a continued push towards convergence of voice, cable and data networks and an increase in interactive applications, such as networked games, which also influence the nature of current Internet traffic. The trend toward convergence can be seen in the Voice over Internet Protocol (VoIP), which is now used by many companies worldwide to carry phone calls over their IP based data networks. These changes led the network research community to spend much of the 1990s determining the network requirements and behaviour of these new applications and re-evaluating the capability of the existing Internet architecture to meet these requirements.

One of the areas identified as needing additional support in the Internet architecture was a mechanism to provide Quality of Service (QoS). QoS is a specification of level of performance or behaviour from the network that is typically guaranteed to an application or user. The specific parameters which define QoS vary depending on the application and the user requirements. It can be defined in terms ranging from service relative to other network users, such as level one service receiving twice the network resources of level two service, to specific measures such as upper bounds on end-to-end delay experienced by the application. QoS is important to time-sensitive applications such as live voice and video as the perceived performance of these applications can be degraded by large delays in transmission or loss of information in transmission. A certain minimum level of network performance is required for these applications to be usable at all, and higher levels of performance may be desirable to ensure that the error level is not distracting to the user. Studies indicate that degraded video has a significant negative impact on interactions and degraded voice has a significant impact on understanding and learning, some of the main criteria for using the network as a communication tool [69] [98].

The existing Internet architecture has very limited capability to offer QoS and is primarily designed to optimise the overall network performance. Although *type of service* can be specified according to the standard, few implementations actually use the type of service feature [77]. The type of service is also very general specifying only the preference to minimize delay, maximize throughput, maximize reliability or minimize cost (in terms of dollars). Only one of these options can be chosen per application, providing no way to offer different levels of service. For instance, if several applications set the type of service to minimize delay, they will all receive the same level of service regardless of their individual delay requirements. Time-sensitive applications, however, have different levels of sensitivity to delay. For example, voice is more sensitive to delay effects than video. As time-sensitive applications become more prevalent on the Internet, it will become more important to differentiate between time-sensitive and non-time-sensitive applications to allow the time-sensitive applications to receive higher QoS. It will also be desirable to differentiate between different time-sensitive applications to take advantage of the differences in their QoS requirements when determining the total network resources required to service the various applications.

Increasing available resources to accommodate all potential traffic is sometimes proposed as a solution to providing sufficient QoS to time-sensitive applications, however it is not practical for several reasons. First, resources are finite and already the Internet is experiencing data loss rates of approximately eight percent [49]. Clearly the resources are not sufficient. When excess resources are discussed, it is often in the context of backbones. Given the heterogeneity of the Internet, the resources available will vary in different parts of the network. It is likely that the resources are always going to be limited, relative to the traffic, in some parts of the network. To offer an end-to-end QoS, the network must provide a mechanism which works well both when resources are abundant and when resources are scarce. As greater resources do become available and performance improves, it is likely that more people will make greater use of the existing time-sensitive applications; and new applications, which previously were infeasible due to the limited resources, are likely to emerge, thereby increasing the demand for the resources as the availability of those resources increases.

The second reason why just increasing resources is not sufficient is that even if adequate resources were available the efficiency of the network would be compromised if the network has to provide to all traffic a QoS high enough to meet the most stringent requirements (the only option if traffic is not differentiated). The excess resources required to offer high QoS to all network traffic would increase the cost of data communication for all traffic, not just time-sensitive traffic.

This need for isolation of different traffic was a key factor in the design of Asynchronous Transfer Mode (ATM) networks [41]. ATM networks provide the ability to set up multiple virtual circuits through the network with known performance. The performance characteristics can differ on these virtual circuits allowing them to be customized for different traffic groups. To provide the same type of service as ATM, the Internet protocols must be extended to provide mechanisms for reserving network resources along a given path. It should do this while still allowing non time-sensitive applications to be handled with high reliability and efficient use of network resources.

One clear lesson from the past decade is that it is difficult to predict the architecture, users or applications of future wide area networks. Any approach to providing QoS should be flexible enough to be applicable to different network architectures and accommodate the control of the various network resources independently, so that the network can customize the resource requirements to suit the application.

1.1 Differentiated and Integrated Service: A Tale of Two Approaches

The Internet Engineering Task Force (IETF) divides the issue of providing QoS on the network into two areas: differentiated and integrated service. Differentiated service aims to provide QoS to a customer based on the customer's aggregated traffic. The customer and provider negotiate a quantitative agreement (e.g. 100 kbps, 20 kb burst) or qualitative agreement (e.g. traffic treated as higher priority than normal traffic) for service and the provider provisions their network to offer such service. Providers of different parts of the network can create agreements between themselves to offer end-to-end differentiated service. The goal of differentiated service is to be able to offer users different service levels at different costs or, if required, the illusion of an end-to-end leased line.

Differentiated service alone is unlikely to provide a complete solution to the problem of providing QoS to time-sensitive applications. The service agreements in differentiated service are designed for aggregated traffic from the customer, not individual traffic flows (such as a single video stream). The customer is left with the responsibility of ensuring multiple time-sensitive applications originating from their network do not interfere with each other due to their aggregated traffic characteristics exceeding the specification agreed to on the differentiated network. The customer must also be able to determine their communication requirements for the differentiated service agreement based on their existing or expected traffic flows. In order to allow efficient use of the network resources, it is likely that the customer will want to renegotiate their differentiated service to adapt to changes in the application mix. To do this, the customer must be able to determine the changes to required resources as applications join or leave the aggregated traffic.

Integrated service aims to solve this problem by managing traffic at the level of the individual traffic flows. Integrated service manages the flows by having each individual flow requiring a guaranteed QoS provide its traffic specification to the integrated services network elements along the flow's path. These network elements then determine if sufficient resources are available to provide the flow with the requested QoS. If not, then the application must renegotiate for a lower QoS or risk variable performance. If the resources are available, these resources are reserved for the life of the application or until renegotiated. The exact methods for managing the flow at the network elements and determining resource availability are an area of open research and discussion and the main topic of this thesis.

It is unlikely that either integrated or differentiated service will alone solve the problem of providing QoS. Differentiated service merely pushes the problem of managing the time-sensitive traffic onto the user, where it still must be dealt with to provide protection to individual traffic flows. Integrated service raises scalability problems since maintaining state information of all flows at backbone routers is likely to create too heavy a computational burden to be practical. The final solution is likely to involve a combination of both, with integrated service provided on smaller networks and differentiated service provided on the network backbone. This combination meets the requirements of both approaches. Integrated service management can provide accurate specification of the aggregated traffic needed for negotiating an appropriate level of differentiated service; and differentiated service can provide a guaranteed QoS over the network backbone which allows the integrated services at the sending and receiving networks to treat the backbone as a single link with known behaviour in order to provide end-to-end integrated service in a scalable manner.

This flexibility is an important part of any move to combine the various types of existing special purpose networks; phone, data, television, into a single integrated network. These networks have different characteristics which their applications rely upon and these differing characteristics must be integrated to provide the same level of service available on the original networks while maintaining efficient use of finite network resources.

1.2 Managing QoS Agreements

One common characteristic of both integrated and differentiated service is the need to determine from traffic specifications (either per flow or aggregated) the type and quantity of network resources required to meet the QoS requirements. Both services must also have a mechanism for determining if there are sufficient available resources to allow a new traffic flow to be accepted and ensuring the availability of the required resources for all traffic flows that are accepted throughout the lifetime of those flows. These three tasks: determining the network resources required, determining whether to accept traffic and ensuring promised resources remain available will be termed *traffic management*.

The design of the traffic management for a WAN is highly dependent on how QoS will be specified. The QoS is, in turn, dependent on the characteristics of the applications which will use the WAN. The applications of interest in this thesis (i.e. the applications we want to provide QoS to) are part of the class of *real-time* applications.

Real-time applications are applications for which the time when a result occurs is as important to the correctness of the application as the accuracy of the result itself. In the context of networked applications, it is the timeliness of the communication that is critical. It is not sufficient to just receive the data. The data must also arrive within a bounded period of time. Arriving too late or too early can cause QoS bounds to be violated. In live applications (e.g. video-conferencing) it is unlikely that data will arrive too early, but it can arrive too late. If data is delayed significantly the user will be out of sync with the person they are communicating with. For example, a person might “interrupt” another speaker since it can be difficult to determine if the delay is due to the network or due to the person not speaking. Even in non-interactive applications, such as radio programs where the playback can be partially stored and played back with a delay (i.e. streaming), long delays can cause the playback application to catch up with the buffered information causing a pause in the playback. In these applications it is also possible for the delay to be too low. If the receiver’s buffer size is set based on average delays, a large amount of data arriving with low delays in a short period of time can overflow the buffers causing some of the data to be lost.

Real-time applications are typically classified into two groups: hard and soft. Although the distinction between hard and soft real-time is defined in several, often conflicting, ways, the fundamental characteristic in all of the definitions is that hard real-time systems have deadlines which *must* be met. Soft real-time systems have deadlines beyond which it may still be useful to continue processing the event or after which the event can be discarded without adverse effect. Systems such as video/voice can be difficult to classify as the classification is dependent on the QoS desires of the user. Typically such systems are considered soft real-time since they can tolerate loss. However, a user who is recording a live video feed may require perfect or near perfect reception of the video signal. There may be a bound below which the performance does not meet the required QoS. Under these circumstances, the system becomes a hard real-time problem with the requirement to *guarantee* that this lower bound on performance is met. A user who is watching a live performance may be willing to accept a lower quality picture or a variable quality picture with an emphasis on reducing the cost. Under this circumstance, the system becomes a soft real-time problem. Ultimately one must consider the user's requirements for subjective quality as well as the objective behaviour of the application's network traffic to form the QoS specification. This QoS specification can then serve as the hard real-time constraint of the network application.

Several factors can affect the QoS requirements of an application. To most efficiently use the network, the resource requirements should not be based solely on the network resources consumed by the application under normal conditions. When determining what network resources are needed, factors such as the level of tolerance the application has for reduced network performance if the full amount of resources is not available should be considered. Also, the user may have further requirements, such as maximum cost. It is unlikely that users will be able to specify low level network resource requirements of their applications, such as bandwidth and buffer requirements. A user will most likely want an interface indicating a scale of quality and cost which they can adjust dynamically to meet their needs. So one expects that the QoS given to the traffic management protocol will specify a minimally acceptable standard based on knowledge of quantitative application requirements with input from the user on qualitative requirements. The full specification of QoS may be quite complex and the best method for integrating user requirements with traffic specification to determine the QoS specification is still an area of open research. The exact determination of QoS for applications is, therefore, beyond the scope of this thesis. This thesis instead adopts QoS based on the acceptable traffic behaviour of a flow, and assumes that these requirements are captured by examination of the application (e.g. specifying the actual characteristics of a video stream) or estimations based on typical requirements of such applications. This does not prevent other factors becoming a part of the QoS specification; but it does assume the existence of a mechanism for converting QoS specifications into a description of the traffic flow's behaviour. For example, if the QoS requirements of the user implies an acceptable loss rate of 5%, then the flow's specification of maximum burst size, throughput, etc. is adjusted to reflect this before being passed to the traffic management system. This thesis instead addresses the issue of managing such traffic flows with hard real-time guarantees.

Given the QoS specification in terms of expected traffic behaviour (e.g. maximum delay, loss, etc.) and the specification of the flow's traffic characteristics (e.g. average rate, burst size, etc.), the traffic management system must convert this information into a specification of network resource requirements and then ensure the availability of these resources to provide a connection that satisfies the QoS specification or inform the user that the requested QoS can not be met under the current network conditions. The user can then decide whether to accept a lower standard or try at another time when the network is less busy. Once traffic is accepted at an agreed QoS, it is important that the agreement is met. Both integrated and differentiated service must monitor traffic as it enters the network. Traffic that does not meet agreed the specification is discarded or may not receive the same QoS guarantees. The traffic management protocol should also provide a mechanism to allow new applications to join the traffic mix or to modify the agreed QoS. This involves determining the effect that the addition of the application will have on existing QoS guarantees to other traffic and deciding on a service level which will not violate other QoS agreements or exceed the resources available in the network. The traffic management protocol should also have a mechanism to release resources when traffic flows are terminated.

1.3 Building End-to-End QoS

The traffic management described so far assumes that the network resources between the sender and receiver are entirely under the control of a single management system. Although this is typically true in local area networks (LANs) it is seldom true in larger networks such as WANs. The Internet is not under the control of a single entity and is not even homogeneous. Standard protocols may be implemented in different ways resulting in varying performance over different parts of the network.

The resources available in different parts of the network can vary significantly as well. It is reasonable to conclude that the ability to support QoS will also vary across parts of the WAN. Some parts may provide only differentiated service, others both integrated and differentiated. The final requirement for providing QoS in WANs is the ability to link these various networks with their different capabilities to provide an end-to-end guarantee of the QoS specification.

Building this link requires an agreed protocol for specifying and sharing QoS specifications. Within the Internet, this function can be provided by the Resource ReSerVation Protocol (RSVP) [97]. RSVP is a transport layer protocol used for traffic control. The protocol makes reservations of resources for individual traffic flows by transporting the QoS specification to the various network elements along the path from receiver to sender and refreshing the QoS specification on a regular basis. The refresh allows for dynamic changes in QoS specification and also ensures that flows which are no longer active do not retain resources. This is important given that the alternative, sending a tear down message to remove the reservation, is susceptible to lost messages leaving reservations in place that are no longer in use.

RSVP, Integrated Service and Differentiated Service combine together to form the architecture upon which QoS can be provided. However, they all focus on the specification of interactions between parts of the network and the services which should be provided and not the underlying mechanisms used to determine what resources are required to provide the QoS guarantees and whether these resources are available. This is not an omission on the part of RSVP or the descriptions of integrated and differentiated services. The goal of the working groups involved in these areas has been to provide a general framework for establishing reservations and communicating QoS specifications, not to dictate the manner in which QoS is guaranteed. This thesis addresses this lower level issue of traffic management: converting traffic specifications into network resource requirements and efficiently allocating these limited network resources in WANs . The work in this thesis is presented with reference to the Internet as the working environment; but this is only for the purpose of illustration. The results are not limited to the Internet but do assume the availability of protocols for controlling routing, setting up reservations and providing integrated and differentiated service.

1.4 Contributions of Thesis

Providing network traffic management for real-time applications is challenging. There are many trade-offs that can be made between the complexity and the accuracy of the analysis that must be performed in determining the network resources required by real-time traffic. There are also trade-offs in the complexity of implementation of the network components and the performance that can be offered to real-time traffic. Traffic management models exist in the literature for providing support for real-time flows based on various queuing disciplines. Bounds on the QoS that can be provided for a given set of network resources and traffic are known for most of these queuing disciplines; but many of these bounds are based on analysis that introduces restrictions in traffic flow or network topology.

The aim of this thesis is to examine the existing approaches and to offer a more accurate method of determining bounds while minimising the restrictions on network behaviour. The primary contribution of this thesis is a new analysis of the network performance that can be offered to real-time traffic based on the specification of the network traffic flows and the available network resources. This analysis considers link effects on traffic arrival and provides tighter bounds on QoS than the analysis currently used in the literature. The analysis is applied to derive equations for upper bounds on queuing delay and buffer requirements in FIFO networks without rate controllers which are significantly more accurate than currently published bounds.

Two further contributions of this thesis address the problem of maintaining delay bound guarantees in the presence of network faults. The first contribution is a method for reducing the overhead of the resources needed to guarantee recovery on an alternative path with the same performance guarantees as the original path. The second contribution is an algorithm for choosing the node from which to build a secondary path and a method for accounting for fault recovery time within the delay guarantees of the real-time channels.

1.5 Outline of Thesis

This thesis is divided into three main parts: Chapters 2 and 3 provide the background for the analysis, Chapters 4 through 6 present and evaluate the analysis and provide a framework for providing traffic management based on the analysis. Chapter 7 describes a mechanism for providing resource efficient fault tolerance to guaranteed real-time network traffic. Chapter 2 begins with a detailed discussion of the network model. This first section defines the attributes and behaviour of the network that are assumed to be present by the analysis. Although the Internet is cited as an example of a suitable network in this thesis, the analysis can be applied to any other network architecture which meets the requirements presented in this chapter. The following section evaluates the appropriateness of scaling the techniques used in real-time local area networks (LANs) to wide area networks. This is followed by a classification of the various approaches to servicing real-time traffic flows in WANs with an evaluation of the type of QoS each can provide and the strengths and weaknesses of these different approaches. The final section identifies and discusses the core requirements for providing real-time communication with guaranteed QoS.

Chapter 3 introduces the relationship between traffic arrival, service and the QoS bounds of delay, jitter and packet loss. This chapter surveys various published service disciplines describing the design and QoS bounds offered by each. The disciplines are classified into a taxonomy based on their characteristics. This chapter concludes by identifying assumptions that are made in the work currently published about traffic arrival and service and how these assumptions cause the QoS bounds to be overly pessimistic.

Chapter 4 introduces a method for more accurately determining the arrival and service pattern to reduce the pessimism of the QoS bounds. This chapter also demonstrates how applying these methods can significantly reduce delay bounds in FIFO networks without requiring rate controllers or traffic shapers at each node. Chapter 5 introduces an end-to-end scheme for establishing a real-time traffic flow based on the analysis presented in chapter 4.

Chapter 6 presents an analysis of the performance of the bounds defined in chapter 4 by comparing the analytical results to performance seen in simulation. The results are also compared to bounds published in the literature for FIFO service. The simulations show a significant improvement in the accuracy of the bounds over previously published work.

One of the advantages of the routing schemes used in packet-switched networks is the ability to dynamically adapt to changes in the network topology due to congestion or failure of links. Chapter 7 examines the issues of providing reliable communication to time sensitive applications in the presence of network failures and proposes a method for reducing the overhead caused by the reservation of additional resources for providing redundant data paths.

Chapter 8 presents conclusions of the thesis and future research directions in the area of QoS support in WANs.

Chapter 2

Requirements and Goals for Real-Time Communication in WANs

Any real-time traffic management that *guarantees* a QoS relies on some degree of predictable behaviour within the WAN. To predetermine the QoS that will be experienced by a traffic flow in the network, we must know a priori the path the flow will travel over and the characteristics of this path (e.g. the delays due to transmission, the behaviour of the network elements, and the congestion due to competing flows). This chapter begins with a description of the network model which is assumed by the traffic management system in this thesis and identifies what behaviour needed to support QoS already exists on networks that match this model and what additional constraints on behaviour need to be added.

Based on this model, techniques that are currently used to provide performance bounds in LANs are examined in order to determine which of these techniques might also be used to provide real-time WANs. General techniques for providing differentiated and integrated service to traffic flows in WANs are then introduced and their appropriateness for providing QoS guarantees is evaluated.

Predictable behaviour within the network only provides the groundwork for QoS guarantees. It is still necessary to deal with the management requirements at the ingress and egress points of each section of network that is under the control of a single traffic management system responsible for providing QoS guarantees to that section of network. In addition, the end-to-end management to link these autonomous sections of network together into an end-to-end QoS guaranteed path for the traffic flow must also be implemented. Issues such as collecting flow information, specifying QoS requirements and ensuring that flows adhere to their specifications are presented along with techniques for implementing these parts of the traffic management system. These various techniques are evaluated on the basis of accuracy in capturing traffic flow behaviour and their complexity of implementation. Although these evaluation points do not directly affect the correctness of the methods (i.e. whether the methods are able to provide QoS guarantees), they do affect the cost of implementation and the applicability of the various solutions.

2.1 Network Architecture

The WAN model assumed in this thesis is a collection of network nodes connected by links (figure 2-1).

The internode links are point-to-point, i.e. a single link directly connects only two nodes. The configuration of the network nodes and links (*topology*) can be arbitrary. Each node either behaves as a *host*, which generates or receives application traffic, or a *router*, which forwards network traffic towards its destination. It is possible for a single node to behave both as a host and a router. The term host applies to a node when it is *behaving* as a host and likewise for the term router. The source and destination pairs are assumed to be connected, either directly or through one or more intermediate nodes. Each link represents either a *physical link* or a *virtual link* of known *bandwidth* (the number of bits which can be transmitted per second) and *latency* (length of time for a bit to travel across the link). A virtual link appears as a single link to other nodes; but may actually be a network of nodes and links as shown

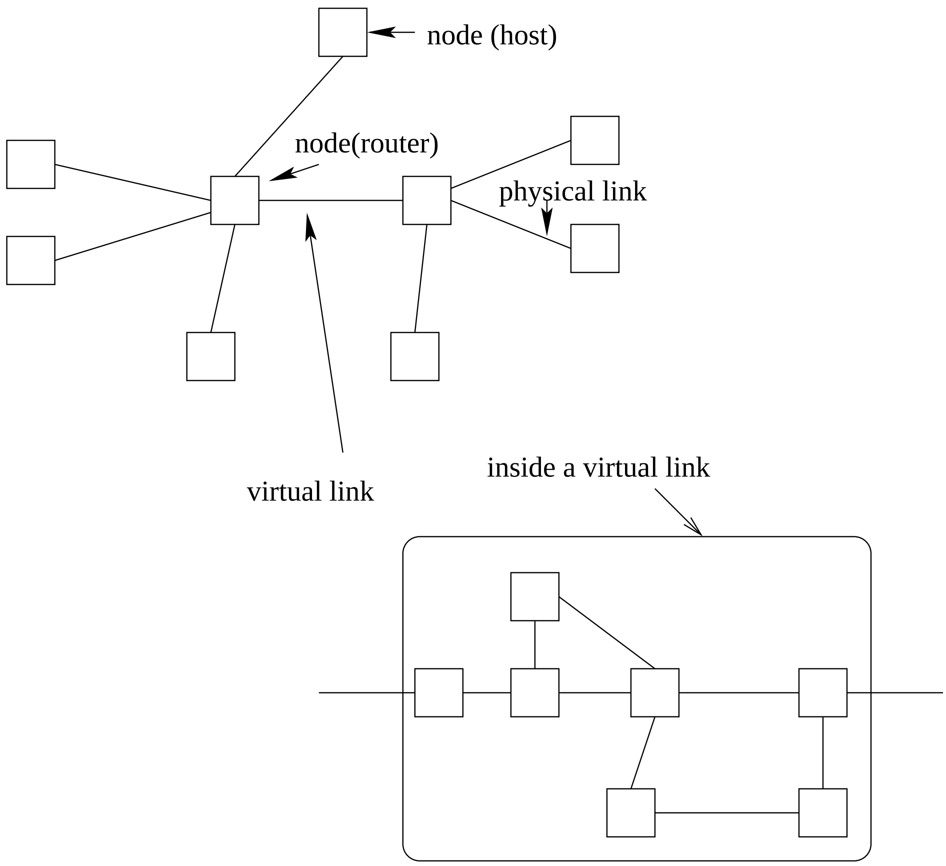

Figure 2-1: Network Architecture

in figure 2-1. For example, a section of network supporting differentiated service that guarantees a given bandwidth and latency can be modeled as a single virtual link of the defined performance. Given that the networks of interest are WANs, single links can span long distances and the source and destination are likely to be separated by several such links.

The network is assumed to be packet switched, i.e. data from a given flow is broken down by the host at the source end into units (*packets*) of a known maximum size before being sent on the network. The maximum packet size depends on the network and each autonomous network can define it's own maximum size. Each packet is sent from the originating host to a connected router. These packets contain the source and destination host addresses. The router receives the packets and forwards them towards their destination according its local routing information. The routers use a *store and* *forward* policy. This means that the router receives the entire packet and then forwards the packet on to the next router towards the destination or to the destination itself if the router is connected directly to the destination. The router nodes are responsible for maintaining information that allows each router to decide which of its output links should be used for forwarding a packet towards its destination. It is assumed that there is a way to influence this routing information to force all of the packets of a particular flow to travel a known path. Within the Internet, *source based routing* allows the source to specify the path the packets will take [77]. Although this is sufficient, it requires the originating host to know what paths through the network are able to provide the required QoS. Instead of assuming the source knows an appropriate route, it would be better for the route to be determined as part of a network protocol for route discovery and establishment.

The network router node model is depicted in figure 2-2.

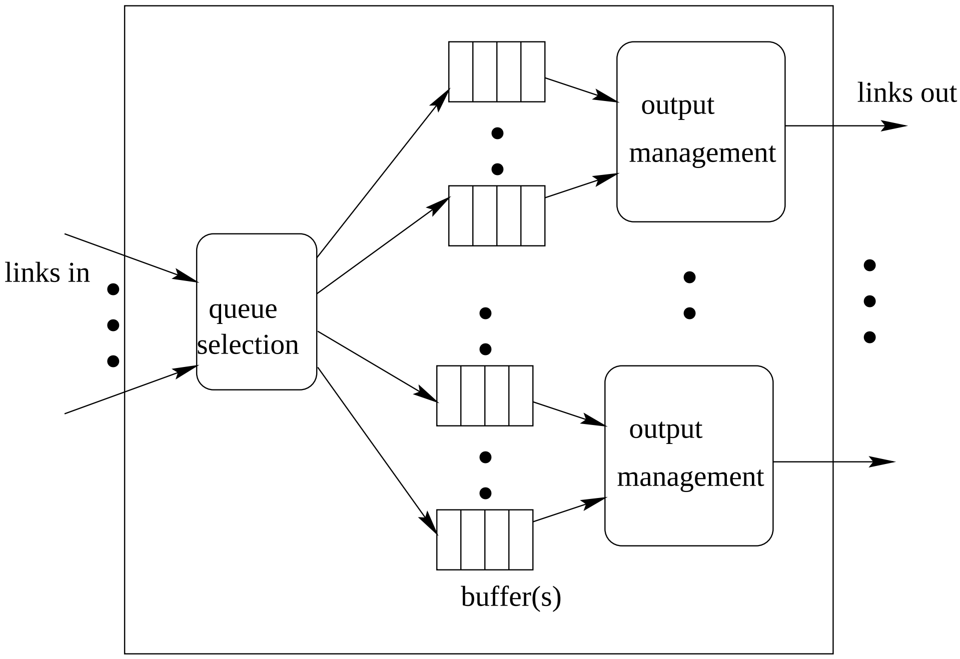

Figure 2-2: Network Router Node Architecture

Packets arrive from other nodes on the input links and are queued in a buffer. This buffer can be an input buffer where the packets are stored as they arrive and then later switched to the appropriate output link, or output buffered, where packets are switched to a buffer associated with a particular output link. Output buffering is shown in the figure; but the work presented in this thesis does not rely on the buffering model. Bufferless routers are rare, but they can be modeled by a router with a buffer set to size zero. A router may have a single buffer or multiple buffers of various sizes used for queuing. The size is specified by the maximum number of bits which can be stored in the buffer. The number of buffers and the order in which packets are placed into buffers is determined by the queuing discipline.[1] Packets contending for a given output link are selected from the buffers for transmission by the output manager whose behaviour is also determined by the queuing discipline. The routers are assumed to be *non-preemptive*, once a packet begins transmission it is not interrupted to send a different packet.

Each router node has one or more input and output links connecting it to other nodes. These links are assumed to be unidirectional although the model can be extended to bidirectional links by including two unidirectional links for each bidirectional link. The unidirectional model allows for the representation of links with unequal bandwidth in the two directions.

The network architecture requirements for providing guaranteed QoS are:

1. a predictable path through the network
2. the static characteristics of this path, e.g. time needed for a packet to traverse a link
3. the dynamic characteristics of the path, e.g. the congestion experienced at each router along the path.

The network model as described so far provides us with partial support for a predictable path through source routing, although the model does not provide us a way of finding a path which can provide the required QoS. This path discovery will need to be added. We can determine static characteristics of the network performance, such

[1] The queuing disciplines are discussed in section 3.1.

as link latency, from the network model. Physical links have defined behaviour and virtual links can not represent networks that have unpredictable behaviour, so their behaviour is also well defined. Currently the network, as modeled, does not provide us with the dynamic characteristics of the path. The network provides no way to determine what other traffic will compete with a given traffic flow. Although we may be able to make predictions based on expected traffic, the network has no mechanism to ensure that our predictions match the actual traffic seen. To guarantee QoS we must add to the existing network model:

1. mechanisms to discover and establish paths through the network capable of providing the required QoS (assuming such paths exist)

2. determine what traffic will compete with a given flow

3. ensure that the predicted competing traffic is indeed what is actually experienced by the flow.

2.2 Scaling Real-Time LAN Techniques

Local Area Networks (LANs) that can provide real-time guarantees have existed since the 1980s. Bounds on the maximum delay between the time a packet is ready to be sent and the time it arrives at its destination can be provided for Token Bus and Token Ring [99] [74] architectures, and Ethernet, with the addition of another protocol layer [81]. It is worth determining what characteristics are shared in LAN and WAN communication and what are their differences. With this information, we can decide which mechanisms used to provide QoS guarantees in LANs can be applied to WANs as well.

Token rings, as their name implies, are formed as a set of hosts connected in a point-to-point manner in a ring topology. Token bus is similar to token ring, except for topology (hosts are connected to a shared bus rather than in a ring). A token is passed through the network and a host may only send data while holding the token.

The time a host may hold the token is limited to the *token holding time* (THT). After this time, the host must pass the token to the next host in the ring. This guarantees that the maximum amount of time that a node must wait before it can send a message is limited by the *token rotation time* (TRT)[2]:

$$TRT \leq Total\_Hosts * THT + transmission\_latency\_of\_ring$$

Unlike token rings, the nodes in a WAN are not configured in a logically sequential order. The number of routers which a packet will pass through can change dynamically based on current network conditions. Also, the hosts which are transmitting through these routers can change dynamically, so the value of $Total\_Hosts$ is not a fixed value in WANs. Even if the hosts transmitting through the routers were known and only transmitted based on possession of a token, the token rotation time (i.e. the time to pass the token from a host to the host logically furthest from it) would be too long to be practical. Token based schemes do not scale to WANs. However, we can draw some similarities between the token rotation time and the congestion delay experienced at a single router output link. Each flow sharing a link consumes a portion of the link's bandwidth resources. The token holding time is similar to the bandwidth resources allocated to each flow in that the allocated bandwidth represents the amount of the link's capacity the flow can make unavailable to other flows, thereby potentially increasing the time to send another flow's data on the shared link. Likewise in the token ring, the token holding time determines the maximum time that a host can delay the sending of another host's data. A single router node can therefore be thought of as having a delay due to competing flows similar to the token ring's delay due to competing hosts. In the router the delay is due to congestion (other flows making use of shared bandwidth) and in the token ring the delay is due to the time needed to acquire the token.

Early Ethernet and current wireless Ethernet use broadcast for communication. It is necessary to establish a protocol for accessing the medium to allow hosts to

[2]The actual behaviour of most token rings is more complex than this; but it is sufficient for the discussion to understand the basis of bounding the TRT.

communicate without interference from other hosts. Ethernet was originally based on the Carrier Sense, Multiple Access with Collision Detection (CSMA/CD) technique. A host on a CSMA/CD network attempts to access the network by sensing if the shared network is currently carrying data. If the network is silent, it begins to send data. In the event that another host also begins sending at this time (or any time before detecting the first host's data) a collision occurs. The hosts detect the collision and stop sending data for a period of time. The amount of time which the host waits is chosen from a distribution such that it is unlikely the two hosts will each wait for the same amount of time. However, it is possible that the hosts' data will again collide or another host may attempt to access the network and cause a collision. Although the probability of repeated collision may be small, the network can not *guarantee* that a host will be able to send its data within a set period of time. The same is true of the wireless access protocol CSMA/CA. The determination of upper bounds presented in [81] involves building a token based protocol in software on top of the underlying Ethernet. This effectively turns the Ethernet into a token bus. To be suitable for carrying hard real-time traffic, a network must be based on a method of network resource sharing which can guarantee traffic access to the network within a bounded period of time. This makes CSMA/CD and CSMA/CA based networks unsuitable for carrying hard real-time traffic without further modification.

Modern Ethernet is typically switched which puts each host in an isolated collision domain. This effectively eliminates collisions and forms point-to-point links between hosts thereby significantly reducing delay. Delays are still not guaranteed as links are host-to-switch and switch-to-host rather than host-to-host. So two hosts both trying to communicate to a third host will have to share the available bandwidth between the switch and third host. The level of link sharing in WAN communication is much higher than in a switched LAN. It is not feasible to support the level of point-to-point connectivity that a switched LAN has in a WAN. However, the principle of reducing delay through isolation of traffic is one that we can adapt to the WAN.

Asynchronous Transfer Mode (ATM) is a technology used in both LANs and WANs [41]. ATM networks use packet switching as the transmission method and transmit fixed sized packets called *cells*. ATM routing is based on *virtual circuits*, which define the path for routers to forward packets. When a host starts an application that will send packets on the network, it first sends a request to establish a path. This request is sent to the router, which then determines the next router towards the destination. This router then becomes the next hop for all packets to be sent on the virtual circuit. In ATM the mechanism for making routing decision for packets on a virtual circuit can be independent of the mechanism for establishing the route for a new virtual circuit. ATM allows the definition of different types of service: constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR) and unspecified bit rate (UBR). CBR, VBR and ABR provide guaranteed QoS while UBR provides best effort service. ATM is capable of providing guaranteed QoS since virtual paths and circuits can be protected from each other by allocating bandwidth specifically for the use of a particular virtual circuit or group of virtual circuits. Several aspects of ATM architecture are indicative of the requirements for guaranteed QoS: fixed path through the network for all packets on a virtual circuit, protection of virtual circuit performance from the effect of other virtual circuits (in the case of ATM, through bandwidth allocation) and a mechanism for setting up virtual circuits given known traffic characteristics. The main disadvantage raised of the ATM architecture is the requirement that all traffic is organised into virtual circuits, including best effort traffic. If a virtual circuit fails, a new one must be established rather than allowing packets to travel on any available link as in the Internet. Ideally the overhead of virtual circuits should only be paid by traffic requiring QoS guarantees but not by best effort traffic.

WANs differ from LANs in two important characteristics that affect real-time data: typically distances between nodes in the network are longer in WANs and the topology tends to be irregular. Shared media LANs, such as original Ethernet and token ring, and switched LANs do not scale to the longer distances and rely on the fixed topology to determine path. Neither approach is suitable for building a real-time

WAN, however we can use the characteristics of fixed path and a priori determination of each host's affect on the shared resources as guidelines for developing real-time WAN mechanisms. ATM is closer to the goal, which is expected since it was designed to work as a WAN architecture as well as a LAN architecture. The virtual circuit model, specification of source behaviour and reservation of resources are all important to providing guaranteed QoS. However, a model which allows non-real-time traffic to avoid the cost of establishing channels would be more flexible and is a goal in this thesis.

2.3 A Survey of Techniques for Real-Time Network Management in WANs

The techniques currently used or proposed for managing real-time traffic in WANs can be classified into three categories. The simplest approach is the *best effort* approach. The best effort approach does not reserve network capacity for traffic flows; but gives real-time traffic some level of priority over non-real-time traffic. This approach is simple since it does not need to make use of traffic flow characteristics or resource reservation. The second approach is *bandwidth* or *"rate-based" reservation.* This approach is the simplest of the reservation approaches. Bandwidth is reserved for a connection based on the delay requirements of the source. This approach can guarantee that the traffic will arrive within a known delay at the destination; but it is unable to give accurate predictions about the variation in delay, known as *jitter.* It also requires reshaping of the traffic flow at each router. The final network management approach is *schedulability analysis.* Schedulability analysis determines whether or not the performance which can be offered at a particular node is sufficient to meet QoS requirements based on the traffic flow characteristics of the incoming traffic, both existing and new, at the node. Schedulability analysis can support simple methods of handling packets at the router without explicit bandwidth reservation; but at the cost of higher complexity in establishing flows. Each of these approaches represents a trade-off between complexity of management, efficiency of network use, and flexibility in resource allocation. The following sections explore these trade-offs in further detail.

2.3.1 Best Effort Approaches

The simplest scheme proposed for handling real-time traffic is to give real-time packets priority over non-real-time traffic to increase the probability of a packet gaining access to an outgoing link [15]. This approach has the advantage of being simple to implement and manage. The basic requirement is just two queues: one for real-time packets and one for non-real-time packets. Packets from the non-real-time queue are sent only when the real-time queue is empty. This will guarantee that real-time traffic will have a smaller delay at switches than non-real-time traffic. If the real-time queue fills, the non-real-time queue is used for overflow, pushing out non-real-time packets when full.

This scheme is very efficient but creates several problems. Non-real time traffic can experience starvation in such a scheme if real-time traffic always takes priority. If the incoming flow of real-time traffic is greater or equal to the available outgoing bandwidth non-real-time packets may stay in the queue indefinitely. Also, with no control over the acceptance of new real-time traffic, a new real-time flow will increase the queuing delay of existing sources without regard to the delay requirements of the existing real-time flows. Variations on this priority idea exist that include sharing of the output link to allow non-real-time traffic at least a minimum level of access to the link. This can be accomplished by guaranteeing a minimum rate for non-priority traffic (e.g. guarantee that one low priority packet is sent for every 100 high priority packets) or by adjusting the sending rate of the priority and non-priority queues based on the relative queue sizes.

Best effort schemes can increase the *throughput* of real-time traffic, i.e. the actual amount of the bandwidth available to real-time traffic. However, they can not guarantee bounds on the delays experienced by or the loss of real-time packets. Unbounded delays are due to uncontrolled local delays caused by congestion at routers along the path. Locally, if the rate of the incoming traffic destined for a particular output link is greater than the bandwidth of that link for a sufficiently long period of time the buffers will overflow and packets of real-time traffic can be lost.

Currently, the standard Internet protocol (IP version 4) implements only best effort approaches, which aim to minimise congestion and loss; but do not avoid it entirely. TCP (the Internet's connection-oriented transport protocol) congestion control is not router based, as described above. Instead, it is host based where the host adjusts the rate at which it sends packets by observing how many packets are successfully being sent over the network. TCP connections define a window which limits the maximum number of unacknowledged packets for a single connection in the network. Once a connection has sent the maximum number of packets allowed by the window size, it must wait for acknowledgment(s) of receipt of the packet(s) from the receiver before sending additional packets. The window size is increased when packets sent are acknowledged and decreased when packets are not acknowledged within a set time out period. In this way the host adjusts it's sending rate based on the network delivery rate [77]. In addition to congestion control mechanisms which react to congestion in the network when it occurs, congestion avoidance mechanisms exist which try to prevent congestion from occurring. Congestion avoidance mechanisms such as *Random Early Detection* (RED) [27] and the window sizing mechanisms in *TCP-Vegas* [66] exist in the Internet. RED routers drop packets randomly at a specified probability whenever the congestion reaches a threshold. The goal is to get the host to reduce its sending rate before the congestion is high enough to cause the router's buffers to overflow. When the host does not receive an acknowledgement of the dropped packet, it will reduce its window size and therefore its sending rate. TCP-Vegas compares the measured throughput seen to an expected throughput ($Expected\_Throughput = Window\_Size/Round\_Trip\_Time\_with\_No\_Congestion$). The source decreases or increases the window size if the difference between the actual and expected throughput exceeds or falls below a given threshold. Although all of these mechanisms reduce the congestion at routers within the network, none of them can *guarantee* no loss of packets will be experienced by a particular flow.

Although TCP, through acknowledgments and resends of unacknowledged packets, can give a high probability that a packet will eventually arrive at its destination, and through congestion control and avoidance mechanisms can improve overall network performance, it is still not sufficient to guarantee bounds on delay. Real-time applications are unlikely to tolerate end-to-end delays that are long enough to allow for resends or to be flexible enough to have their source rates adjusted in response to dynamic changes in network congestion. There is always a chance of a mismatch between the source's rate and the rate that can be offered through the TCP based network. The fact that the reliability and congestion mechanisms of TCP are ineffective for real-time applications is part of the reason for the more common use of the *User Datagram Protocol* (UDP) for real-time Internet traffic. UDP is a connectionless protocol. There is no information in packets relating them to one another, therefore there is no inherent ordering of UDP packets. UDP does not provide reliable delivery of packets and does not employ congestion control mechanisms. Since UDP does not have to provide any of this functionality, which is of little benefit to real-time applications, packets can be sent with lower computational overhead and without the sending rate being limited by the congestion control mechanism. UDP, however, does not guarantee delivery of packets at all much less offer guarantees on maximum or minimum packet delays. Although best effort approaches may be suitable for supporting a low cost, variable performance option for non-critical real-time applications, it is unlikely to be flexible enough to support the range of QoS requirements desired for future real-time applications. As the goal of this thesis is to provide guaranteed QoS for real-time applications, best effort approaches are not suitable. They may, however, be used to support non-real-time flows in conjunction with separate methods for supporting real-time flows. To allow non-real-time traffic to be handled simply and to take advantage of dynamic routing based on network congestion it is assumed that non-real-time traffic will be handled by a best effort mechanism separate from the real-time traffic. Since the focus of this thesis is on guaranteed QoS, best effort approaches are not discussed further. The functionality missing from best effort approaches is the ability to *reserve* resources required by a traffic flow at the routers in order to ensure that the flow's packets are not affected by dynamic changes in congestion. The next two sections explore approaches for making such reservations.

2.3.2 Bandwidth Reservation Approaches

Bandwidth reservation based schemes provide QoS in a manner similar to the telephone circuit switched system, which reserves bandwidth for each connection with sufficient capacity to handle speech. Unlike a circuit switched system, these schemes are designed to work on packet switched networks and to allow other traffic to make use of the reserved resources when they are idle. Bandwidth reservations schemes reserve sufficient bandwidth and buffer space on the routers along the flow's path to meet the QoS requirements. When a new flow is established, it specifies the bandwidth it requires to ensure its packets will reach their destination within a bounded delay time.[3] The flow also specifies the maximum size of a burst of packets it may send and the router allocates sufficient buffer space to accommodate this burst. With these resources set aside for the use of the flow and the flow adhering to the agreed behaviour, the QoS would be met if not for the effect of other flows.

When a flow's packets enter a router that is not busy, they can be forwarded immediately. However, if other flows are competing for the same link the packets may be delayed. As long as adequate bandwidth has been reserved, the flow is assured that they will not be delayed too long (i.e. they will meet the delay bounds that the bandwidth reservation is based on). The buffer reservation assures that they will not be dropped while waiting due to a buffer overflow. However, because the packets may be treated differently (forwarded immediately or delayed for some time up to the agreed bound) depending on the presence of other flows, the flow's behaviour can begin to change. This distortion of behaviour is shown in figure 2-3.

As flow one's packets enter router 1, no other flows are using the router and the traffic is forwarded as it is received. The spacing between the packets (the rate)

[3]The relation between delay and bandwidth allocated is defined in Chapter 3.

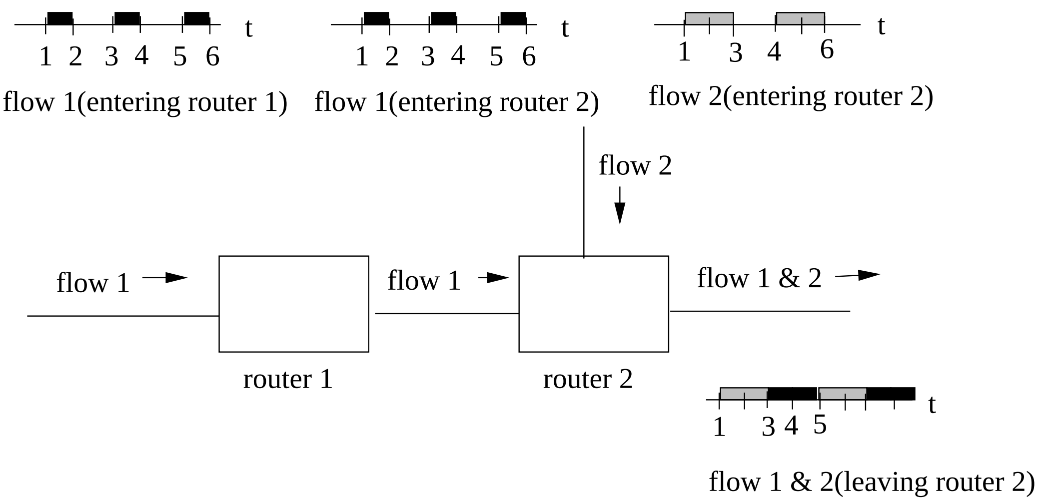

Figure 2-3: Distortion of a Flow

remains the same. At router 2, another flow also has bandwidth reserved at the router and packets from the first flow are delayed while the packet from the other flow is sent. Then, because there are no other packets queued, the next packet from the first flow is sent immediately. On leaving router 2, the first flow's first two packets are now closer together effectively creating a small burst which was not due to the flow's own characteristics but due to cross traffic. Since the presence or absence of cross traffic was not a consideration in the allocation of resources at subsequent routers and flow one's traffic no longer adheres to its original shape, the resources at subsequent routers may be insufficient to provide the promised delay and loss characteristics. The distortion may cause buffers to overflow and/or the delay of packets to increase due to the potentially higher than agreed burst rate, even though the average rate has not changed.

To solve this problem, bandwidth reservation methods use *shaping* at each router. The shape of the traffic does not have to be the same as the original flow and does not have to be the same at each router; but the resource reservations made to guarantee the delay bounds and buffering are based on this new traffic shape rather than the original shape. In the example, the traffic could be reshaped by delaying the second packet to reestablish the original rate.

In [60] [61], bandwidth reservation methods have been shown to provide delay guarantees, but there are some disadvantages. No one has yet discovered a way of providing a lower bound on delay. Therefore, the minimum delay at a router is zero and the jitter (delay variation) bound is the same as the delay bound. For applications sensitive to jitter (video), this may not be sufficient. In addition, the reshaping of flows at the routers throttles the flow. Even if the next router is free to send packets, the shaper may have to hold the packet so the packets do not arrive at the next router. This forced idleness may not make the most efficient use of the network resources.

2.3.3 Schedulability Analysis

A second approach to ensuring adequate resources remain available to guarantee QoS is *schedulability analysis*. Schedulability analysis determines the point of maximum backlog given a set of flows and compares the time to clear that maximum backlog with the required delay bound. If the time to clear the maximum backlog is less than the delay bound and the maximum backlog does not exceed the available buffering, then the set of flows is schedulable. For a given time t, the *backlog* is the maximum number of packets that could have arrived from all flows minus the number of packets sent by the output link at that time.

This differs from bandwidth reservation in that a proportion of the bandwidth and buffering is not explicitly reserved for a particular flow and therefore, the bounds on delay are not based on this allocation of bandwidth, as they are in bandwidth reservation schemes. Instead, schedulability analysis determines what performance bounds can be offered by a router based on the traffic characteristics of flows using that router. When determining bounds which can be offered by the router, schedulability analysis determines the effect the new traffic will have on the performance of all guaranteed flows at the router. If the performance at the router given the new traffic would fail to satisfy the requirements of one or more flows which have already been guaranteed, the new flow is rejected.

The disadvantage of such a scheme is the increased computational complexity in determining bounds. It is easier to determine the bandwidth required to offer a specified delay and then check to see if this bandwidth is available than it is to determine the delay bounds based on the potentially large number of traffic flows. However, it can be beneficial to decouple the delay guarantees from the allocation of bandwidth. This can allow a flow with low bandwidth requirements that requires a small delay to be guaranteed the delay without allocating more bandwidth than the flow actually requires. The complexity in calculating performance bounds for schedulability analysis occurs at channel establishment time, prior to the sending of data. Since the performance cost is paid before the real-time flow is sending data, this higher complexity may be worth the potentially higher network utilisation. Both schedulability analysis and bandwidth reservation can provide guaranteed QoS bounds for real-time applications. Both methods are explored further in Chapter 3 through a comparison of the performance bounds offered by several proposed router queuing models which make use of either schedulability or bandwidth reservation. New formulas for providing tighter QoS bounds to schedulability based reservations are given in Chapter 4

Having identified the need to provide a mechanism for determining whether the available resources are sufficient to meet the specified performance bounds of a set of guaranteed flows and identified two general approaches to providing this mechanism, the next section defines what additional information and support are needed for the management of real-time flows. These requirements are shared by both bandwidth reservation based and schedulability based methods, and the section explains how traffic management based on each of the two techniques addresses these requirements.

2.4 End-to-End Requirements for Real-Time Network Management of QoS Guarantees

Real-Time Network Management tasks can be split into those that require only local, router, information and those that require information about the entire path of the flow (end-to-end). The previous section discussed the tasks performed by the router based on local information (i.e. router resource reservation). This section presents the end-to-end tasks, which include determining a route for the connection, specifying QoS requirements, splitting the end-to-end QoS bounds into bounds for each node, determining the source traffic specification and ensuring compliance with the specification.

2.4.1 Specification of per Flow Performance Requirements

QoS requirements differ depending on the application. There are, however, three parameters that define most network application requirements: *delay*, *jitter* and *packet loss*. Delay is defined as the time between the sending of the first bit of the packet from the source and the receipt of the last bit of the packet by the destination. The delay affects the quality of real-time applications and in interactive applications excess delay can cause difficulty for the participants in synchronising their interaction with each other. Real-time network management should therefore support the specification of an upper bound on the acceptable delay and guarantee this bound throughout the life of the flow (or until renegotiated).

Several components make up the delay: propagation, transmission, queuing and switching. The *propagation delay* is the delay between sending of the first bit on a link from the sending host and receipt of the last bit at the receiving host. In a physical link this is the delay due to the distance which must be traveled and the time it takes for the signal to travel that distance in the given medium. The minimum possible propagation delay is defined as:

$$propagation = distance/speed\_of\_light\_in\_medium$$

In a virtual link, this delay may be affected by several factors within the network represented by the virtual link. However, an upper bound on the delay through the virtual link must be given. The propagation delay is constant across a link whether physical or virtual.

At each router, the packet must be switched to the appropriate outgoing link and transmitted on the link to the next node. The delays involved in this process are the *switching delay* and the *transmission delay*. The switching delay is dependent on the switch architecture and is small compared to the other delays [20]. The transmission delay is dependent on the bandwidth of the link and is defined as:

$$transmission = bits\_to\_send/link\_bandwidth$$

Propagation, transmission and switching delay are experienced by packets even in the absence of other traffic. These delays are determined only by physical properties of the nodes and links that the flow traverses not by any dynamic network characteristics such as congestion. In addition to these constant delays, packets experience a *queuing delay* due to contention with other traffic in the network. The value of the queuing delay is dependent on the method for selecting packets for sending at a router (queuing model) and the amount of traffic competing for the same link. The queuing delay of a flow is equal to the sum of the queuing delays at each router along the flow's paths.

The delay bound that can be offered to a flow for each hop in the end-to-end path is calculated as:

$$delay = propagation + switching + transmission + queuing$$

The end-to-end delay consists of the sum of these per hop delays. Propagation and transmission delay are constant for a given packet size and a given path through the network so these can be determined a priori. Although switching delay can vary, the delay is very small in comparison with the other delays and is therefore typically ignored, or alternatively a pessimistic upper bound can be used with little effect on the end-to-end delay. Determining bounds on queuing delay is therefore the challenge for providing end-to-end delay bounds.

Jitter is the variation in delay which is experienced by packets belonging to the same flow. It is the difference between the maximum delay and the minimum delay. Jitter causes higher instantaneous rates, as packets arrive at a faster rate than their specified average time. If the variation is large, the receiver will need to have sufficient buffering to hold packets which are arriving at a rate faster than they can be used by the application. The received traffic can become bursty within the network, as shown in figure 2-3; and if sufficient buffering is not available, packets will be dropped. Applications such as video are sensitive to jitter and therefore the network management should ideally allow the specification of bounds on jitter. Since the minimum delay due to queuing can be zero and the maximum is the queuing delay, jitter has an upper bound of the value of queuing delay. If bounded jitter is not required by the application, then the jitter requirement, by default, is taken to be equal to the queuing delay.

In addition to packet loss at the receiver due to excessive jitter, packets can be lost within the network if sufficient buffering does not exist in the nodes to hold packets during bursts and delays. Both video and voice applications are sensitive to packet loss. The amount of tolerable packet loss is often dependent not only on the application; but also on the quality desired by the viewer. A viewer may be willing to accept lower quality due to packet loss for a video conference; but may be less willing to accept the same quality while watching a movie from a video on demand system or while recording a "live" session.

Delay, jitter and packet loss bounds are the parameters for specifying QoS in this thesis. There are likely to be additional specification parameters such as quality and cost [76]. A trade-off between the desired QoS and the cost of the connection is likely to be a factor in determining the final QoS guarantees; but these additional parameters are beyond the scope of this thesis.

2.4.2 Flow Traffic Specification

Since the queuing delay is dependent on the flows competing for a link within a router, we must be able to characterize the flow accurately. Different real-time applications have distinct source characteristics which may be affected by compression and generation techniques. The model chosen must be flexible enough to capture the characteristics that can be used to compute the QoS bounds possible for these different flows. Network traffic sources can be broadly classified into three types: *periodic*, *sporadic* and *aperiodic*. Periodic traffic sources generate set size packets at a regular rate. This class of traffic is also called constant bit rate (CBR) traffic and is common to many sampling or sensor applications which do not use compression. Sporadic sources generate packets at a set maximum rate but may vary their rate up to this maximum. This class of traffic is also called variable bit rate (VBR) traffic and is common to video signals that have undergone compressions and voice traffic. Various VBR traffic flows can differ in their characteristics. For example, voice traffic is characterized by active periods of fixed size packets followed by periods of silence (on-off source). Compressed video does not show silent periods; but packets are of variable size. Aperiodic traffic sources generate packets at varying rates with no defined maximum or minimum rate. Since the rate can not be defined, aperiodic traffic can not be offered guarantees on delay and will not be considered. Flow specifications should be able to accommodate both periodic and sporadic traffic.

Two models are widely used in the literature for specifying real-time flows. The (σ, ρ) model [18][60] represents a flow's traffic by its average rate ρ and maximum burst size, σ. The second model is defined as a tuple $(x_{min}, x_{avg}, I, S_{max})$ that includes the minimum time between packet arrivals, x_{min}, the average inter-arrival time between packets, x_{avg}, the interval over which this average is taken, I, and the maximum packet size, S_{max}. This model was introduced in [23], and is an extension of [18][19]. We will refer to these models as the rate-based and packet-based models respectively.

Simple periodic sources, which generate one packet every τ time units, can be accurately represented by either model. With the rate based model, a periodic flow with packet size S and rate S/τ can be represented as: $(S, S/\tau)$. The flow sends at a constant rate of S bits every τ time units, and the peak and average rate are therefore equal. Since the rate will always be the same, there is never a burst larger than the one packet. In the packet based model, the flow can be represented as (τ, τ, τ, S). The minimum and average time between packets is τ. The pattern repeats itself every τ time units, so this is a sensible value to measure the average over. For simple periodic traffic, the values of S and τ are sufficient and the extra values in the packet based model add no additional information.

The extra terms in the packet based model represent how consistent a flow is in its packet rate (i.e. how smooth or bursty the flow is). Simple periodic sources are perfectly consistent so no differences are apparent between packet and rate based models and the value of x_{min}, x_{ave} and I appear redundant. However, when the periodic flow can create bursts of more than one packet, the values of x_{min}, x_{ave} and I are used to capture this behaviour. For example, consider a periodic source which sends bursts of n packets of S bits each every τ time units. In the rate based model this flow can be represented as $(nS, nS/\tau)$. In the packet based model this flow could be represented as $(S/r_l, \tau/n, \tau, S)$ where r_l is the transmission rate of the incoming link. Since n packets are generated starting at each interval they will arrive on the link one after the other as the link becomes available. Therefore the minimum time between the arrival of the n packets is the time it takes to receive one packet from the link (i.e. S/r_l). The pattern repeats every τ time units, which is the lowest bound on the averaging interval. The number of packets sent over this interval is n therefore the average time between packets is τ/n. Again the models are very similar to the simple periodic source with the exception of a variation between the values of x_{ave} and x_{min} in the packet model, which captures the multiple packet burst behaviour. This is also captured by the rate based model since the burst size is equal to the average rate over time τ, which implies all of the nS packets in an interval of τ can be sent as a single burst.

The representation of sporadic sources shows the advantages of the additional terms in the packet based model. For comparison assume a video flow has a sporadic behaviour with a packet size of S bits, an average rate of nS/τ bits/s and a peak rate of $2nS/\tau$ bits/s For simplicity we will represent the average rate as r and the peak rate as $2r$. For the rate based model, we also need to know how long this burst can last. Call this max burst length b secs. This flow would then be represented in the rate based model as $(2r * b, r)$. In the packet based model we also need to know the averaging interval I. In the packet based model this flow would be represented as $(S/2r, S/r, I, S)$.

The two models are related to each other.[4] The maximum burst size is related as follows:

$$\sigma = I/x_{ave} * S$$

The maximum number of packets that can be sent in any interval, I, is I/x_{ave} and the number of bits in each packet is S. Similarly, the average rate is related as:

$$\rho = 1/x_{ave} * S$$

From the first equation, one can see that many flows with different packet based specifications can map to the same rate based specification. It is possible for flows with different values of I to map to the same value of σ, depending on the value of x_{ave}. Different I values indicate different frequencies of burstiness but this is not captured in the (σ, ρ) model.

The worst case volume of traffic from a flow j at time t is determined in the rate based model by [60]:

$$V(t) = \sigma_j + \rho_j t$$

This is obviously pessimistic as it assumes the existence of a burst at time zero. For the packet based model the worst case volume from flow j at time t is [91]:

$$V(t) = \lfloor \frac{t}{I_j} \rfloor \frac{I_j S_j}{x_{ave,j}} + \min\{\lceil (\frac{t}{I_j} - \lfloor \frac{t}{I_j} \rfloor) \frac{I_j}{x_{min,j}} \rceil, \frac{I_j}{x_{ave,j}}\} S_j$$

[4]This relation is straightforward; but has not appeared in the literature to the best of my knowledge.

The specification of traffic volume is obviously simpler in computational terms in the rate based model. However, substituting the equivalent rate based terms into the packet based equation for volume gives:

$$V(t) = \lfloor \frac{t}{I_j} \rfloor \rho + \min\{\lceil (\frac{t}{I_j} - \lfloor \frac{t}{I_j} \rfloor) \frac{I_j}{x_{min,j}} \rceil * S_j, \sigma\}$$

This model of traffic volume from the packet based model is more accurate than the one based solely on the values of (σ, ρ), due to the many to one mapping for different values of I. This increased accuracy results in a potentially smaller volume of traffic calculated for the same flow. The new formulas introduced in Chapter 4 use the packet based model to take advantage of this greater accuracy in determining traffic volumes.

2.4.3 Mapping Applications to Flow Specification

Neither of the flow specification models discussed in Section 2.4.2 model all flow behaviours precisely. It is possible to model constant bit rate, i.e. simple periodic, flows precisely as such flows produce exactly one packet every x_{min} time units in the packet based model and S/ρ time units in the rate based model. The models are less accurate with variable rate, aperiodic, flows. Variable rate flows, such as video, exhibit long-range dependent behaviour [22][10][65] and the models do not attempt to capture the actual varying rates during any time interval. For example, consider the packets of two aperiodic traffic flows shown in figure 2-4. The black sections represent when the flow is sending data on the network.

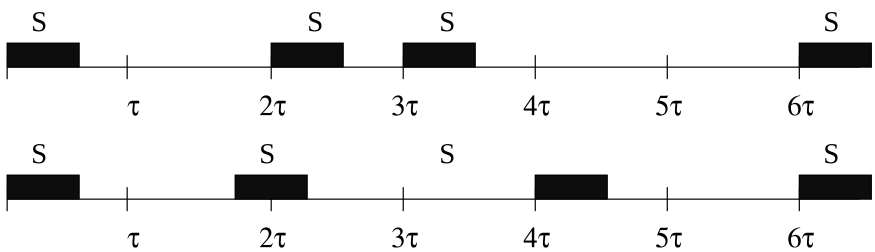

Figure 2-4: Unique Aperiodic Flows with Identical Specification

Both of the flows meet the packet based specification $(\tau, 2\tau, 6\tau, S)$: the average time between packets over interval 6τ is 2τ and packets have a minimum spacing of τ. However, the actual flows themselves are not identical.

Accurately characterising a variable rate flow is likely to require a large number of parameters making computation of bounds based on the parameters intractable. Enforcing such a flow would also be difficult [46]. Despite capturing periodic traffic behaviour, the specification models are not actually designed to model the flow but rather define bounds on the behaviour of the flow (average rate, peak rate, maximum burst). It is tractable to determine these bounds on the behaviour of an application and to measure a flow's compliance with these bounds. Further research into flow modeling may allow us to derive a more complex and more accurate models of the actual flow behaviour in the future. A more accurate model would allow greater accuracy in determining queuing delays, but is likely to bring about a greater computational cost.

For "live" applications that are not constant bit rate, it is not easy to characterise a source in advance. For example, a video conference on a packet switched network which is using compression may vary significantly in rate and packet size. Until the conference occurs, it is not possible to predict exactly what behaviour will be seen. Two options can be taken to handle such situations:

1. Choose a pessimistic model with burst and rate bounds greater than expected for such applications, which guarantees performance.

2. Adjust resources dynamically to meet the required QoS.

The first approach is likely to be wasteful of resources, but is the only way to guarantee QoS can be met. However, few "live" applications have such strict requirements. A dynamic system that allows the user to adjust the perceived performance during a real-time session and adjusts the reservations to meet this performance dynamically is likely to provide the flexibility needed. Such a system requires mapping of perceived performance onto QoS parameters and dynamic update of reservations within the network. None of the proposed methods for providing QoS guarantees directly address the dynamic modification of reservations. Some work has explored dynamic QoS, but it is still an open area of research [63] [12].

Both the packet based and rate based model are sufficient to capture the bounds on source behaviour of real-time traffic which is known a priori and to allow calculation of delay, jitter and packet loss bounds. However both are likely to be a pessimistic representation of the flow and therefore the bounds. The (σ, ρ) model results in computationally simpler calculations of traffic volumes but is less accurate than the source based model. Therefore, these calculated volumes will be more pessimistic (i.e. larger than those calculated by the packet based volume equation). Given that both models are just upper bounds on behaviour, they are both likely to give more pessimistic traffic volumes than the actual sources. However, they offer a tractable way of modeling the flows with a sufficient accuracy.

2.4.4 Routing

Routing is the mechanism for determining the path that packets in a flow take through the network. It includes two tasks: route discovery and packet forwarding. Route discovery involves finding path(s) through the network between a source and destination host pair. This information is then used to build tables at each router which specify the output link to use for forwarding packets to a given destination. To guarantee QoS for a real-time flow, the packets of the flow must follow the path on which the guarantee is based. Because of this requirement for predictable routing, the earliest work on real-time support within networks was for networks with regular topologies [74][1][56][40][99][14][78][3].

To discover routes through a network, information about the connectivity and cost of paths needs to be distributed to the various nodes. This can be done by existing link state protocols, such as OSPF [57], or distance vector protocols, such as RIP [42]. Both of these protocols share global link cost information between routers in order to build local routing tables. Similar protocols could be used to discover routes that are likely to be able to support QoS guarantees of a flow by using delay, jitter and loss characteristics in the cost of links.

Route discovery is a dynamic process since the costs of paths in terms of delay, jitter and loss properties continually change as the flows are set up or torn down in the network. However, it is unlikely to be beneficial to move a flow from a path on which it has a guarantee to another path even if a lower cost path becomes available. Other real-time flows may in the meantime set up reservations on the lower cost path that preclude the moving flow from receiving a guarantee. Even if it did succeed, it would merely be getting the same guarantee that it already had. Therefore, it is assumed packet forwarding on a given flow is static: all packets of the flow follow the same path.

Static packet forwarding does come at the cost. Allowing packets to travel on different routes allows the network to adapt to dynamic congestion. This can make more efficient use of the network resources since each packet can be routed along the path in the network which is optimal[5] at any given time. It is worth exploring protocols for renegotiating guaranteed paths in order to balance load between lightly loaded and heavily loaded links. This dynamic path management is, however, beyond the scope of this thesis.

Dynamic forwarding also adds a degree of fault-tolerance, which is lost if a flow must always follow a set path. When a link within the network fails in a dynamic router, the packet can be routed around the fault using an alternative route. With a set path, the path becomes unusable and guarantees are no longer valid. It is, however, possible to provide fault tolerant flows with guaranteed QoS without the need to reserve a second redundant path. The problem of fault tolerance will be returned to in Chapter 7.

For the purposes of this thesis, a route discovery protocol, which finds routes that are likely to be able to support QoS guarantees, is assumed to exist. An optimal determination of routes is known to be NP-hard, but efficient sub-optimal methods for determining routes likely to meet QoS specification exist [59][54][38]. This coupled

[5]The definition of optimal may vary but often is defined as the shortest or least loaded path.

with mechanisms for "advertising" the quality offered on such routes as in [75] provides a way of finding out about potential routes for flows. One or more routes may be advertised and it is up to the *connection admission control* (CAC) algorithm, discussed in the next section, to establish the flow and its QoS guarantees, if possible, on one of these routes. Such a fixed route concept exists in ATM under the name of *virtual circuits*, due to ATM networks layering of circuit switched properties on top of packet switched networks. The Tenet group uses the term *real-time channel* to define a known route which packets from a real-time connection travel [24]. In the RSVP protocol the term *flow* is used to define a set of related packets following a set path [97]. These terms are used interchangeably, but flow will be used primarily.

To distinguish packets that must be routed along the same path from those which may be routed dynamically, packets carry in their header an identification of the circuit, flow or channel they belong to. It is based on this identifier, and not the destination, that packet forwarding decisions are made. Non-real-time traffic would not contain this identifier or would have it set to indicate that the packets can be routed based on network congestion, or other schemes aimed at overall network efficiency. Although the discussion so far has assumed a single flow as the model on which routing is based, it is worth noting that the routing could also be based on a collection of flows (aggregated traffic) as would be expected for differentiated service networks. The identifier determines the routing and servicing of the packets and any group of packets which require the same service and routing can meaningfully share the same identifier.

Determining a suitable route for a real-time connection is an area of active research. Shorter paths are more likely to meet delay requirements; but congestion must also be considered. In the case of real-time connections, paths congested with other real-time traffic are less likely to accept new connections than those congested with non-real-time traffic. So the goals of overall network efficiency and real-time bounds may conflict in determining routes. Some work has been done on finding paths to meet QoS requirements [55] [72], but determining a routing algorithm which best manages these conflicting demands continues to be a challenge for further research. The existence of such a mechanism, however, is an assumption of the remaining chapters.

2.4.5 Connection Admission Control

In addition to the specification of QoS requirements, flow behaviour and a mechanism for finding potential routes in place, the network still needs a signaling protocol for sending the QoS and flow specifications along the potential routes and determining if any of the proposed paths can provide the requested QoS. It is necessary to have this end-to-end protocol as an application's QoS requirements are between the source and destination hosts. An application is not interested in the performance of particular routers, only in the end-to-end performance. However congestion control and resource allocation, which contribute to the end-to-end queuing delay and packet loss, occur on a per node basis. So a protocol for translating the end-to-end requirements into per node requirements is needed. The breaking down of the end-to-end requirements into per node requirements and determining if the performance of the nodes in the path can collectively provide the required QoS is called *connection admission control* (CAC).

An obvious approach to dividing up the delay, jitter and loss bounds would be to divide them evenly over the nodes. In a path with n nodes, each node would be allowed to contribute up to $1/n$ of the end-to-end delay, jitter and loss bounds. Such an approach, while simple to implement, is not adaptive to varying network conditions and may reject a real-time connections due to relative lack of resources at a single congested node even if the sum of the bounds that could be offered by the individual nodes is sufficient to meet the QoS bounds. Although some nodes may have higher delays, other nodes may have small delays and the combination of high and low delays may be sufficient to meet the end-to-end requirements.

It is also not sufficient to determine the minimum bounds a router can offer and then use this value to determine if the flow can be accepted. Any new flows established through a router will increase the minimum queuing delay of that router. Therefore, once a flow is accepted at a router, if the flow assumes that the current minimum delay will be available, no other flows can be established on that router as they will increase the delay, thereby invalidating the delay assumed by the existing flows. To allow new channels to be established a scheme is needed which only reserves enough resources to guarantee the *maximum* delay which a flow can tolerate at a router rather than the *minimum* delay which the router can provide.

Issues in and methods for establishing end-to-end flows based on guaranteeing the maximum delay bounds have been presented in [97] [23] [75]. The establishment method presented in [23] is specific to the traffic sources used by real-time channels. RSVP [97] is a general signalling protocol. In both documents, the request to set up a flow is sent along the path and each router determines what bound on delay and packet loss can be guaranteed. If the bound at that router exceeds the QoS specification then the connection is rejected all reservations along this path are removed. The reservation request is made from destination to source. The reservation is receiver initiated since the receiver has access to information about the quality of delivery the user expects. The sender may or may not have information about the capabilities of the receiver. By allowing the receiver to specify the required performance, the receiver's hardware/software and the user's requirements can be taken into consideration rather than just the specification of the flow. This approach is also more scalable for multicast applications. As multiple receivers request connections towards the source, their requests can be combined into a single request at joins (figure 2-5).

Receiver initiated reservations was first proposed in [97]; but the design in [23] did not preclude receiver initiated reservations and this was later adopted. The adoption of *soft-state* was also proposed in [97]. With soft-state, information about reservations times out and new reservation messages must be sent periodically to maintain the reservations. Explicit teardown messages were proposed for [23]. Soft-state adds overhead to the protocol since state must be refreshed, but does allow a mechanism for removing reservations if a setup or teardown message is lost.

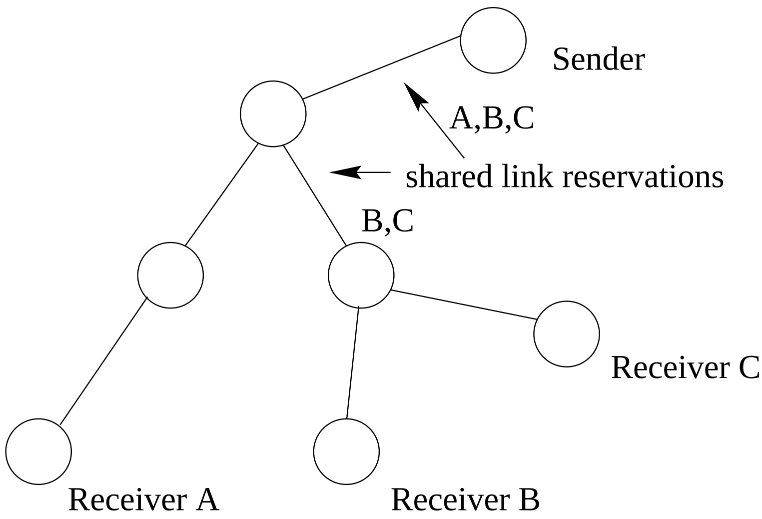

Figure 2-5: Link Sharing in Receiver Initiated Reservations

RSVP and real-time channels differ slightly in their approach on how the reservations are made. In the real-time channels model, the request passes upstream through each router and the router determines what QoS bounds *can* be offered to the connection at that particular node. The reservation request maintains a sum of these bounds. If the sum exceeds an end-to-end bound, then the channel is rejected and all reservations for the connection are removed. At the final node, the difference between the offered bounds and the required end-to-end bounds is determined and this *slack* in the requirements is distributed back to the routers on the path allowing them to relax the bounds that were initially reserved for the flow.[6] RSVP defines a one-pass approach in which the receiver specifies per hop requirements and these requirements are either reserved or rejected at each hop. However this can be extended to one hop with advertising (OPWA) [75] to allow the minimum delay that can be provided by the path to be advertised to the receiver before the reservation is made. The ability to specify slack in the reservation request is included in RSVP. Real-time channels can therefore be implemented within the framework of RSVP signalling. Chapter 5 describes a CAC algorithm that works with the new analysis introduced in Chapter 4. This CAC can also be integrated into RSVP.

[6] Methods for distributing slack are discussed in chapter 5.

2.4.6 Traffic Policing and Shaping

The correctness of the worst case limits on delay and loss relies on the traffic characteristics of all flows adhering to the original specifications that were used to make the resource reservations. If the sources begin sending at a higher rate or if a flow has a higher maximum rate within the network due to distortion, the calculated delay and loss bounds may become invalid. To ensure that a flow adheres to its agreed traffic characteristics, *traffic policing* and *traffic shaping* are used.

Traffic policing ensures that a flow meets its specification at the source by monitoring the flow as it enters the network and discarding or marking any packets which do not conform to the agreed traffic specification. Marked packets may be discarded if congestion is encountered and receive lesser or no guarantees.

Even if a source is well behaved, the traffic from a particular flow may not comply with the flow specification agreed at the edge of the network. Varying congestion at the routers can cause distortion of the traffic flow, creating bursts. The routers must either reconstruct the traffic to a given traffic specification (traffic shaping) or must account for the distortion in determining the QoS bounds.

In the packet based and rate based flow specification models, the reshaping is done based on the rate information. Traffic shaping can occur at both the network entry point and at individual routers in order to modify the traffic characteristics. This can be useful to reform traffic to conform with its original characteristics if affected by distortion due to varying congestion. If a packet arrives too soon after the previous packet, causing the rate from the connection to be higher than expected, the packet is held in a buffer until enough time has elapsed to conform to the specified rate. Networks that reshape the flow based on rate specifications are said to implement *rate control*. Such schemes are also called *non-work conserving* since they may hold a packet in order to reshape the traffic even if the required output link is idle.

Rate control can be implemented in many ways with the implementation chosen affecting the delay and loss bounds that can be guaranteed. One model commonly used in the literature is the leaky bucket regulator [80]. The rate based specification model can be shaped with a bucket size of σ and a leak rate of ρ. Packets arriving at a node are queued in the bucket and leaked from the bucket at rate ρ. If the bucket is full, (i.e. more than σ bits are queued), packets will be dropped. This is used in schemes such as (σ, ρ)-regulator [18][19] and generalised processor sharing [60][61].

Rate control is also used with packet based models in [5][90][88]. Although, leaky bucket regulators could be used by converting a packet based specification into a rate based specification, a different approach can be used in order to preserve the accuracy of the packet based specification. In the packet based model, packets are not released at each node until the minimum interarrival time, x_{min} has passed since the sending of the previous packet from the same source. This guarantees that at each hop the source traffic characteristics are maintained.

It is also possible to implement a *work conserving* scheme. To be work conserving the routers must send waiting packets whenever the required link is idle. This means that the flow may distort and this distortion must be accounted for when determining the delay and loss bounds. Accounting for the distortion within the network is more complicated than using rate control as the flow characteristics seen at a router are directly dependent on the effect of congestion at other routers earlier in the path.

Non-work conserving (rate controlled) schemes reduce the buffering requirements within the network as distortion can not cause increased burstiness of a flow. Rate control also makes it easier for the CAC protocol of real-time channels to redistribute slack, since changing guarantees at a router does not affect the traffic seen by other routers. However, because these schemes may be forced to hold packets even when the required output link is idle, they can increase the average delay of all flows in the network. Although the average delay may be increased, [37] showed that at least the *worst case* delay is not increased by the addition of rate control. There is also the additional expense of building the rate controllers into the routers. The trade-off between work conserving and non-work conserving schemes is the trade-off between improving the average case performance and improving the worst case performance. The work presented in Chapter 4 allows for a work conserving scheme.

2.5 Conclusion

This chapter has outlined the requirements and approaches for providing QoS guarantees to real-time traffic flows. Best effort methods are suitable when no firm QoS bounds are required. Both bandwidth reservation and schedulability analysis can be applied to support QoS when guaranteed bounds are required. For the former, these bounds will be tied to the bandwidth allocation, for the latter, they will be determined by the source traffic specification of the flows. Hybrid systems have also been proposed providing both schedulability analysis for admission control with its relevant scheduler and a separate reservation based scheduler, a link scheduler, for sharing excess bandwidth in times of low congestion and monitoring use in times of congestion [28]. Such approaches may ultimately provide the best trade-offs of simplicity and accuracy.

Several requirements are fundamental to both methods of providing QoS guarantees. Over the end-to-end connection the network must support:

1. known routing
2. methods for specifying QoS requirements
3. methods for distributing QoS bounds over the nodes in the path
4. specification of network flow characteristics.

Routers must be able to:

1. determine the source characteristics of traffic on their input links
2. reserve resources necessary to meet the local delay and loss limits
3. ensure isolation of performance guarantees from competing connections.

The delay and loss bounds that can be offered by a router are largely determined by the packet handling policy the router implements. The next chapter examines some of the packet service policies proposed in the literature and compares the implementations and QoS guarantees provided by these various policies.

Chapter 3

Router Scheduling Policies for Real-Time Communication

3.1 Introduction

The previous chapter identified the tasks that need to be performed to provide guaranteed QoS to real-time applications and divided these tasks into those that are performed end-to-end and those that are performed at each router. Although all of these tasks are necessary to provide QoS guarantees, the actual bounds on QoS, (i.e. jitter, delay and packet loss) are determined by the service received by a flow relative to other competing flows at the routers along the flow's path. It is at these routers that the dynamic arrival of traffic from various flows brings about congestion. The end-to-end tasks only serve to support the establishment of flows over the routers and converting the end-to-end QoS bounds into per router bounds. In this chapter we therefore examine the per router tasks in detail and evaluate the published methods of performing these tasks. In the previous chapter, we identified the per router tasks as:

1. Define the characteristics of traffic on the input links.

2. Reserve resources necessary to meet the local QoS bounds.

3. Ensure that guaranteed QoS bounds are not violated by competing traffic.

The chapter begins with an explanation of how performance of these tasks can guarantee QoS bounds by defining the relationship between the bounded traffic arrival patterns of flows as seen by a router, the service behaviour (or queuing model) of the router, and the QoS bounds that can be offered by the router based on the arrivals and service. In the existing router models proposed in the literature for supporting guarantees, the traffic arrival patterns are bounded by one or both of the traffic models discussed in section 2.4.2 or equivalent models. However, there is a far greater variation in proposed service behaviours. These service policies vary in the complexity of their implementation, the computational complexity of determining the bounds offered, and the accuracy of these bounds. The service policies also share some similarities and can be classified based on a few common behaviours. A taxonomy of these fundamental characteristics is defined to provide a basis for classifying service policies.

The remainder of the chapter examines several router queuing policies that have been introduced in the literature for providing deterministic service for real-time traffic, classifies them within the taxonomy and gives the best QoS bounds currently published for each policy. The advantages and disadvantages of these various policies and the causes of inaccuracies in the published QoS bounds are discussed and serve as the motivation for a new scheduling analysis introduced in Chapter 4.

3.2 Relationship of Traffic Arrival, Service Policy and QoS Bounds

The delay, jitter and packet loss experienced by flows competing for a shared output link vary over time depending on the the number of packets queued for service at the router and the order in which these and future packets are serviced. The QoS specification defines upper limits on these parameters and the specification can be guaranteed as long as these upper limits are not exceeded during the lifetime of a flow. To provide this guarantee the router must be able to determine the *worst* *case* bounds that might be experienced by a flow. These bounds can be directly related to congestion, as will be shown later, and therefore the problem becomes one of determining the worst case congestion that could occur based on the input traffic and the service policy. It is possible that this worst case may never appear depending on actual congestion seen, which may be significantly less than the worst case depending on the pattern of arrival, but the aim is to provide a bound that is *guaranteed* not to be exceeded.

The congestion at time t for a particular outgoing link l, is the difference between the amount of traffic that has arrived at time t for l and the amount of this traffic that has been serviced (i.e. sent to the next node) at t. This relationship can be shown by a service curve [18], as in figure 3-1.

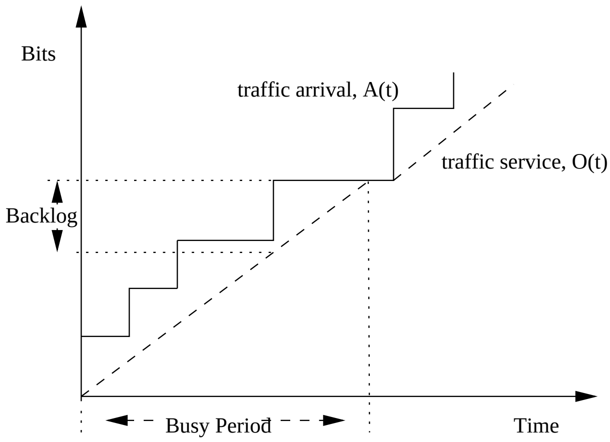

Figure 3-1: Backlog and QoS at a Node

The traffic arrival function, $A(t)$, indicates the number of bits that have been received at time t from all flows sharing the output link. The arriving traffic is characterized by step functions because of the store and forward behaviour of the routers. The arriving packets are not available for sending until the entire packet has been received. In a network which allows forwarding of packets before they fully arrive, the incoming traffic curve may be smooth. The step function, however, leads to potentially greater backlog as the outgoing link may be idle while awaiting the complete arrival of packets. For example, in figure 3-1, the step after the marked busy period could have been partially serviced after the point where the traffic arrival and traffic service curves intersect. The step function is used in the analysis in this thesis as the more general model. The extension to a smooth model can be made for cut through networks, but the reduction in backlog is small as is the resulting difference in calculated delay bounds.

The traffic service function $O(t)$, represents the number of bits which have been sent by time t. Note that it is possible that the link was idle during some of this time period. This occurs in the figure after the marked busy period. For a short interval the service function has a slope of zero. During this period there are no packets to send.

The service function need not be smooth; but a steady rate service is shown in the figure. The outgoing link is typically able to service traffic at a fixed rate. This is represented by the service function in the figure, which services a given number of bits per second. At any point in which the traffic service function intersects the traffic arrival function, the outgoing link has caught up with the incoming traffic and no traffic is waiting to be sent.

The congestion or *backlog* experienced at time t is the difference between the input traffic that has arrived at time t for the output link and the amount of this traffic that has been sent on the output link at time t. The backlog will depend on the service discipline. When traffic arriving after t will be sent *after* traffic that has arrived by time t, the worst case delay seen by packets arriving at time t will be the time needed to clear the backlog that exists at t. Call this backlog $b(t)$. If the service rate of the output link is defined to be r_{out}, the delay at time t can be expressed as $d(t) = b(t)/r_{out}$. The number of bits lost at time t is the difference between the backlog and the available buffer space. The buffering required to prevent packet loss at time t is therefore equal to the backlog at that time, $b(t)$.

The worst case upper bound on delay and packet loss seen by a flow is determined by the maximum value of the backlog over the lifetime of the flow. However, we can limit the time over which we must search for the maximum backlog by recognizing some characteristics of the flow and router behaviour. An output link is not saturated as long as it meets the condition:

$$\sum_{i=0}^{j} S_{max,i}/x_{ave,i} \leq r_{out} \tag{1}$$

That is, the sum of the average rates of all of the sources sharing an output link does not exceed the rate of that output link. For any non-saturated outgoing link the service function will intersect with the arrival function at various times. The time intervals between points where the functions diverge and then intersect is called a *busy period*. At the start and end of a busy period no packets are waiting to be sent. If we can determine during which of these busy periods the worst case backlog occurs, then we need only examine the backlog during this interval to find the upper bounds on backlog and therefore delay and packet loss.

We can cause the first busy period to have the worst case backlog by generating the worst case traffic arrival on each flow at the beginning of this busy period. At time zero there will be no traffic waiting for the outgoing link. Traffic is then generated at the maximum rate allowed by the flow specifications. Intuitively one would expect that if the arrival is as bursty as possible, this will maximise the traffic arrival function $A(t)$, and that since the traffic service function does not vary, this behaviour will maximise the backlog. For a formal proof of this behaviour see the section on *all greedy* and *staggered-greedy* regimes in [60][61].

Several packet service disciplines do not service packets in the order of their arrival time but on other values such as their *deadline*. The deadline of a packet is the packet's expected arrival time plus the delay bound for the flow at the router. In routers employing such schemes, it is possible for packets to be scheduled ahead of those already queued. This can lead to the maximum delay occurring at a time other than at the time of maximum backlog. For example, consider three flows with the worst case packet arrival behaviour shown in figure 3-2.

Flows 1 and 2 have a delay bound of 2τ. Flow 3 has a delay bound of 5τ. Assume that the output link is idle at time some time t. Since any traffic that arrived earlier

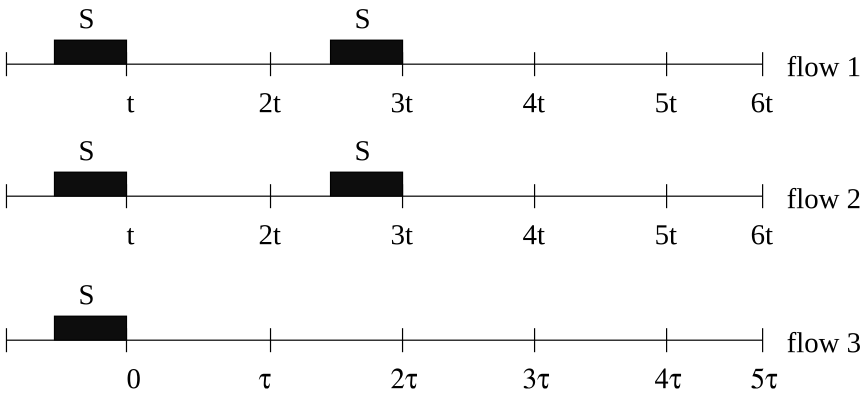

Figure 3-2: Individual Flow Arrivals

than this time has been sent, we don't need to consider earlier times, so we consider this time to be zero. Given a service rate of S/τ bits/s on the output link, the service curve for this example is shown in figure 3-3.

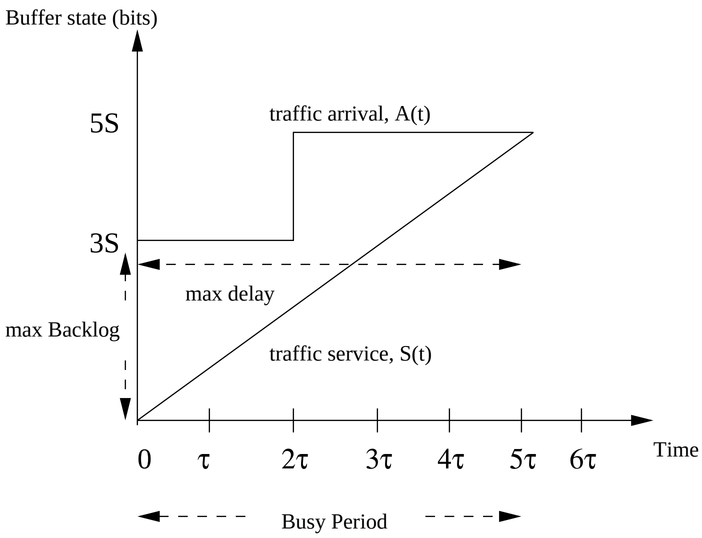

Figure 3-3: Service Curve

The maximum backlog over the busy period first occurs at time $t = 0$ when all three flows have packets queued. The backlog may equal this peak at other times (as it does at time 2τ; but this backlog is never exceeded. However, the worst case delay experienced by flow 3 is greater than the 3τ needed to clear a backlog of $3S$. At time

2τ the output link will have serviced two packets. Since flow 1 and 2 have an earlier deadline than flow 3, the first packets from flow 1 and 2 will be chosen for service. At time 2τ, two more packets from flows 1 and 2 arrive; both with a deadline of 4τ (i.e. 2τ (arrival time) $+2\tau$ (delay bound)). Again these deadlines are earlier than flow 3 (0 (arrival time) $+5\tau$ (delay bound)) so they will be serviced before the flow 3 packet. The flow 3 packet will not be fully sent until time 5τ giving it a total delay of 5τ.

We can still use the service curve to determine worst case delay and backlog and the worst case will still occur when all flows send at their maximum rate at the start of the busy period. However, we need to take into account that the worst case delay for a flow may not simply be the time to clear the worst case backlog. The worst case backlog still defines the buffering requirements necessary to prevent packet loss. The minimum delay can be determined in a similar manner to maximum delay by starting a busy period with flows generating the minimum amount of traffic and determining the delay under such an arrival. Jitter can then be determined by the difference between the minimum and maximum delays.

It is relatively easy to determine the service function as this is just the rate at which the output link can send packets. The arrival function is also straightforward, assuming the incoming flows adhere to the traffic specification. This implies that the service disciplines rely on mechanisms for ensuring this specification is met, or alternatively account for the possible variations from the specification due to varying congestion. The challenge is to determine a way of efficiently finding the worst case delay, jitter and backlog based on the arrival and service functions. The policy for selecting the next packet to service determines these values as shown in the example and therefore is critical to the QoS guarantees that can be offered by the network. The remainder of this chapter discusses the service behaviour of various service disciplines and compares the QoS bounds they can offer.

3.3 Current Service Disciplines for Real-Time Networks

The service discipline of a router defines the order in which arriving packets gain access to the shared output link. Although real-time packet scheduling is closely related to queuing theory, it is distinguished by the aim of minimizing worst case rather than average case delay within a real-time, deterministic (as opposed to probabilistic) network. The term *flow theory* has been suggested to distinguish between the two [17]. Flow theory has been studied extensively in the literature, and scheduling disciplines that bound maximum delay have been proposed including FIFO [91], Earliest Deadline First [23], Earliest Deadline First with Jitter [82], static priority queues [90], processor sharing [62], and virtual clock [96][95][26][84]. A comparison of several disciplines is given in [87][89][93][94][2]. The following sections describe the proposed packet scheduling disciplines for routers and the QoS bounds each can guarantee. It extends the existing comparison papers to include additional scheduling disciplines and identifying other criteria for comparison.

3.3.1 FIFO

The service discipline with the simplest implementation is a first in first out or FIFO queue (figure 3-4).

As packets arrive at a node, they are placed in a queue for the appropriate output link and serviced from this queue based on order of arrival. The FIFO model makes determining the maximum backlog and delay at a router simpler than most other disciplines since for any time t traffic arriving after t can not delay packets that have arrived by time t. We can not make this assumption for many of the other service disciplines listed above.

Because arriving packets can not preempt queued packets, the worst case backlog and delay will occur when all flows attempt to access the link simultaneously at their maximum rate. If j flows are competing for access to an output link with rate r_{out},

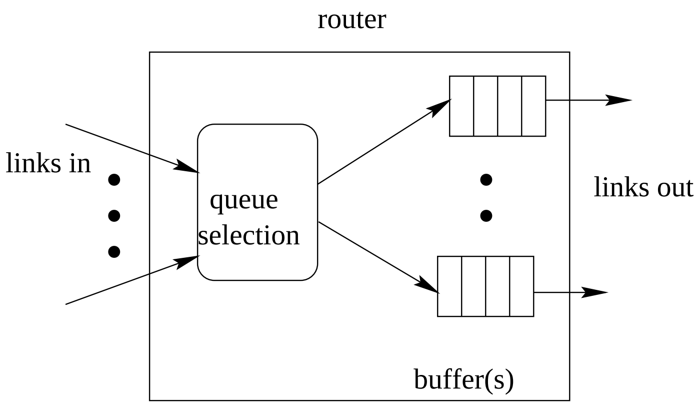

Figure 3-4: FIFO

and a flow i has a maximum packet size of $S_{max,i}$, then under the assumption that the total maximum rate of all flows sharing the output link is less than or equal to the output link rate, i.e.:

$$\sum_{i=0}^{j} \frac{S_{max,i}}{x_{min,i}} \leq r_{out} \tag{2}$$

then the maximum delay experienced by any packet is bounded by the time to send one packet from each flow:

$$d = \sum_{i=0}^{j} \frac{S_{max,i}}{r_{out}} \tag{3}$$

This worst case delay will occur for the last packet from the initial burst to access the output link. This packet must wait for the other sources to complete their transmission of their packets and for its own transmission.

We can determine worst case backlog without the restriction of equation 2 as long as the sum of the average rates is less than the output link rate:

$$\sum_{i=0}^{j} \frac{S_{max,i}}{x_{ave,i}} \leq r_{out} \tag{4}$$

The resulting worst case backlog occurs when all of the flows' packets arrive at their peak rate until sending further packets would exceed their specified average rate. The sources then stop sending until the end of the averaging interval. This maximizes the burstiness of the flows. A proof of this worst case behaviour and the related bounds are given in [91] and [71]. These bounds are typically larger than those in equation 3, but since the restriction of equation 2 is assumed for some of the other scheduling disciplines, equation 3 is used for comparison [1].

If the flows are reshaped to meet their original specifications at each node n, then the end-to-end delay over a H hop path is:

$$D = \sum_{n=0}^{H} d_n \tag{5}$$

If the flows are not reshaped, then delay at a node n is dependent on nodes 1..n-1. For the details of delay and backlog bounds for unshaped traffic, see [71].

3.3.2 Priority Queues

An extension on FIFO queues is to have multiple priority queues, as shown in figure 3-5.

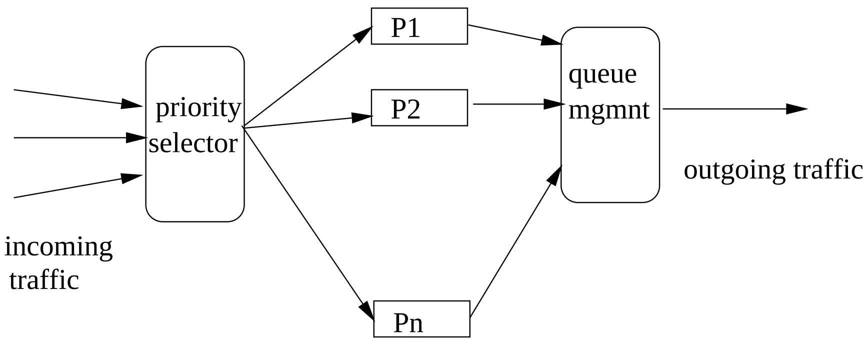

Figure 3-5: Priority Queues

Each incoming flow has a priority fixed at the time of flow establishment. As traffic arrives, a priority selection mechanism places each packet in a queue appropriate to its priority level. Packets in these queues are then selected for the output link by a queue management policy. This queue management policy and the method of ordering packets in the priority queues can vary; but a simple policy of placing packets in the priority queues in FIFO order and selecting the highest priority queue with queued packets results in the maximum delay for for flows of priority p given in equation 6.

[1]Both the restrictive and less restrictive bounds are used in the simulations in chapter 6.

This is similar to equation 3. The only difference is that only flows in *higher* priority queues or those that share the same priority queue need be included in the summation.

$$d = \sum_{\forall i : p_i \geq p} \frac{S_{max,i}}{r_{out}} \tag{6}$$

Cruz [18] derived delay bounds for such a model for the rate based (σ, ρ) traffic model. For the packet based model delay bounds are given in [92][87] under the name of rate controlled static priority (RCSP) queues. Both of these results rely on the addition of regulators that reshape the traffic to adhere to the flow specifications at each node. Packets are assigned *eligibility times* based on the flow's traffic specification. For example, in the packet based model, if the value of x_{min} is set to 2 time units, then if the previous packet arrives at time t the next packet will not become eligible until time $t + 2$. If the packet arrives before this time, the packet is held by the regulator until its eligibility time arrives. Under these conditions, the delay bounds for a given priority class is the same as defined in equation 6.

3.3.2.1 Rotating Priority Queues

The Rotating Priority Queues discipline (RPQ) [52] extends the priority queuing model to bring the scheduling to a closer approximation of Earliest Deadline First (EDF) without the need for packet ordering or selection from the queue (see section 3.3.3). RPQ is illustrated in figure 3-6. The central hub represents the queue priority and the spokes are FIFO queues. The FIFO queues are rotated around the hub every Δ time units, so a given queue will have different priority levels as it is rotated.

When a packet arrives at the scheduler, it is placed in the queue p, of lowest possible priority (largest number) that still allows the deadline of the packet d to be met. This can be guaranteed if $d \geq d_p \geq p * \Delta$ where Δ is the rotation interval and d_p is the deadline for the queue p.

Packets are selected for output in FIFO order from each queue. Packets in queue 0 are sent first. If a sending queue empties, then packets from the next highest priority queue, $q + 1$, are sent. A restriction exists on accepting a new flow that all packets in

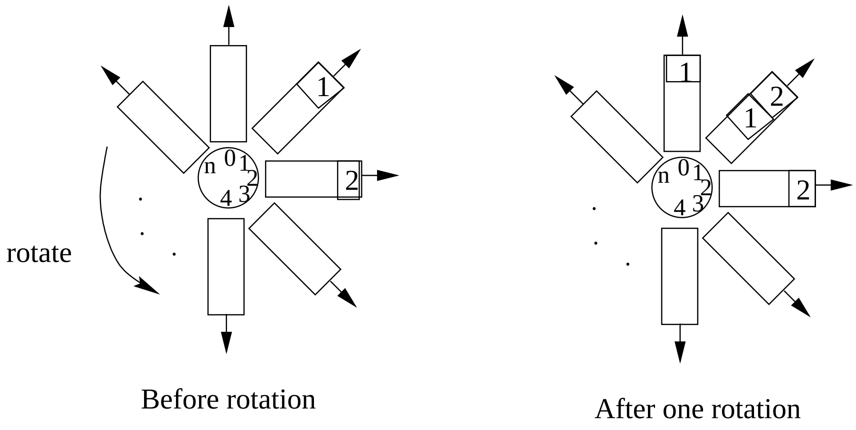

Figure 3-6: Rotating Priority Queues

queue zero must be able to be sent before the next rotation interval. At the end of each rotation interval Δ, the queues are "rotated" counterclockwise such that queue n becomes queue $n-1$ and queue 0 becomes queue n.

A set of flows grouped into priority classes of $C_1, C_2, \ldots, C_n$ with a maximum packet transmission time of c_{max} is schedulable under RPQ if and only if the time required to transmit the total traffic arriving from all flows by time t is less than or equal to t:

$$\sum_{i \in C_1} A_i(t-d_1) + \sum_{i \notin C_1} A_i(t-d_i+\Delta) + c_{max} \leq t \tag{7}$$

Intuitively, the total packets generated by the flows of priority 1, C_1, are the packets generated in the period $(0, t-d_1)$. Packets generated after $t-d_1$ will not be required to be served by time t. For connections of priority $2 \rightarrow n$, only packets generated in the period $(0, t-d_i+\Delta)$ will need to be served. The additional Δ is due to the fact that packets in queue 1 can be served immediately, whereas other queues must be rotated before they can be served. c_{max} exists to allow for the transmission of a packet already accessing the link, since preemption of a packet that has already begun transmission is not allowed. The formal proof of this condition is given in [52].

As $\Delta \rightarrow 0$, RPQ $\rightarrow$ EDF. The main difference between EDF and RPQ is that for RPQ the delay is based on the chosen value of Δ and the ordering of packets is not guaranteed to match their actual delay requirements. For example, a flow with a delay bound of 3 and a flow with a delay bound of 4 may both be assigned to a priority level which guarantees a delay bound of 3. Within this priority level, there is no guarantee that packets from the flow with delay bound 3 will be serviced first.

As $\Delta \rightarrow \infty$, RPQ $\rightarrow$ static priority queuing. Packets in queue p will only be serviced if there are no packets remaining in any queues $< p$. Closed form bounds on delay are given in [52] but only for the case in which there is one flow per priority level. For multiple connections per priority level, deadlines must be assigned and then tested for schedulability. If the set of flows is schedulable, then all of the packets of a given priority level will be sent by the time of rotation Δ. The maximum delay of packets of priority level p in an RPQ scheduler is therefore $p * \Delta$.

3.3.3 Earliest Deadline First

The Earliest Deadline First (EDF) scheduling policy, also called Earliest Due Date (EDD), schedules packets in the order of their deadlines. The deadline of a packet is defined to be the arrival time of the packet plus the delay bound assigned to the packet. When a packet arrives at a node, it is inserted into a sorted priority queue based on this deadline. For the single node case, non-preemptive EDF is known to be optimal under the restriction that maximum packet transmission time is identical for all flows [30] [29]. This is consistent with the known optimality of EDF as a real-time scheduling algorithm [53]. Under the restriction that all flows have a deadline equal to their packet interarrival time, the flows will be schedulable as long as the link utilisation is less than 100%.

EDF scheduling alone does not guarantee a specific delay bound at a node, only that if it is possible to schedule all of the packets, then EDF will succeed in scheduling them. In addition, EDF was proposed in a single processor environment, so it needs to be extended to the chain of schedulers comparable to a multi-hop path of nodes in a network.

3.3.3.1 Delay Earliest Due Date

Delay-EDD [23] extends the idea of EDF scheduling into an admission control protocol which determines at each node what minimum delay bound can be met for a new traffic flow specification without violating delay guarantees to existing traffic flows. Flows are regulated by assignment of a deadline equivalent to the deadline that would have been assigned if the packet conformed to its agreed rate. For example, if a flow agrees to a packet rate of 5 packets/time unit, then the deadline assigned to the jth packet will be $j * 1/5 + d$, where d is the delay bound guaranteed to the flow, regardless of its actual arrival time at the node. This allows Delay-EDD to accommodate traffic distortion caused by variations in congestion at earlier nodes. When a packet arrives for a given outgoing link, the packet is assigned a deadline as described above and then placed in a sorted priority queue for the outgoing link based on this deadline.

Delay-EDD requires that the outgoing links must not be saturated, that is the total traffic rate of incoming flows sharing the link must be less than the link's rate. In addition, Delay-EDD must ensure that the *scheduler* is not saturated. It is possible for the scheduler to be saturated even if the link is not saturated. Consider the node illustrated in figure 3-7.

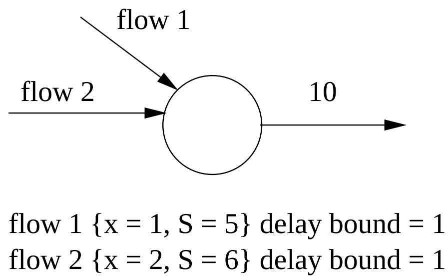

Figure 3-7: Scheduler Saturation

The incoming rate of flow 1 is 5 (5 bits every 1 time unit) and flow 2 has a rate of 3 (6 bits every 2 time units). The total incoming rate is 8 which is less than the outgoing link rate of 10, so the outgoing link itself is not saturated. It is still not possible to meet the delay bounds of the flows. If one packet arrives from each flow, the time taken to service both packets is $5/10+6/10 = 11/10 = 1.1$ time units. Given both flows have the delay bound of 1, either may be serviced first. The packet that is serviced second will not be sent until 1.1 time units after its arrival. Therefore, it is guaranteed that one of the flows will miss its deadline. Since the ordering of packets with the same deadline is arbitrary, the packet that misses its deadline is also arbitrary. We can not provide a guarantee that either flow will meet its delay bound. A test of node saturation is necessary but not sufficient to guarantee the flows are schedulable. An additional constraint is given in [23] that can be used to guarantee that scheduler saturation does not occur. Given the condition that for a set of J channels each flow i has a minimum interarrival time greater or equal to the time required to service one packet from each of the flows:

$$\forall i \in J : x_{min,i} \geq \sum_{k=1}^{J} S_k/r_{out} \tag{8}$$

the flows can be scheduled if the delay bound of each flow i, d_i meets the following criteria:

$$\forall i \in J : d_i \geq \sum_{k=1}^{J} S_k/r_{out} + S_{max}/r_{out}. \tag{9}$$

The last term is included to allow for the delay caused by a lower priority packet that is currently being serviced and can not be preempted.

These tests are similar to those which must be done in each queue under the RPQ scheduler. Unlike RPQ, the delay is not tied to any choice of scheduler controlled value (such as Δ). Instead, the minimum delay which can be offered at each node is determined solely by the competing traffic, and if the sum of these delays is less than or equal to the required end-to-end delay then the channel can be established. Sufficient conditions for schedulability of Delay-EDD are presented in [23]. This paper also defines the delay constraints for channels that are not constrained by equation 8. Necessary and sufficient conditions are derived in [102]. Necessary and sufficient conditions using the (σ, ρ) model are derived in [29].

3.3.3.2 Jitter Earliest Due Date

Continuous media applications such as video and voice are affected not only by delay but also by variations in delay. Packets arriving too soon can negatively impact the application by making higher buffering demands on the receiver. This variation in delay is called *jitter*. Jitter Earliest Due Date [82] extends Delay-EDD to provide lower bounds, as well as upper bounds, on delay and new buffer requirements for flows with the bounded jitter.

Jitter-EDD nodes are identical to Delay-EDD nodes with the following exceptions:

- In addition to the flow regulators that enforce the minimum packet spacing (i.e. the maximum rate), there are also jitter regulators for each flow that enforce a *minimum* delay.
- In addition to per node delay bounds, jitter bounds are also defined for each flow at a node.

The jitter-EDD node is shown in figure 3-8.

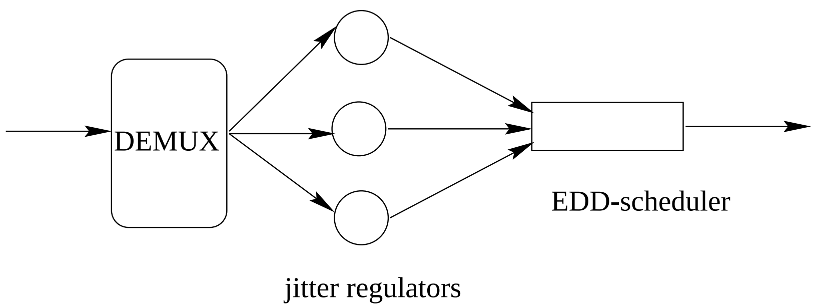

Figure 3-8: Jitter-EDD node

As packets arrive on an input link, they are separated by flow and passed to a jitter regulator assigned to that flow. The regulator assigns an *eligibility time* to the packet. The packet is not forwarded to the EDD-scheduler until this eligibility time has arrived. The eligibility time of a packet arriving at time t with delay bound d and jitter bound J is equal to $t + d - J$. Packets from a flow that bounds jitter will therefore not be sent more than J time units before their deadline. Packets from a flow that does not need to bound jitter would be eligible for scheduling at their time of arrival.

The scheme as described so far will guarantee the jitter behaviour at a single node based on the original specification of the flow. However, the traffic regulator, which ensures that the traffic meets its flow specification, only enforces minimum time between packets. It does not enforce a maximum time between packets, which would force the source to conform to its original jitter behaviour. The flow regulators for J-EDD must be extended to remove any jitter experienced by the packet in the previous node.

A packet can be sent any time τ between $t+d-J$ and $t+d$. To determine the packet jitter at the previous node, the difference between a packets deadline $(t+d)$ and the time it is actually sent on the output link of the previous node $(t+d-J \leq \tau \leq t+d)$ is stamped on the packet. At the next node, the packet is held at the regulators until this time has elapsed before being sent to the EDD-scheduler. This forces packets to be delayed if they arrive early. The jitter-EDD policy effectively moves the buffering of packets from the receiver to the network and reduces the overall buffering required at each node by reducing the number of packets which could potentially arrive early (i.e. reducing the worst case scenario). The constraints on delay are the same as delay-EDD.

3.3.4 Stop and Go

Stop and Go [33][32] bounds flow delays by dividing a link's sending time into fixed size frames and assigning flows to frames whose sending rate matches the delay bound requirements of the flow. The implementation of a stop and go node with G different frame types is shown in figure 3-9.

Packets are marked with their frame type and as they arrive they are sent to the frame regulator for that type. The frame regulator manages two sets of frames: those

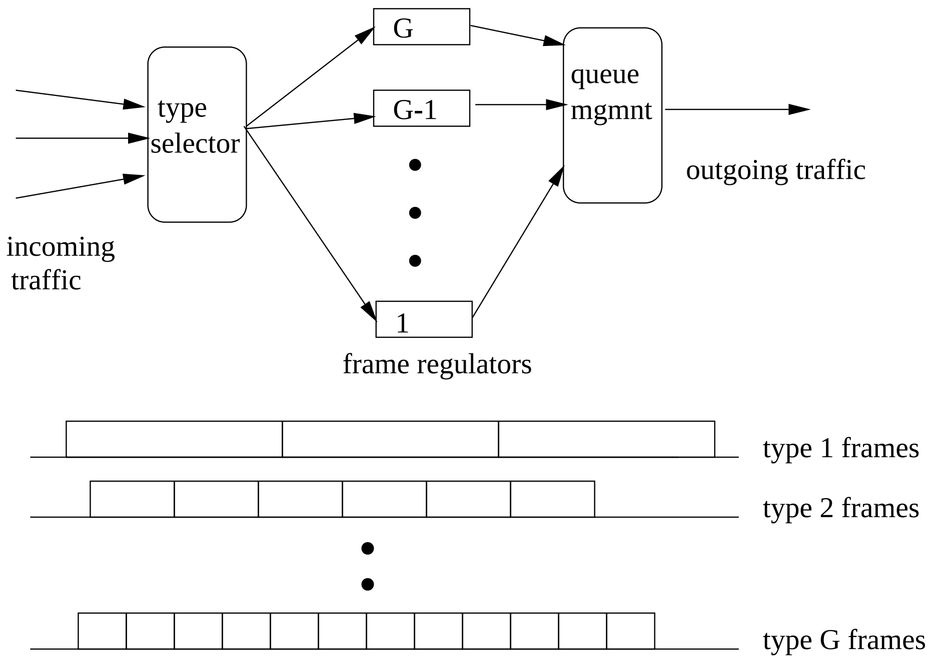

Figure 3-9: Stop and Go

arriving and those departing. The arrival and departure frames of a given type are the same length as shown in figure 3-10.

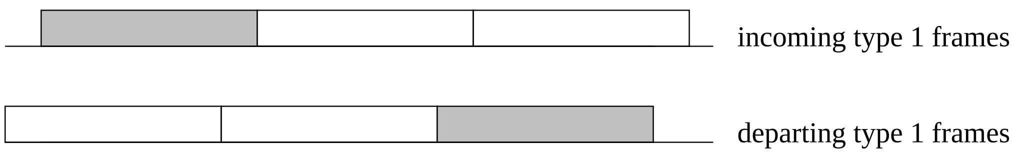

Figure 3-10: Incoming and Outgoing Frames

As the packets arrive they are associated with the current incoming frame.

Packets in a given incoming frame are not eligible to be sent until the incoming frame has fully arrived. So a packet arriving at time τ at the beginning of an incoming frame with an arrival duration of T must wait until $\tau + T$ before becoming eligible to be sent. Additionally, packets must be eligible at the *start* of a departing frame to be sent during that departing frame. So in figure 3-10, the packets in the first arriving frame (the shaded area), can not be sent in the second departing frame as their frame has not fully arrived at the time the second departing frame begins service. Therefore, they must wait until the third departing frame. Due to this alignment of incoming and outgoing frames, the delay experienced by a packet in a frame of length T will be between T and $2T$. The delay T occurs when the arrival frame arrives at the same time the departure frame begins sending. In this case the worst case delay occurs for the first packet of the arrival frame which must wait time T for the full arrival of its frame. The delay $2T$ occurs when the arrival frame arrives instantaneously after the departure frame begins sending. In this case, in addition to waiting for the full arrival of the frame, the first packet must also wait for the next departure frame T time units later.

Jitter is also controlled by the framing strategy. In [33] the jitter is stated to be between $-T$ and T. However, this assumes that packets can be reordered between an arrival frame and a departure frame (i.e. the last packet in the arrival frame could be the first one sent in the departure frame causing a jitter of $-T$). Given the simple implementation based on FIFO queuing of packets of a particular frame type, as suggested in the paper, a fairer comparison, would be a maximum jitter of T, which is the variation in the delay that can be experienced as given above.

To avoid packet loss, a node must allocate buffer space for each frame type equal to $3 * r_g * T$ where r_g is the total rate allocated to connections of type g. This buffering allows for the worst case when a frame arrives just after the start of a departure frame. The node must buffer the departing frame's packets, the newly arrived frame's packets and the packet's of the frame which may now be starting to arrive. The buffering requirements will depend on the alignment of arrival and departure frames and may be less than this amount but will not be more.

The above bounds are dependent on admission conditions as given in [33]. The conditions important to the comparison with other disciplines are the requirements that:

1. The source traffic associated with a frame type of length T must send no more than an agreed rate $r * T$ bits during any frame period.

2. For all g frame types, the following must be met:

$$\sum_{n=1}^{g} \sum_{i \in g} r_{g,i} T_g + S_{max} \leq T_g * r_{out} \tag{10}$$

The first constraint ensures that the traffic adheres to its agreed rate. The second constraint ensures that the traffic will not exceed the output link rates. Intuitively, the second constraint states that all the traffic which could arrive during a frame of length T_g is guaranteed to be able to be sent by the end of the frame time. The additional term of S_{max} allows for the sending of one maximum size packet to allow for a lower priority packet still on the link due to the non-preemptive nature of stop and go.

3.3.5 Virtual Clock

The Virtual Clock (VC) discipline [96][95] is based on a priority queue, similar to Delay-EDD. Unlike Delay-EDD, the ordering in the queue is not based on a deadline. Instead it is based on the time which packets would have been sent if the node were using Time Division Multiplexing (TDM) to schedule packets. In TDM the available sending time of the link is split amongst the connections and connections can only send during their assigned time slice. VC protects flows from the effects of other flows, similar to TDM, but unlike TDM, the queue is work conserving since the TDM is only used to determine the *order* of service not when packets are allowed to be sent. VC takes advantage of any excess bandwidth which may be available whenever flows do not use their full time allotment.

The discipline works by maintaining a real-time clock and two virtual timers for each flow, *Virtual Clock (VC)* and *auxiliary Virtual Clock (auxVC)*, which are used to provide flow monitoring and packet scheduling. A flow is specified by its average rate AR and the time interval over which this rate is determined AI. Whenever a packet arrives the timers are advanced by the minimum packet spacing that maintains the average rate ($1/AR$ if all packets on a flow are the same size). Effectively this sets the timers to the earliest time that the next packet is eligible for sending. The packets are stamped with a priority of auxVC and placed in a sorted priority queue for sending.

Every $AI * AR$ packets, VC is compared to the real-time clock. If VC is running ahead of the real-time clock, then the packets are arriving faster than agreed and action is taken to restrict the misbehaving flow. If it is running behind the real-time clock, then VC is set to the current real-time to prevent the flow from sending at a lower rate than AR in one interval followed by a higher rate than AR in the next without being detected. VC is compared with the real-time clock every $AI * AR$ packets rather than every AI seconds. If the check were made every AI seconds, the flow could send more than $AI * AR$ packets and the misbehaving flow would not be discovered until AI seconds had passed. By examining the rate based on the number of packets sent, the misbehaving flow will be discovered as soon as it sends a packet that exceeds its agreed rate.

Packet scheduling needs to be handled by a separate timer. It is not immediately obvious why VC alone can not be used to schedule the packets since it keeps an ordering based on arrival time and send time. The reason a separate auxVC is needed is that a flow could send no packets for a long time and then send a large burst towards the end of AI and still meet its average rate when considered over the whole interval. Since the VC of the flow would not have advanced, it would be small compared to other flows that had been sending more regularly. The bursty flow would gain priority over the other flows since its packets would be sent first. It has effectively gained "credit" for not using its resources earlier. Since those resources (link output) can not be saved up, the discipline must prevent such events. The auxVC timer prevents this by resetting itself to the real-time clock if this is greater than its current value at each packet arrival, thereby ensuring a bursty flow will not be serviced before other flows of equal or greater rate allocations.

When Virtual Clock was first proposed, no method was provided for determining bounds on end-to-end delay. More recently, end-to-end delay bounds have been independently determined in [35] and [26]. Extensions to VC have been proposed to allow for flows with variable rates in both non-work conserving [50] and work conserving [37][36] models.

3.3.6 Leave-in-Time

The leave-in-time service discipline [25] is closely related to virtual clock and jitter-EDD. The leave-in-time node is designed in the same way as jitter-EDD shown in figure 3-8. Like VC and J-EDD, leave-in-time determines a transmission deadline for each packet and places packets in a sorted priority queue for transmission based on this deadline. Like J-EDD, but unlike VC, leave-in-time also defines an eligibility time for each packets. Packets can not be placed in the sending queue until this eligibility time is reached.

The transmission deadline for a flow i packet in leave-in-time is determined by the time the packet would have been sent if it was being serviced from a FIFO queue by a server of a defined rate r_i, i.e. S_i/r_i (S_i is the packet size). This is very similar to the service defined in VC which defines the transmission deadline based on the service that would have been received if the packet were being served based on TDM.

Unlike jitter-EDD, [25] gives bounds based on the rate based (σ, ρ) model, and the regulators must enforce this model. Since it is a combination of J-EDD and VC, leave-in-time provides upper bounds on end-to-end delay, delay jitter, buffer space, and an upper bound on the probability distribution of end-to-end delays. The bound on distribution is the primary contribution to new QoS guarantees. This could be useful to applications which can tolerate some bounded loss.

3.3.7 Fair Queuing

The fair treatment of packets from multiple sources in WANs was identified as an issue for congestion control before providing guarantees became a major topic of research. The existing congestion mechanisms (window-based congestion) rely on the cooperation of the source node to reduce its sending window when congestion is detected. However, there was no mechanism to prevent a source from ignoring congestion conditions and continuing to send at a high rate. A mechanism was needed to prevent a misbehaving source from gaining more than its fair share of the available resources. If N flows are sharing an output link, then a fair scheme would allocate $1/N$ of the link bandwidth to each of them. The fair queuing discipline, was introduced in [21], to allocate a fair share of the bandwidth to each source.

Fair queuing services each flow for a set amount of transmission time in a round-robin fashion. Any unused transmission time is fairly distributed among all of the flows. Fair queuing can not be implemented at the packet level unless all packets are the same length. Otherwise, a flow with long packets will again receive a disproportionate amount of the link bandwidth. Servicing packets on a bit by bit basis, which would be fair, is not practical in networks. The scheme proposed in [21] uses a priority queue and assigns an ordering based on when the packet would be sent, were a bit by bit scheme in place. Packets are then serviced based on this assigned ordering.

Traffic with different priorities can be accommodated within the fair queuing model by adding a weighting to the allocation of a channel depending on its priority. The weighting reflects the number of bits per round robin cycle which the flow can send. So a weighting of 2 would give a flow double the bandwidth allocation of flows with a weighting of 1. Fair queuing effectively provides protection from misbehaving flows, guaranteeing that a flow will receive a minimum amount of service even during periods of high congestion.

3.3.7.1 Packet by Packet Generalised Processor Sharing

Fair queuing as proposed in [21] only provides a minimum guaranteed service. No traffic regulation or bounds on delay or packet loss were derived. Such bounds were derived for the rate-based source model by Parekh [62] under the name Packet by Packet Generalised Processor Sharing (PGPS). Bit by bit fair queuing is also known as generalised processor sharing (GPS).

Parekh proved that in a network of arbitrary topology using GPS scheduling where the weight assigned to a flow was in proportion to its rate ρ, an upper bound on delay can be determined solely using the rates and maximum burst size of flows through a node. As long as the total rate of the flows is less than the link rate and the flows conform to a σ, ρ model, then adding additional flows will not result in existing flows missing their deadlines [62][60][61]. The deadlines are also extended into the more practical case where packets rather than bits are scheduled (PGPS).

The design of a PGPS node is similar to VC. A priority queue is used and packets are scheduled in the priority queue based on the time they would have been serviced under a GPS server. Variations on PGPS scheduling and analysis are presented in [37][36][9][34][30].

3.4 A Taxonomy of Real-Time Packet Scheduling Disciplines

From the description of the proposed schedulers, some common characteristics emerge. The general real-time router node consists of two components

1. Traffic regulator - the first or both of:
    - peak rate regulator
    - jitter regulator
2. Queue scheduler

Jitter regulation is not present in all disciplines and for FCFS the queue scheduler is trivial (i.e. place at end of queue), but traffic regulation and some sort of queuing discipline is common to all of the proposed techniques. The disciplines can be classified by how they implement these two components and the resulting behaviour.

Most of the disciplines protect flows from the behaviour of other flows by ordering their packets based on when they *should* receive service as determined by their deadlines or promised service rate. As the packets arrive, they are stamped with their service time and placed in the output queue. Since the packets are placed in the output queue as soon as their order is determined, such services will never allow the output link to be idle if there is a packet to send. These disciplines are said to be *work conserving*. Delay-EDD, VC, leave-in-time, RPQ and PGPS are work conserving. In contrast J-EDD, Stop and Go, and RCSP assign packets an eligibility time and packets may not be placed in the output queue until the eligibility time arrives. The output link may remain idle even if packets are available to be sent. These disciplines are said to be *non-work conserving*.

The choice of a work conserving or a non-work conserving design is a trade-off between average efficiency and jitter control. Because they make maximum use of network resources, work conserving disciplines result in a typically lower *average* delay for traffic and more efficient use of the network resources. However, because the work conserving models only place a lower bound on a flow's service rate, they can not control jitter. Jitter can only be bounded by enforcing an *upper* bound on the service rate, ensuring that packets are not serviced too fast even if the faster service does not affect other flows. The relationship between bounded jitter and non-work conserving design is why jitter bounds are only provided for those disciplines that are non-work conserving and the implementation of these disciplines include jitter regulators.

It is possible to have a work conserving or non-work conserving node that does not provide delay guarantees, since this characteristic of packet service does not in itself provide any protection of a flow from other competing flows. To distinguish between disciplines that do provide guaranteed protection of flows and those that do not, the terms *rate allocating* and *rate controlled* are used in the context of real-time nodes that guarantee protection of a flow's service [93]. *Rate allocating* disciplines will service packets at a rate higher than the flow's specified rate *if doing so will not affect the rate allocation to other flows*. *Rate controlled* disciplines will *not* service packets at the higher rate regardless of whether the higher service rate affects other flows or not. Rate controlled disciplines are a sub-class of non-work conserving disciplines and rate allocating disciplines are a sub-class of work conserving disciplines.

Most of the disciplines tie the allocation of delay directly to the allocation of bandwidth. Stop and Go assigns a real-time flow to a pre-set delay based on bandwidth allocation (frame size), PGPS determines delay based on the weight (again the relative bandwidth allocation), and Virtual Clock orders packets based on the time division of bandwidth. Aras et.al. [2] labels such disciplines as *rate based.* The alternative approach is to determine the delay based on possible arrival patterns of competing traffic. This approach is referred to as *scheduler based.* Although it is still a condition that the total traffic flows do not exceed the outgoing bandwidth, the individual flows are not protected from each other by their bandwidth allocation, but solely by their queue ordering and the delay bound that can be guaranteed by that ordering. The rate based approach is less complex when determining if a new flow can be accepted since only the rate needs to be considered, but the parameters of bandwidth and delay are not independent. It is not possible in such schemes to have a low delay and low bandwidth allocation, which may be desirable for an application that has small amounts of time critical data (such as a sampling application).

Scheduler based disciplines protect flows by determining the effect on delay of adding an additional flow and checking if this would violate any existing delay guarantees. Because determining the effect of the new flow must consider the arrival patterns of the existing flows and the scheduling policy, scheduler based disciplines have a higher computational complexity for determining whether a new flow can be accepted. However, because delay is not tied to bandwidth, a low delay low bandwidth connection can be supported.

Scheduling disciplines may be preemptive or non-preemptive. In a preemptive discipline, the transmission of a lower priority packet can be discontinued by the arrival of a higher priority packet. When a non-preemptive discipline is used, the transmission of the lower priority packet is not discontinued even if a higher priority packet arrives while it is being transmitted. It is unlikely that stopping a single packet on an output link during transmission will significantly reduce delay and is likely to waste resources overall. One alternative proposal is a partially preemptive scheme [37] [36]. In this

	rate/scheduler based	rate allocating/controlled
FIFO	scheduler	allocating
Priority queue	scheduler	allocating
RPQ	scheduler	allocating
Delay-EDD	scheduler	allocating
Jitter-EDD	scheduler	controlled
RCSP	scheduler	controlled
Virtual Clock	rate	allocating
PGPS	rate	allocating
Stop and Go	rate	controlled
Leave in Time	rate	controlled

Table 1: Behaviour Taxonomy

scheme, individual fragments of a packet are not preempted. However, a higher priority packet will be transmitted before the remaining fragments of a lower priority packet. This scheme is useful in nodes where fragmentation is likely. Because of the overhead involved in preempting a packet or fragment already in transmission, preemption at the packet level is assumed not to occur in the network scheduling schemes. The most common approach is to assume non-preemptive behaviour and include the worst case sending time of the one packet in the delay bound. Because the small reduction in delay is unlikely to justify the cost of preemption, and because all of the introduced models assume non-preemptive behaviour, this characteristic is not included in the classification.

The main characteristics of the various disciplines are summarized in tables 1 and 2. Table 1 compares the disciplines based on their behaviour. The fact that disciplines exist across all the combinations of rate/scheduler and allocating/controlled is an indication of the fact that there is not one clearly superior approach and the choices represent trade-offs. Jitter control requires a jitter regulator (and therefore is rate controlled). Jitter controlling (rate controlled) disciplines exist for both the scheduler based and the rate based models. Where jitter regulation is not required rate allocation is sufficient and can provide better average performance. Scheduler disciplines allow bandwidth and delay to be allocated separately but at an increased computational cost when determining whether a flow can be accepted into the traffic mix. Rate allocating disciplines tend to be simpler to implement but can not efficiently accommodate applications whose bandwidth and delay requirements differ.

Table 2 compares the implementation of each of the real-time node components shown in general in figure 3-11. To aid in comparing the various disciplines, some of the notation in the table has been converted from the original notation into the equivalent notation in the packet or rate based models.

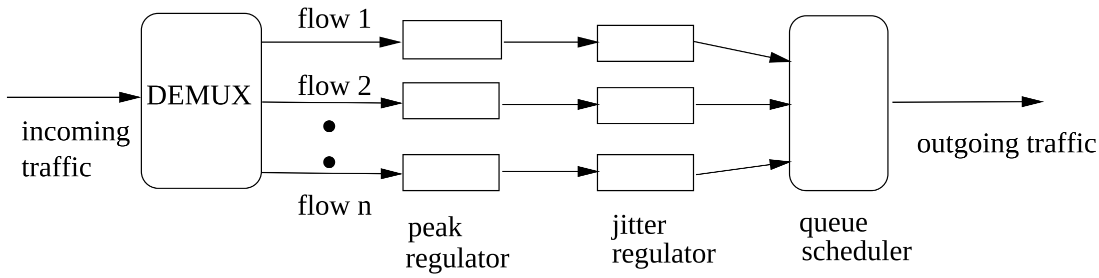

Figure 3-11: The General Real-Time Node

All of the models perform some form of peak rate regulation, either through a rate controller or by setting the deadline/service time to reflect the expected arrival time of the packet (based on previous packet arrivals) rather than the actual arrival time. The scheduler models require that packets are spaced by at least x_{min} time units. Therefore they will not become eligible to be scheduled until x_{min} after the arrival of the previous packet. The rate based models shape traffic based on average rate and packets are not eligible to be sent until they conform to their average rate ρ.

Only four of the models provide jitter control. Three of the four (all but stop and go) control jitter by the use of a jitter regulator that further restricts the eligibility time. The eligibility time of a packet at node n, E_n, is defined the be the arrival time plus a holding time. The holding time of the disciplines is defined in the table (d_n is the delay bound at node n, $\hat{d}_n$ is the actual delay that occurs at node n, j_n is the jitter

	peak regulator	jitter regulator	scheduler
FIFO	x_{min}		FCFS
Priority queue	x_{min}		static priority
RPQ	x_{min}		EDF
Delay-EDD	x_{min}		EDF
Jitter-EDD	x_{min}	$d_{n-1} - \hat{d}_{n-1} + d_n - j_n$	EDF
RCSP	x_{min}	$E_{n-1} + d_{n-1}$	static priority
Virtual Clock	ρ, I		TDM
PGPS	σ, ρ		WFQ
Stop and Go	ρ, I	I	TDM
Leave in time	σ, ρ	$2(d_{n-1} - \hat{d}_{n-1})$	ρ FCFS[2]

Table 2: Implementation

bound at node n). The goal of all of these holding times is to remove the jitter from the flow.

Stop and go bounds jitter through the size of its frames. Since traffic is regulated so that no more than $\rho * T$ bits are sent during any frame of size T, T represents the interval over which the average rate is calculated. Therefore, T has been replaced in the table by its equivalent in the packet based flow model, the averaging interval I. Since packets are guaranteed to be held for at least one frame length I, and no more than two frame lengths, the jitter bound is always I for all packets of level I service.

The disciplines implement a variety of schedulers. The QoS bounds that are possible within a disciplines are primarily determined by this choice of scheduler. The next section compares these QoS bounds.

3.5 Comparing the Service Discipline Bounds

Table 3 compares the QoS bounds offered by each of the service disciplines for a schedulable set of flows as reported in the literature. The bounds of the packet based

[2] FCFS server of rate ρ

	max end-to-end delay	jitter
FIFO	$H * \sum_{i=0}^{j} S_{max,i}/r_{out}$	
Priority Queue	$H * \sum_{\forall i: p_i \geq p} S_{max,i}/r_{out}$	
RPQ	$H * p\Delta$	
Delay-EDD	$H * \sum_{\forall i: d_i \leq d} S_{max,i}/r_{out}$	
Jitter-EDD	$H * \sum_{\forall i: d_i \leq d} S_{max,i}/r_{out}$	x_{min}
RCSP	$H * \sum_{\forall i: p_i \geq p} S_{max,i}/r_{out}$	x_{min}
Virtual Clock	$\frac{\sigma + 2(H-1)S_{max,i}}{\rho}$	
PGPS	$\frac{\sigma + 2(H-1)S_{max,i}}{\rho}$	
Stop and Go	$2HI$	I
Leave in Time	$\frac{\sigma + 2(H-1)S_{max,i}}{\rho}$	σ/ρ

Table 3: Delay and Jitter Bounds

models in the table assume uniform output link rates and traffic across the hops. This is to simplify comparison with the rate based models which assume uniform service rate and burst size across the hops. The bounds are for a set of j flows. For disciplines in which the delay can differ for different flows sharing the same link, the delay is for flow i. All of the bounds assume either rate control or traffic shapers, however some bounds assume that the peak rate does not exceed the output link rate (FIFO, Priority Queue, Delay EDD, Jitter EDD, RCSP, Stop and Go) and the others assume that the peak rate can exceed the output rate, but the size of this burst is limited (Virtual Clock, PGPS, Leave in time). As such it is difficult to make a direct comparison between the bounds in these two different classes. Since the bounds of scheduler based disciplines are dependent on the characteristics of the competing flows and rate based disciplines are only based on the allocated rate, it is easy to select traffic combinations where one or the other approach will offer lower delay or jitter bounds. The purpose of the comparison is not so much to show which discipline offers the lowest delay bound, but rather to show the types of delay bounds that can be offered and what these bounds

H	number of hops in flow path
j	number of flows
p	priority level
i	flow number
Δ	RPQ rotation time
d	delay
S_{max}	maximum packet size
r_{out}	output link rate
x_{min}	minimum packet interarrival time
σ	maximum burst size
ρ	average rate
I	interval over which average rate is calculated

Table 4: Parameters

are based on. Jitter bounds are only given for the rate controlled disciplines. For disciplines that do not bound jitter, the jitter is equal to the maximum delay.

Table 4 defines the variable parameters used in the equations. In the table, H represents the number of hops in the flow's path. x_{min} is the minimum interarrival time between packets of a flow. I is the frame length for the stop and go frame. p represents the priority level of the flow (the queue number of the rotating priority queue) and Δ represents the minimum rotation time in RPQ. S_{max} represents the maximum packet size of a flow. ρ represents the average flow rate and σ represents the maximum flow burst.

All of the proposed disciplines are non-preemptive. This means that all of the disciplines have a delay component of

$$\sum_{n=1}^{H} S_{max}/r_{out,n} \tag{11}$$

to account for a single lower priority packet that is being sent on the required outgoing link and can not be preempted. Since this value goes to zero as $r_{out,n} \to \infty$ and given

	max end-to-end delay	jitter
FIFO	$H * \sum_{i=0}^{j} S_{max,i}/r_{out}$	
Priority Queue	$H * \sum_{\forall i:p_i \geq p} S_{max,i}/r_{out}$	
RPQ	$H * p\Delta$	
Delay-EDD	$H * \sum_{\forall i:d_i \leq d} S_{max,i}/r_{out}$	
Jitter-EDD	$H * \sum_{\forall i:d_i \leq d} S_{max,i}/r_{out}$	x_{min}
RCSP	$H * \sum_{\forall i:p_i \geq p} S_{max,i}/r_{out}$	x_{min}
Virtual Clock	$I + 2(H-1) * x_{ave}$	
PGPS	$I + 2(H-1) * x_{ave}$	
Stop and Go	$2HI$	I
Leave in Time	$I + 2(H-1) * x_{ave}$	I

Table 5: Delay and Jitter Bounds (Packet Based Model)

that the transmission rate of high speed networks continues to increase, it is reasonable to drop this term. Therefore to simplify the table and comparison, this component has not been included; but should be included when actually determining bounds if fine granularity of delay bounds is important to the application.

Some interesting similarities appear when we substitute the equivalent packet based traffic model values for the rate based traffic model values as defined in section 2.4.2. The result of this substitution is shown in table 5. If the average time between packets equals the time to send a packet on the output link (i.e. $x_{ave} = S_{max}/r_{out}$), then the delay bounds are nearly identical. However, this assumes a bandwidth allocation equal to the average rate. A different rate of service can be chosen in the rate based service resulting in a different delay bound. The rate based discipline bounds of a flow are clearly independent of other flows. This does not imply that a flow can guarantee a small delay bound merely by choosing a high value for ρ without affecting real-time traffic at all. Although the choice will not change the delay bounds of other existing real-time flows, the total rate of all real-time channels is limited by the outgoing link rate. Therefore, choosing a high ρ will increase the minimum delay possible for future flows desiring real-time service. This is also true of choosing small frame sizes for stop and go, since the frame size represents an allocation of bandwidth. Although the scheduler based RPQ, does not show the effects of other sessions in the bounds in the table, the dependence occurs in the schedulability tests to determine whether flow i can be assigned to queue q.

The above comparisons are the best cases for scheduler based disciplines and the general case for rate based disciplines. This might lead one to believe that rate based disciplines would be a better choice if independence of bandwidth and delay were not a primary consideration. For example, this might be a preference in a bandwidth rich environment. However, some recent work has shown that selection of shapers which shape traffic peaks based on the delay requirements can produce bounds equivalent to (and in limited cases better than) GPS schemes [30] for scheduler based disciplines. Both approaches merit further study.

Table 6 shows the per flow buffer requirements at node h needed to prevent packet loss for each of the service disciplines[3] The buffer requirements reveal an additional advantage to controlling jitter. Only in the rate controlled disciplines do the buffer requirements not increase with the number of hops.

Given the different approaches to determining delay, it is difficult to make a one to one comparison between the QoS offered by the rate based and the scheduler based methods. As can be seen from this summary, there is not a clear general best approach. The appropriateness of a discipline is affected by the type of guarantees required by the application and the network resources available. It is not even certain in networks lacking central control that a single service can be assumed end-to-end. Further research into schemes for general analysis of networks with combined heterogeneous scheduling, such as in [92][35] (non-work conserving schedulers) and [31] (work conserving schedulers) is needed. Hybrid nodes that can support multiple service disciplines depending on application/network trade-offs also merit study.

[3] These bounds are those found in the papers but reduced to common terms and assuming a fixed packet size to simplify comparison. For the original notation and for variable length packets, refer to the references in the sections describing each service discipline.

	buffer
FIFO	$\sum_{n=0}^{h} d_n / x_{min}$
Priority Queue	$\sum_{n=0}^{h} d_n / x_{min}$
RPQ	$\sum_{n=0}^{h} p_n * \Delta_n / x_{min}$
Delay-EDD	$\sum_{n=0}^{h} d_n / x_{min}$
Jitter-EDD	$(J_{h-1} + d_h)/x_{min}$
RCSP	$(d_{h-1} + d_h)/x_{min}$
Virtual Clock	$\sigma + 2(h-1) * S_{max}$
PGPS	$\sigma + 2(h-1) * S_{max}$
Stop and Go	$3I * \rho$
Leave in Time	$\sigma + (d_{h-1} + d_h)\rho$

Table 6: Buffer Bounds

So far only the QoS guarantees of the service disciplines have been discussed. However, implementation issues (efficiency, complexity) will also influence the suitability of a service discipline. The following sections identify some of the implementation issues and compare the scheduling disciplines in light of their implementation requirements.

3.5.1 Switch Complexity

The real-time service disciplines all require specialized hardware and/or software that is not necessary in best effort switches. Switches implementing real-time disciplines must regulate the flows and schedule access to the output link. Although weighted fair queuing (WFQ) [21], which does schedule access to the output link, is commonly implemented in existing best effort switches, the access is based on fair allocation and not based on the requirements of the flow[4]. Because access focuses on fair allocation, the existing best effort switches do not regulate the incoming flows. Schedulers other

[4]The existence of WFQ in modern switches is an advantage to PGPS which is based on such a scheduler.

than WFQ are rare in practice, although they do exist in software for some routers. The amount of special purpose hardware/software required by a discipline will affect the cost of implementing switches and therefore the desirability of manufacturing such switches.

With the exception of FIFO, all schemes require multiple buffers for each outgoing link. If the multiple buffers are provided through virtual separation of a shared buffer, then additional buffer management is needed to select the next packet for the output link. For example, EDD scheduling requires a search through the deadlines of queued packets to find the next packet to send, a complexity of O(N), or, alternatively, at the arrival of each packet a search on the queue to find the insertion point of the packet a complexity of O(log N).

The schemes that use a deadline to determine how to order packets for service require the switch to keep track of the arrival time of the last packet on each flow in order to determine the earliest expected arrival time of the new packet (i.e. previous packet arrival + x_{min}). This implies the need for an internal clock. Although it is not crucial for the clocks of different switches within the network to be synchronized, variations in speed may require a pessimistic allocation of input buffering since it would be possible for one switch to send packets faster than the next switch expects.

Bandwidth based schemes also require regulators to control input arrivals. The (σ, ρ) flow specification can be regulated by a leaky bucket regulator [80] by defining a bucket depth of σ and a leak rate of ρ. In a leaky bucket regulator, tokens are generated at a rate of ρ and placed in the bucket. If more than σ tokens accumulate, they are dropped (limiting the maximum number of tokens to σ). As packets arrive, they may be sent as long as there is a token available in the leaky bucket regulating their flow.

For rate controlled disciplines, each switch will need to have sufficient buffering capacity to hold non-eligible packets, even when the switch has nothing to send. However, this buffering is significantly less than that required for non-rate controlled disciplines [87][89] and therefore is not an argument against rate control.

	buffer requirements	bandwidth/delay independence	scheduling complexity	admission control complexity
high	rate allocating	scheduler based	sorted pri. queues	scheduler based
low	rate controlled	rate based	FIFO	rate based

Table 7: Design Trade-offs

In addition to the management of existing traffic, the relative complexity of the calculation of QoS bounds also needs to be considered. The rate allocating disciplines can determine QoS bounds at a node independent of traffic conditions at other nodes, which leads to a simpler admission control. However, this comes at the trade-off of not being able to separate delay and bandwidth allocations. The trade-offs in design are summarized in table 7.

All of the real-time management schemes result in additional switch complexity. As with the delay bounds, the best choice is dependent on the relative importance of each of these characteristics. It is important in designing scheduling disciplines to consider these implementation costs since the additional costs of a discipline may not be justified for the ability to provide a tighter QoS guarantee.

3.5.2 Over-Reservation of Resources

The QoS bounds determined by the real-time scheduling disciplines given in the previous section and background to this thesis are generally quite pessimistic when compared to actual delay distributions. Even when flows are on-off bursty sources which send packets for the longest interval and highest rate possible (given their agreed traffic specifications), the actual bounds seen tend to be significantly lower than that calculated by the equations for the various disciplines [85] [86]. Due to the pessimistic QoS bounds, greater resources are reserved than necessary and flows that could be carried in the network with a given delay bound may be rejected. This is sometimes cited as an argument for probabilistic bounds rather than guaranteed bounds.

In current wide area data networks, non real-time data makes up the majority of traffic. Roughly twenty percent of the flows in a recent measurement of Internet traffic were UDP flows (typically used for real-time applications on the Internet). These flows only made up roughly five percent of of the bytes being sent [79]. Real-time reservations only affect the acceptance of new real-time flows, not non real-time packets. The over reservation of resources can be used to ensure a high probability that non-real-time traffic will be serviced even if all network resources are reserved for real-time traffic. The inefficiency of the real-time traffic reservations does not therefore need to imply inefficiency of the network overall. However, it is desirable to work towards more accurate calculation of resource requirements for real-time reservations, as we should not assume the continuation of the imbalance between real-time and non-real-time application demands.

Guaranteeing delay bounds is tightly linked to reservation of network resources. The smaller the necessary delay bounds, the greater the amount of network resources that must be reserved. As this relationship is not linear, accepting a low rate of delay bound violation may be preferable in order to increase network utilisation. Probabilistic and predictive services have been proposed to allow this trade-off.

Probabilistic service still allows a connection to specify the acceptable level of violation which can be tolerated. Typically this is in the form of a probability less than 100% than QoS bounds will be met. Predictive service scheduling uses current traffic behaviour to predict future behaviour. Predictive service does not provide a guarantee, but such systems can be very useful in applications which can tolerate temporary and/or low levels of packet loss [16] [51]. [45] gives an example of such a predictive service based on measurement of traffic at the node. In this service an admission control based on weighted fair queuing for predictive service determines the initial delay bounds. These bounds are then adjusted based on the actual measured delays seen during a given measurement window. This method is sensitive to correlated traffic however. If incoming flows burst at the same time, delay violations can occur.

Given that WAN traffic has been shown to be auto correlated [65] occasional delay violations are likely.

Probabilistic and predictive service offer a high probability of meeting delay bounds with less management and reservation overhead for real-time flows. Even probabilistic service tends to be pessimistic, but the evidence supports that loosening the QoS bounds is likely to have a significant benefit in freeing resources with a very small increase in probability of violating agreed bounds [13]. Probabilistic and predictive approaches are likely to form a part of the tools used in supporting real-time applications in networks; but for critical applications, efficiency is secondary to performance. Since scheduling for anything other than the worst case can lead to packet loss and delay bound violations, such statistical methods are suited only to soft real-time applications. Guaranteed service is likely to be necessary for critical applications such as telemedicine. Since non-real-time data traffic is likely to continue to be a major part of network traffic, the extra resources required for guaranteeing QoS bounds can still be used by non-real-time traffic that can be dropped in the rare case that congestion does reach the point of affecting real-time connections.

3.6 Conclusions

Many of the design issues discussed in this chapter involve mutually conflicting goals. For example efficient use of the network requires more accurate accounting of the network traffic and therefore more complex admission control and schedulability analysis. It is not possible, nor sensible, to determine a "best" approach to providing real-time flow management. Further research into the usage patterns of real-time applications and the QoS specifications which best map to user requirements are needed to determine which approach to use. As networks and applications develop, the management requirements and the suitability of the various schemes are likely to change as well.

There is however, scope to improve upon the existing real-time disciplines, especially in the accuracy of scheduler based flow management. Current QoS bounds for the worst case delay and worst case buffer requirements for real-time network traffic are based on analysis that makes two assumptions.

1. The switches are rate controlled (i.e. the source traffic pattern is reconstructed at each node) or have traffic shaping (i.e. the traffic arriving at the next node is guaranteed to adhere to a given rate and maximum burst size).

2. Traffic from multiple channels sharing a physical link can arrive simultaneously.

It is beneficial to remove the first assumption to allow for the removal of rate control and shaping hardware/software in the switches. Although this does increase the complexity of determining QoS bounds, the removal of the assumption allows a choice of which trade-off is to be made (complexity in control or in flow admission).

The second assumption simplifies the determination of QoS bounds but it is inaccurate. Packets on a link have an inherent ordering which affects their arrival time and hence the delays experienced. The removal of this assumption allows tighter, less pessimistic QoS bounds to be determined. D. Ferrari [23] first suggested the potential affect of the ordering of packets on links. Parekh and Gallagher [60][61] also note that the delays could be reduced if the approach they term "all greedy", in which all sessions are able to send packets at the same time, were substituted with one in which two sessions served by the same node just before contending for a subsequent node could not be "all greedy". However, neither applied these observations in practice in the formulation of their worst case delay and buffering calculations.

Chapter 4 presents analysis which removes the restriction of the first assumption and reduces the calculated worst case delay and buffer requirements by replacing the second assumption with a more accurate network node model. Although the work presented in the following chapters is presented in the context of a FIFO scheduler, by following the principles outlined in the chapter, the first observation can be extended to cover other scheduler based real-time disciplines and the the removal of the second assumption can be extended to both rate and scheduler based disciplines.

Chapter 4

A More Accurate FIFO Schedulability Analysis for Non-Rate Controlled Traffic Flows

4.1 Introduction

This chapter examines extensions to current schedulability analysis to include two characteristics of network traffic that are typically left unexplored in the literature. First, the potentially desirable characteristic of not requiring rate control or traffic shaping within network switches. This would allow a more simplified switch architecture. However, removing the per-hop control can significantly increase the burstiness of traffic requiring service at a node. As shown in this chapter and in [86][85], this can lead to very pessimistic worst case delays that are significantly higher than actual delays in rate controlled networks.

This chapter shows that the delay bound calculated for a node in a non-controlled network is dependent on the delay bounds calculated at earlier nodes in the path. Due to this dependency, any inaccuracy in the delay calculated at a node will propagate more than additively to further node delay calculations and result in an end-to-end inaccuracy far greater than just the inaccuracy of the single node. Therefore, in nodes that do not perform rate control or traffic shaping, the accuracy of the delay bounds becomes of greater importance.

One characteristic of traffic flowing through a network is that as packets are sent on a link, the packets are ordered and do not arrive simultaneously at the next node. Packets of equal priority transmitted on a given output link after a given packet also of equal priority can not delay the earlier packet at the next node. This serialisation of traffic is not included in current delay analysis in the literature, leading to overly pessimistic delay bounds. Extending analysis to include this characteristic can reduce the pessimism in delay and buffering bounds for both rate controlled and non-rate controlled networks.

This chapter begins with a detailed discussion of the phenomena of serialisation of flows sharing the same output link and the way in which serialisation can be used to reduce the calculated maximum packet backlog at a node. The chapter continues with the per node FIFO analysis of QoS bounds for non-rate controlled traffic considering serialisation. The subsequent section discusses the distortion of a flow's characteristics as it traverses a multihop network and the approaches to capturing this distortion for providing upper bounds on delay at a node. The single hop maximum backlog analysis is then extended to a multiple node model suitable for multi-hop networks. This analysis includes both the effects of serialisation and traffic distortion. The main goal of this chapter is to introduce more accurate analysis for non-rate controlled traffic without traffic shaping.

4.2 Traffic Serialization

Figure 4-1 shows the flow of packets from their entry into the network and through a single node.

At the user-network interface (UNI), traffic from multiple sources (flows 1 to 3) sharing a single physical link is multiplexed. This multiplexing establishes an order on the individual packets on the link. Traditional FIFO queuing analysis of maximum delay is valid in such a node. If J sources are competing for access to an output

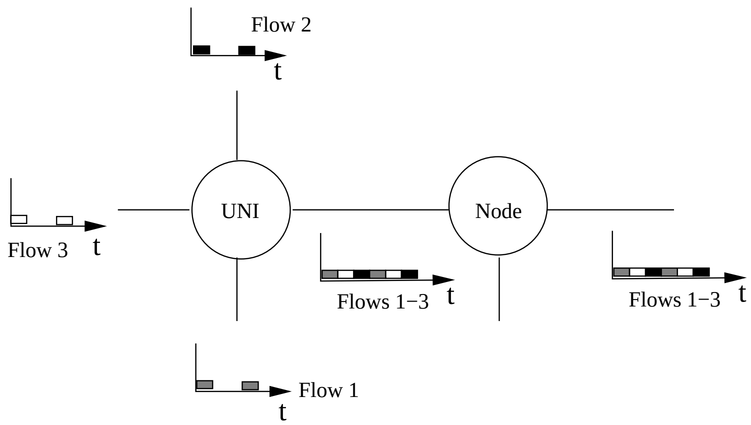

Figure 4-1: Serialization of Channel Traffic

link with rate r_{out}, and source channel j has a maximum packet size of $S_{max,j}$ then if the sum of the source rates is less than the output link rate, the maximum delay experienced by any packet is:

$$d_{max} = \sum_{j=1}^{J} \frac{S_{max,j}}{r_{out}} \tag{12}$$

This delay occurs when all sources attempt to access the link within the same access period. One of the sources, the last to access the link, must wait for the other sources to complete their transmission of a packet and for the transmission of its own packet.

At the next node packets on the link do not arrive simultaneously. Instead they arrive in the order in which they were queued at the previous node. If no cross traffic is present at the node, and the incoming link bandwidth does not exceed the outgoing link bandwidth, no queuing delay is experienced at the node. Traditional FIFO flow analysis fails to consider this effect and calculates an additional delay equal to equation 12 on the assumption that all sources on the incoming link can again have packets that arrive simultaneously.

Even when the input link rate is greater than the output link rate, the delay caused by packets multiplexed on the same link is less than the delay caused by packets from all sources arriving simultaneously. Consider that the first packet will not experience any queuing delay. The second packet, however, will be delayed while waiting for the first packet to be sent. The arrival time of the second packet is later than the first packet due to serialisation. During the time between the arrival of the first packet and the second packet the output link will begin transmitting the first packet. The backlog at the node at the time the second packet arrives will be reduced by the number of bits that the output link was able to transmit before the second packet's arrival. Over a time period Δt, the backlog is equal to the traffic arriving over Δt, (i.e. $A(\Delta t)$), minus the amount of traffic which would be transmitted on the output link. The amount of traffic transmitted on the output link is equal to the output link rate multiplied by the time taken for the traffic to arrive from the input link. For the case of no cross traffic (i.e. only one input link), the backlog can be stated as:

$$b(\Delta t) = A(\Delta t) - \frac{A(\Delta t)}{r_{in}} * r_{out} \tag{13}$$

So the total backlog seen is the total packets arriving multiplied by $1 - r_{out}/r_{jn}$. This reduction in backlog is extended to the more general case with cross traffic in section 4.4.

By considering the effects of serialization in a network, two principles can be used to reduce the pessimism of delay bound calculations.

1. The number of packets which can arrive during a given time interval from a set of channels multiplexed on a single input link is limited not only by the generation rate of the channel sources, but is also limited by the input link rate.

2. In a network with FIFO scheduling, the delay caused by packets arriving on the same link is reduced by the proportion of the output link rate to the input link rate.

4.3 Idle Periods on Input Links

In order to apply the first principle to determine the number of bits that can arrive during a given time interval we must be able to determine the maximum amount of traffic that could have been generated by both the sources (i.e. flows) and the link. Traffic from an input link can arrive at the node at the maximum rate of the link up to the number of bits generated at a particular time. If the packet generation rate exceeds the link speed, packets will be buffered at the prior node until the link becomes available. Therefore, the traffic arriving by time t will be the minimum of the number of bits the link could have generated and the number of bits the flows could have generated by this time. We can define the maximum number of bits arriving on link l generally as:

$$A_l(t) = \min(src\_gen(t), link\_gen(t)) \quad \} \tag{14}$$

where $src\_gen(t)$ is the number of bits which could have been generated by the sources by time t and $link\_gen(t)$ is the number of bits which could have been transmitted by the link by time t.

Although determining the maximum number of bits that the sources can generate in a given time interval is not difficult [1], the maximum number of bits that a link can generate in a given time interval is not merely the rate of the link multiplied by the time (i.e. $r_l * t$). It is possible that during part of the time there were no packets available to be sent and the link was idle. During these idle times the link is not able to generate bits. When determining the maximum number of packets which could be sent by the link at time t, this idle time should be detected and removed from the available transmission time of the link. Leaving the idle time in will not invalidate $A_l(t)$ as an upper bound on the number of bits that could arrive; but will result in the traffic arrival calculation being pessimistic leading to a higher upper delay bound for packets. Therefore, $link\_gen(t) = (t - idle) * r_l$.

We can determine the maximum number of bits transmitted on a link by time t *assuming only a single idle period occurs by t* in the following manner. Assume we have a link of rate r_l with J sporadic sources [2] competing for the link. At time t the number of bits arriving from this link is:

[1] The exact values for $src\_gen(t)$ for the packet based flow model are given in the following sections.

[2] A *sporadic* source is one which has a minimum time between packets but no maximum.

$$
\begin{aligned}
A_l(t) = \min_{\delta=0}^{t} \quad \{ & \\
& \min(src\_gen(0 \rightarrow t-\delta), t-\delta * r_l) \\
& + \min(src\_gen(t-\delta \rightarrow t), \delta * r_l) \\
& \}
\end{aligned}
$$

Intuitively, this equation splits $[0, t]$ into two time periods and states that the number of incoming packets on a link at time t is equal to the minimum of the packets which could be generated by the sources and the packets which could be sent on the link at time t during each time period. The minimum over $\delta \in [0, t]$ is needed in order to choose the two time periods which capture the reduced number of packets that could arrive on a link due to an idle period that occurs before t. $t - \delta$ marks the end of the idle period on the link, so that in the period $[0, t-\delta]$ when the link idle time occurs *src_gen* will correctly limit the number of packets arriving and in the period $[t-\delta, t]$, when no idle link time occurs between transmissions, either the sources or link limits the arrivals based on their generation and transmission capacity.

Since $A_l(t)$ only changes at packet boundaries (partial packets are not eligible to compete for the output link), δ ranges only over the discrete values of possible packet arrival times. Iterating over δ allows us to find the time point where the idle period occurs by determining the time at which the channel generation limits the bits sent.

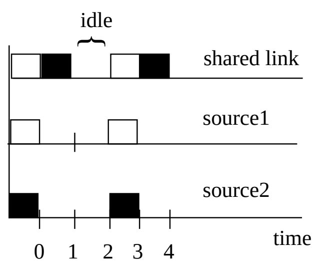

Figure 4-2: Idle Link Periods

Figure 4-2 shows a simple case of two sources sharing a single output link with a transmission rate of one packet per time interval. The sources generate a packet every three time intervals. At time $t = 0$, a packet arrives from each of the two sources. At time $t = 1$, the traffic transmitted on the output link is limited by the link speed (i.e. two packets have arrived, but only one has been sent). At time $t = 3$, the traffic transmitted on the output link is limited by the sources (i.e. 3 packets could have been sent by the link; but only 2 have fully arrived before $t = 3$). At time $t = 4$, 4 packets have arrived and the link could have sent 4 packets. However, the link was idle between time $t = 2$ and time $t = 3$, so it only could have been sending packets for 3 of the 4 time units. By examining all δs and taking the minimum value of $A_l(t)$, we are guaranteed that the idle period will be detected. In our example, $\delta = 1$ (i.e. $t - \delta = 3$) gives the minimum value for $A_l(4)$ with a total of 3 packets being sent on the link at time $t = 4$ (i.e. $A_l(4) = (\min(2, 3) + \min(2, 1)) = 3$, assuming the packet size $S_{max} = 1$ for simplicity).

We can further constrain the values for which we need to calculate $A_l(t)$ by determining the range of δ which is sufficient for detecting the idle period. Consider time $t = 0$ at which all sources begin to generate packets at their maximum rate. Until this initial burst of traffic is transmitted, the number of packets transmitted on the output link is limited by the output link rate. After the burst is sent, the packets transmitted on the output link is limited by the source generation rate. In the period between the sending of the first burst and further generation, the link will be idle. At the time that additional packets are generated, the idle time can be determined by the difference between the packets generated by that time and the packets which could have been sent on the link if it had been busy for the entire period.

At future generation points, the same behaviour holds. If we wish to determine the idle time on the link at time t, we need only find the start of the current busy period. Call the start of the current busy period t'. Since at t' all packets generated before t' must have been sent, the packets sent must be limited by the number of packets generated by the sources by t'. Any remaining capacity of the link beyond that required to send the generated packets is idle. We then only need to calculate any additional idle time between t' and t. A source, j, will generate a packet or packets in this interval if its next generation point at or later than t' is within the interval. We can determine the next generation point by taking the number of generations up to t', rounding up (for the next generation at or after t') and multiplying this number of generations by the interarrival times of packets. This will give us the next packet time after t'. If this time is less than t then a packet will be generated by that source during the interval $[t', t]$. If it is greater, then no packet will be generated in the interval. Therefore we can state that a packet will be generated by source j, during the interval $[t', t]$ if:

$$\lceil \frac{t'}{x_{min,j}} \rceil * x_{min,j} < t \tag{15}$$

To determine $A_l(t)$ we need only examine the values of δ such that $t - \delta$ is the start of a busy period. We will call this domain Δ.

Determining t', the time when the current busy period begins, is complicated by the fact that at any generation point, packets from a previous generation point may still be held over. Therefore, t' can not be assumed to be the next earlier generation point. Earlier generation points which could have generated backlog must also be considered. Figure 4-3 shows an instance of the problem. Four sources with minimum interarrival time, x_{min}, values equal to 6, 6, 6 and 8 are multiplexed on a link.

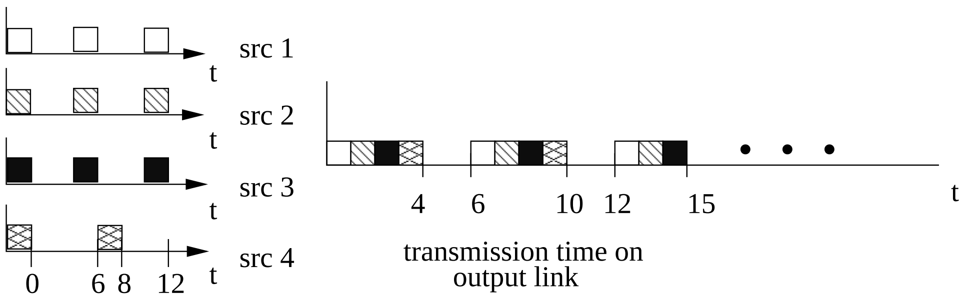

Figure 4-3: Determining the Start of a Busy Period

The output of the link is shown. At time $t = 0$, all sources generate packets creating a burst of four packets which are sent on the output link over the next four time intervals. The link is then idle until the next generation point at time $t = 6$. Three packets are generated at this time, by the three sources with $x_{min} = 6$. These packets are sent over the next three time periods. While these packets are being sent, the source with $x_{min} = 8$ will also generate a packet. If we were to determine the packets sent on the link at time $t = 9$, we cannot merely go back to the last generation point, i.e. $t = 8$, we must consider whether packets generated earlier may still be queued as, for example, the packets generated at time $t = 6$ are.

The first busy period begins at time $t = 0$. It is assumed the link is idle before this time. The beginning of subsequent busy periods can be determined iteratively using Algorithm 1. The domain Δ is defined to be those values of δ for which $t - \delta$ marks the beginning of a busy period.

With the limitations on the range of δ discussed above, we can now define the bits generated on a link of rate r_l with J sporadic sources of minimum interarrival time $x_{min,j}$, at time t as:

$$\begin{aligned} A_l(t) = \min_{\forall \delta \in \Delta} \quad & \{ \\ & \textstyle\sum_{j=1}^{J_l} \lceil \frac{t-\delta}{x_{min,j}} \rceil * S_{max,j} \\ & + \min(\textstyle\sum_{\forall j : \lceil \frac{t-\delta}{x_{min,j}} \rceil * x_{min,j} < t} \lceil \frac{\delta}{x_{min,j}} \rceil * S_{max,j}, \delta * r_l) \\ & \} \end{aligned} \tag{16}$$

The idle time on the link at time t is the difference between $t * r_l$ and $A_l(t)/r_l$.

Existing algorithms for schedulability analysis of the worst case delay and worst case buffer requirements for real-time network traffic have made the assumption that traffic from the multiple channels can arrive simultaneously even in a non-rate controlled network. If this assumption is made then the maximum number of bits generated by a link at some time t is:[3].

$$A_l(t) = \sum_{j=1}^{J_l} \lceil \frac{t}{x_{min,j}} \rceil^+ * S_{max,j} \tag{17}$$

This will be greater than equation 16 except under certain high loads where it may be equal (but never less). By accounting for idle time on links and the limits on traffic arrivals due to link restrictions, we are able to take advantage of a more accurate traffic flow model and calculate a more accurate bound on the traffic volume at a node.

[3] $\lceil x \rceil^+ = \lceil x \rceil$ if $\lceil x \rceil > x$ or $= x + 1$ if $\lceil x \rceil = x$

Algorithm 1 Determining the start of busy periods up to time τ

$NewPackets \leftarrow True$

$BitsGenerated \leftarrow$ the sum of one maximally sized packet per source

$StartTime \leftarrow 0$

$SendingTime \leftarrow BitsGenerated * OutputLinkRate$

while $StartTime < \tau$ **do**

  while $NewPackets = True$ **do**

    $NewPackets \leftarrow False$

    $BitsGenerated \leftarrow 0$

    for all $j \in J$ **do**

      if $\lceil StartTime/x_{min,j} \rceil^{+} * x_{min,j} < SendingTime$ **then**

        $NewPackets \leftarrow True$

      end if

      $BitsGenerated \leftarrow BitsGenerated + \lceil SendingTime/x_{min,j} \rceil^{+} * S_{max,j}$

    end for

    $StartTime \leftarrow SendingTime$

    $SendingTime \leftarrow BitsGenerated * OutputLinkRate$

  end while

  $NextPacketTime \leftarrow \lceil StartTime/x_{min,j} \rceil^{+} * x_{min,j}$ {initialise with any flow $\in J$}

  for all $j \in J$ **do**

    $FlowNextPacket \leftarrow \lceil StartTime/x_{min,j} \rceil^{+} * x_{min,j}$

    if $FlowNextPacket < NextPacketTime$ **then**

      $NextPacketTime \leftarrow FlowNextPacket$

    end if

  end for

  store $NextPacketTime$ {NextPacketTime is the start of the next busy period}

end while

4.3.1 Finding a Node's Maximum Delay, Jitter and Buffer Use

Delay, jitter and buffer use are all dependent on the maximum number of packets which can accumulate at a node. The analysis which follows focuses on determining the maximum backlog in bits at a node. This backlog, b_{max}, is defined as:

$$b_{max} = \max_{t=0}^{\infty}(A(t) - O(t))$$

where $A(t)$ is the total bits generated (arrived at the node) at time t and $O(t)$ is the total bits sent by the output link at time t.[4]

For a FIFO node, the maximum delay at node n can now be defined as the time needed to transmit this backlog:

$$d_{max,n} = b_{max}/r_{out}$$

Since the network is non-rate controlled, the minimum delay occurs when the packet is sent immediately upon arrival. Therefore, the maximum jitter is equal to the maximum delay.

The buffer requirements can be defined as buffering sufficient to hold the maximum backlog:

$$\text{buf}_{max,n} = b_{max}$$

4.4 Per Node Schedulability Analysis

This section derives equations that capture the effects of serialisation to more accurately determine the backlog experienced at time t as multiple flows pass through a single node. In the next section the analysis is extended to determine backlog for traffic which has passed through multiple nodes.

[4] Constraints on the domain for t, including an upper bound by which the maximum backlog will occur, are derived in later sections.

4.4.1 Service Analysis for Combined Peak Channel Rates $\leq$ Output Link Rate

We begin the analysis with the restriction that the sum of the peak rates of all J channels competing for an output link is less than or equal to the rate of the output link.

$$\sum_{j=1}^{J} \frac{S_{max,j}}{x_{min,j}} \leq r_{out}$$

This restriction ensures that once the channels are sending at their maximum rate, the backlog will not increase. Channels are represented by the tuple $\{S_{max}, x_{min}\}$. Any number of channels may be multiplexed on an input link with the restriction that the combined rate of the multiplexed channels must not exceed the capacity of the input link.

4.4.1.1 Sum of Input Link Rates $\leq$ Output Link Rate

The simplest case occurs when the sum of the input link rates is less than or equal to the rate of the output link. The maximum delay occurs when a packet arrives from each of the input links simultaneously. At any time after this instant, the output link will reduce the backlog by an amount greater than or equal to the increase caused by new traffic arriving on the links.

Theorem 1 *Given L input links where each link, l, has J_l channels such that $\sum_{l=1}^{L} r_l \leq r_{out}$, the maximum backlog experienced by any packet is:*

$$b_{max} = \sum_{l=1}^{L} \max_{j=1}^{J_l} S_{max,j} \tag{18}$$

Proof

The maximum rate of input for all links is $\sum_{l=1}^{L} r_l$. Define the time at which the first packets arrive as time 0. Define the maximal size packet that can arrive on a link as $S_{max,l}$ (i.e. $S_{max,l} = \max_{j=1}^{J_l} S_{max,j}$). Assume that at some time t ($t > 0$) the number of packets waiting is greater than L. This implies that:

$$t * \sum_{l=1}^{L} r_l - t * r_{out} > \sum_{l=1}^{L} S_{max,l}$$

$$t(\sum_{l=1}^{L} r_l - r_{out}) > \sum_{l=1}^{L} S_{max,l}$$

$\sum_{l=1}^{L} r_l - r_{out}$ is always negative by the conditions of the theorem. Since packets are of a positive size, the inequality can never hold at any time greater than zero. □

4.4.1.2 Sum of the Input Link Rates > Output Link Rate

If the sum of the input link rates can be greater than the output link rate it is possible that more than one packet may arrive on a single input link during the sending of a single packet on the output link. Since the total channel rate is never greater than the output link rate, the maximum number of packets backlog which could arise from one input link would be equal to the number of channels, J_l, multiplexed on the input link. This backlog will be reduced if the ratio of the input link rate to the output link rate is less than J.

Theorem 2 *Given L input links where each link, l, has J_l channels such that $\sum_{l=1}^{L} r_l > r_{out}$, the number of bits generated for the output link at any time t by the set of input links l is:*

$$A(t) = \sum_{l=1}^{L} \min \left\{ \begin{array}{l} \sum_{j=1}^{J_l} \lceil \frac{t}{x_{min,j}} \rceil^{+} * S_{max,j}, \\ (t - idle_l(t)) * r_l) \end{array} \right\} \tag{19}$$

Proof

At any time t the maximum number of packets which can be generated by a channel, j, is $\lceil t/x_{min,j} \rceil^{+}$. The number of bits generated by the channel is $\lceil t/x_{min,j} \rceil^{+} * S_{max,j}$. For any value of the input link rate, the maximum number of packets can not exceed that generated by the sources. Therefore,

$$A(t) \leq \sum_{l=1}^{L} \sum_{j \in l} \lceil \frac{t}{x_{min,j}} \rceil^{+} * S_{max,j}$$

is the upper limit imposed by the sources.

In the time period t, the input link, l, may send at most $t * r_l$ bits. Some of this time may have been idle as discussed in section 4.2. So the total amount of bits that the link can send in time t is $(t - idle(t)) * r_l$. This is the limit imposed by the link speed. The actual number of bits generated is the smaller of these two limits. □

To determine the maximum backlog we must find the time where the value of the arrival minus the sent packets is maximised. That is we must find the t at which $A(t) - t * r_{out}$ is maximised. To find this the backlog must be calculated over all possible values of t. However, we can reduce this search space by determining the time period during which the maximum backlog can occur. This time is actually bounded by two upper limits.

Theorem 3 *Let there be J channels contending for the same output link. Each channel has the traffic specification of $\{S_{max,j}, x_{min,j}\}$. Given an output rate of r_{out}, if $\sum_{j=1}^{J}(S_{max,j}/x_{min,j}) \leq r_{out}$, The maximum delay will occur by time $t = lcm_{\forall j}(x_{min,j})$.*

Proof

Assume at time t that the output link has not caught up with the input traffic. The lcm($x_{min,j}$) represents the time, t_{regen} at which *all* channels will again generate a packet. t_{regen} is equivalent to time $t = 0$ and the packets generated between time t_{regen} and time $2 * t_{regen}$ will be equal to that generated between time $t = 0$ and t_{regen}. If the backlog has not reached zero during the first period t_{regen}, then the remaining backlog will carry over into the second period. The behaviour will repeat causing an ever increasing backlog at the end of each subsequent period. A continual increase in backlog can only occur when the input rate is greater than the output rate. This contradicts the condition of the theorem. □

Theorem 4 *Let there be J channels contending for the same output link. Each channel has the traffic specification of $\{S_{max,j}, x_{min,j}\}$. Given an output rate of r_{out}, if $\sum_{j=1}^{J}(S_{max,j}/x_{min,j}) \leq r_{out}$, the maximum delay will occur by time*

$$t = \frac{\sum_{j=1}^{J} S_{max,j}}{r_{out} - \sum_{j=1}^{J} S_{max,j}/x_{min,j}} \tag{20}$$

Proof

From equation 19, since the maximum number of packets which are generated by a channel is bounded by $\lceil t/x_{min,j} \rceil^{+}$ (i.e. the link may send fewer but not more packets than the sources generate), the following holds:

$$b(t) \leq \sum_{j=1}^{J} (\lceil t/x_{min,j} \rceil^{+} * S_{max,j}) - t * r_{out}$$

The bound is determined by the time when the backlog is equal to zero. Using the above equation we find the time t such that:

$$b(t) \leq \sum_{j=1}^{J} (\lceil t/x_{min,j} \rceil^{+} * S_{max,j}) - t * r_{out} \leq 0$$

$$\sum_{j=1}^{J} ((t/x_{min,j} + 1) * S_{max,j}) - t * r_{out} \leq 0$$

$$t \sum_{j=1}^{J} S_{max,j}/x_{min,j} + \sum_{j=1}^{J} S_{max,j} - t * r_{out} \leq 0$$

$$\sum_{j=1}^{J} S_{max,j} \leq t(r_{out} - \sum_{j=1}^{J} S_{max,j}/x_{min,j})$$

$$t \geq \frac{\sum_{j=1}^{J} S_{max,j}}{r_{out} - \sum_{j=1}^{J} S_{max,j}/x_{min,j}}$$

□

Note that the bound on the time of the maximum backlog occurrence in theorem 4 is undefined due to a division by zero when the sum of the source input rates equals the output link rate. This occurs because a certain amount of backlog is always present, maintained by the fact that the input and output rates are equal. Therefore, the backlog never reaches zero. In these cases, the least common multiple bound should be used. Either of these bounds on t are sufficient to make the determination of the packets generated computable, and in general when both are defined, the smaller of the two bounds should be used.

4.4.2 Service Analysis for Combined Peak Channel Rates > Output Rate

We now remove the restriction on the channel rates by allowing the sum of the peak rates to exceed the output rate provided the sum of the average rates is not greater than the output link rate. We adopt the channel representation introduced in [91]. A channel is represented by a tuple $\{x_{min}, S_{max}, x_{ave}, I\}$. x_{ave} is the average interarrival time between packets and I is the interval over which this average is calculated.

When the sum of the input link rates is less than the output link rate, the maximum bits generated is the same as shown in section 4.4.1.1. The proof holds since it is only dependent on the input and output link rates and not the rates of the channels.

For the case where the sum of the input rates can be greater than the output rate we have the following theorem:

Theorem 5 *Given a set of links, L, with J_l channels on link l, such that $\sum_{l=1}^{L}\sum_{j=1}^{J_l}(S_{max,j}/x_{min,j}) > r_{out}$ and $\sum_{l=1}^{L}\sum_{j=1}^{J_l}(S_{max,j}/x_{ave,j}) \leq r_{out}$, the maximum number of bits generated by time t is:*

$$A(t) = \sum_{l=1}^{L} \min \left\{ \begin{array}{l} \sum_{j=1}^{J}(\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + (\min\{ \begin{array}{l} \lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \\ \lceil \frac{I_j}{x_{ave,j}} \rceil^+)\} * S_{max,j}, \end{array} \\ (t - idle(t)_l) * r_l \end{array} \right\} \tag{21}$$

Proof

To find the maximum number of bits which can be generated by the sources, time t can be divided into zero or more sections of length I_j and a single section of length less than I_j. By the definition of the channel traffic, in any section of length I_j, at most $\lceil I_j/x_{ave,j} \rceil$ packets will be generated. The number of packets generated during each of these $\lfloor \frac{t}{I_j} \rfloor$ sections of length I_j by channel j is $\lceil \frac{I_j}{x_{ave,j}} \rceil$.

In the remaining time ($t \bmod I_j$) the channel can generate packets at the maximum rate of one packet every $x_{min,j}$ time units. If the number of packets generated at this maximum rate in time $t \bmod I_j$ exceeds the average rate, then only $\lceil I_j/x_{ave,j} \rceil^+$ packets will be generated. Therefore the number of packets generated in the remaining time will be the minimum of the packets generated at the rate of $1/x_{min,j}$ (i.e. $\lceil (t \bmod I_j)/x_{min,j} \rceil^+$) and the bound $\lceil I_j/x_{ave,j} \rceil^+$.

As with equation 19, the idle time of links can be used to determine the limit on bits generated due to the input link rates, $(t - idle(t)_l) * r_l$. The actual bits generated is the minimum of the generation bound and the input link bound. □

We can determine the upper bound on t during which the maximum backlog occurs with the following two theorems.

Theorem 6 *Let there be J channels contending for the same output link. Each channel, j has the traffic specification of $\{S_{max,j},\, x_{min,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out}, if $\sum_{j=1}^{J}(S_{max,j}/x_{min,j}) > r_{out}$ and $\sum_{j=1}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, the maximum delay will occur by time $t = lcm_{\forall j}(I_j)$.*

Proof

The proof is similar to theorem 3. Under worst case conditions (i.e. the maximum traffic generation), all channels will generate packets at the maximum rate until $I_j/x_{ave,j}$ packets have been generated. Call the time immediately before all sources generate packets time zero. The least common multiple of the values of I_j represents the next point at which all channels can again generate packets at the maximum rate. At this point, there can be no packets still waiting to be sent unless $\sum_{j=1}^{J}(S_{max,j}/x_{ave,j}) > r_{out}$ which contradicts the conditions of the theorem. □

Theorem 7 *Let there be J channels contending for the same output link. Each channel has the traffic specification of $\{S_{max,j},\, x_{min,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out}, if $\sum_{j=1}^{J}(S_{max,j}/x_{min,j}) > r_{out}$ and $\sum_{j=1}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, the maximum delay will occur by time*

$$t = \frac{\sum_{j=1}^{J} S_{max,j}(\frac{I_j}{x_{ave,j}} + 1)}{r_{out} - \sum_{j=1}^{J} S_{max,j}(1/x_{ave,j} + 1/I_j)} \qquad (22)$$

Proof

From equation 21, the maximum number of packets generated at time t is bounded by:

$$\sum_{l=1}^{L}[\sum_{j=1}^{J_l}(\lfloor\frac{t}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil+\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}]$$

$\lceil I/x_{ave,j}\rceil^{+}$ will always be greater than or equal to the minimum of $\lceil I/x_{ave,j}\rceil^{+}$ and $\lceil t \bmod I_j/x_{min,j}\rceil^{+}$ so we can make this substitution.

The following therefore holds:

$$\begin{aligned} b(t) \leq & \\ \textstyle\sum_{l=1}^{L} \quad & [\sum_{j=1}^{J} S_{max,j}(\lfloor\frac{t}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil+\lceil\frac{I_j}{x_{ave,j}}\rceil^{+})] \\ & -t*r_{out} \leq 0 \end{aligned}$$

$$\begin{aligned} \textstyle\sum_{l=1}^{L} \quad & [\sum_{j=1}^{J} S_{max,j}(\begin{array}{l} \frac{t}{I_j}*(\frac{I_j}{x_{ave,j}}+1) \\ +\frac{I_j}{x_{ave,j}}+1 \end{array})] \\ & -t*r_{out} \leq 0 \end{aligned}$$

$$\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{t}{x_{ave,j}}+\frac{t}{I_j}+\frac{I_j}{x_{ave,j}}+1)-t*r_{out} \leq 0$$

$$t\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{1}{x_{ave,j}}+\frac{1}{I_j})+\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{I_j}{x_{ave,j}}+1)-t*r_{out} \leq 0$$

$$\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{I_j}{x_{ave,j}}+1) \leq t(r_{out}-\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{1}{x_{ave,j}}+\frac{1}{I_j}))$$

$$t \geq \frac{\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{I_j}{x_{ave,j}}+1)}{r_{out}-\sum_{l=1}^{L}\sum_{j=1}^{J} S_{max,j}(\frac{1}{x_{ave,j}}+\frac{1}{I_j})}$$

The initial backlog will reduce to zero for all values of t greater or equal to the last equation above. We can therefore use the smallest value of t for which the backlog is zero, i.e. when t equals the last equation. $\square$

As discussed after theorem 4, theorem 7 is also undefined for some source input rate values. In theorem 7 division by zero occurs when $\sum_{j=1}^{J} S_{max,j}(1/x_{ave,j} + 1/I_j)$ equals the output link rate. In these cases, the least common multiple bound should be used. When both bounds are valid, the smallest bound should be used.

4.5 Limiting Search Space

Finding the maximum backlog using equations 19 and 21 requires that the backlog be calculated at all time points over a continuous domain up to the bounds calculated in equations 20 and 22. We can reduce the number of data points significantly by observing that the rate of increase (or decrease) of the backlog can only change at times when a link starts (or stops) sending packets. This is illustrated in figures 4-4 and 4-5.

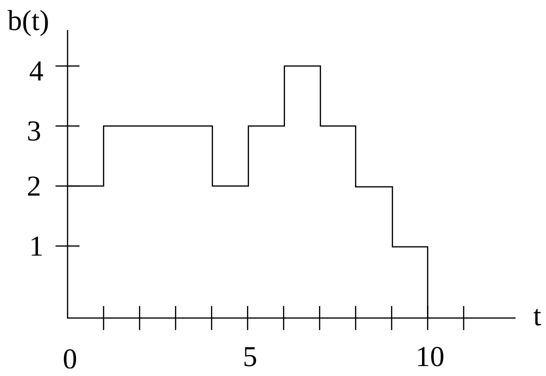

Figure 4-4: Backlog vs. Time

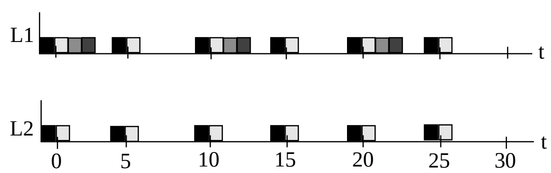

Figure 4-5: Traffic Generated by Links L_0 and L_1

Figure 4-5 shows the arrival of packets over two input links, L_1 and L_2 with rates of one packet per time interval. The output rate, r_{out}, in this example is also one packet per time interval. The two links are generating traffic such that the sum of their peak input rates exceeds the output rate of the node. To maximise the delay, all links send at their maximum rate for as long as possible. Since the sum of the input rates exceeds the output rate the queue will grow steadily, until one of the links temporarily stops generating traffic. When this occurs, the queue length will either continue to grow (if the sum of the remaining links' rates is still greater than the output rate), remain constant (if the sum of the remaining links' rates is equal to the output rate), or decrease (if the sum of the remaining links' rates is less than the output rate). If we examine the backlog at the points at which the queue length decreases, we can be certain that one of these points is a maximum.

Figure 4-4 shows the backlog in packets over time for packets arriving at the node on the two input links and sending on the same output link. As can be seen in figure 4-4, the backlog only decreases at the end of link bursts. The maximum backlog must occur at a time corresponding to one of these decreases. Therefore, we need only check the backlog at times when a link ends a transmission. We can find the end of link bursts using a modified version of Algorithm 1. The only modification needed is to store the $StartTime$ value, which holds the time of the end of the busy period, instead of $NextPacketTime$, which holds the time of the start of the next busy period.

Intuitively one might expect that the maximum backlog will occur when the backlog first begins to decrease. Since the first bursts on the links are the largest which will be generated (due to the worst case scenario), this is not unreasonable. However, it is still possible for the input rate to exceed the output rate over some time interval even after the ends of the first bursts. The example shown in the figures 4-4 and 4-5 illustrates one such case. To assure that the maximum backlog has been found, all times corresponding to the end of a link burst are checked up to the lesser of the bounds shown in theorems 3, 6, 4 and 7.

A similar argument can be made for the detection of the output catching up with the input at the starting point of bursts. If the backlog reaches zero at some point t, the backlog can not increase unless a link begins to generate additional traffic at time t'. The traffic generated will be the same or less than that generated between time zero and t. Therefore, once the backlog reaches zero, no further points need to be examined.

The constraints discussed in this section on the values of t where the maximum backlog and delay will only occur allow us to limit the domain of values for t for which we must calculate the traffic arrival ($A(t)$) and departure ($O(t)$). These constraints apply to both the single node case, already discussed, and the multi-hop case in the next section. To summarize, we can limit the domain of values of t where the maximum backlog occurs to be:

- values that are the end of a link busy period
- values that occur before a backlog of zero is detected

4.6 Traffic Distortion in a Multi-hop Network

Typically in scheduler based real-time networks, the traffic specification at a node is assumed to be the same as the traffic specification at the edge of the network. To accomplish this, rate control is implemented at each node to ensure the traffic adheres to the specification. Without node-by-node rate control, traffic is susceptible to distortion within a network. Even if sources are well behaved[5], cross traffic can cause the traffic to fail to conform to its specification within the network. This effect can be seen in figure 4-6.

At the entrance to the network, the source traffic conforms to a rate of one packet every T time units. At the first node, real-time cross traffic of equal priority is encountered. Packets from all of the incoming links arrive destined for the output link and one is selected to be sent on the output link. In the worst case for the source

[5] Well behaved traffic conforms to a known specification at the source.

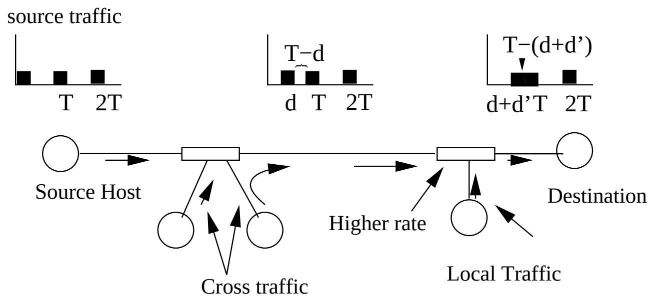

Figure 4-6: Traffic Distortion within a Network

traffic, the packets from the cross traffic will be sent first, forcing the source traffic packet to be queued. While the packet is queued until the outgoing link has sent the packets from the cross traffic, further packets from the source can arrive. These packets may also experience delays due to queuing, but in the worst case for distortion are sent immediately upon arrival. The minimum time between packets is now $T - d$ where d is the amount of time the first packet was delayed. At the next node this effect can occur again adding an additional delay of d', until the source traffic no longer has any spacing between packets and appears as a burst. The result is potentially burstier traffic which no longer meets the original specification of a simple periodic traffic source. If schedulability analysis does not take this distortion into account or the traffic is not restructured to its original specification then this may result in buffer overflow and packet loss at a subsequent node despite the source being well behaved. Ignoring these distortions may also result in an overly optimistic calculated delay and jitter.

Several solutions have been proposed to deal with such traffic distortion at nodes internal to the network for scheduler based flow control. These solutions fall into two main categories: rate controlled (non-work conserving) networks which reconstruct the original traffic pattern at every node and may hold a packet even if the output link is idle [93][92][5][87][23]; or non-rate controlled (work conserving) networks which allow the traffic to be sent from the node as soon as the output link becomes available but do not correct traffic distortion. The non-work conserving, rate controlled, schemes make the task of schedulability analysis simpler since the source traffic characterization can be assumed to be preserved and the same at each node. However, the node itself is made more complex by the need for a rate controller and the network is likely to have lower utilisation when links are left idle with packets being held at a node.

Work conserving schemes make the opposite trade-off: simpler nodes which do not require rate controllers and higher utilisation by not delaying packets when resources are available. These benefits are at the cost of more complex schedulers within the node to select the next packet to be sent [4][2][102] or more complex schedulability analysis which includes the effect of traffic distortion in determining QoS bounds.

In [70] an approach is proposed for a non-rate controlled FIFO network which takes into account the jitter effects in the computation of queuing delay bounds. The results showed the approach was able to provide guaranteed QoS to real-time traffic without rate control or traffic shaping. Removing the need for per node traffic management is at the cost of a higher delay bound. The increase in the bound results from the relationship between the delay bound at a node and the increased burstiness caused by jitter at previous nodes. The delay bound in [70] increases in proportion to the sum of the jitter at previous nodes. This result is highly sensitive to the jitter properties of the traffic. Therefore accurate analysis of the worst case jitter at each node is critical. This accuracy can be improved by refining the scheduling analysis to compute tighter delay bounds by taking into account the serialization of traffic from channels multiplexed on the same link. Such serialization reduces the effective rate of traffic as compared to the traditionally assumed worst case when traffic from individual channels on a link is assumed to arrive simultaneously at each node.

4.7 Multi-hop Delay Bounds for Non-Rate Controlled Real-Time Traffic

Having defined the behaviour of inputs through a single node, where traffic distortion does not apply, this section extends the analysis to the bounds experienced at node n in a multi-hop path. The effects of serialisation are again applied to determine tighter bounds on the packets arriving on a link and the effect of traffic distortion is also included in the bounds.

4.7.1 Service Analysis for Combined Peak Channel Rates $\leq$ Output Link Rate

To simplify analysis, we again begin with the restriction of the peak rates of the combined J input channels to be no more than the rate of the output link.

$$\sum_{j=1}^{J} \frac{S_{\max,j}}{x_{\min,j}} \leq r_{out} \tag{23}$$

In a multihop network, this will inherently restrict the combined peak rate of the channels multiplexed on a single input link to be no greater than the rate of that input link. These two properties ensure that in the worst case, when all sources transmit simultaneously, once all packets from the burst arrive the backlog will not increase further.

4.7.1.1 Sum of Input Link Rates $\leq$ Output Link Rate

Theorem 8 *In a multi-hop network at any node n, where n has L input links, each link l has J_l channels and $\sum_{l=1}^{L} r_l \leq r_{out}$, the maximum backlog experienced by any packet is:*

$$b_{\max} = \sum_{l=1}^{L} \max_{j=1}^{J_l} S_{\max,j} \tag{24}$$

As in theorem 1, since both the total link and source rates does not exceed the outgoing link rate, the backlog is limited by the number of packets which can be simultaneously generated by the link or sources. The proof presented for theorem 1 also holds for this theorem. The backlog experienced is not affected by previous hops.

4.7.1.2 Sum of the Input Link Rates $>$ Output Link Rate

If the restriction in section 4.7.1.1 is removed, multiple packets can arrive from a single input link during the sending of a single packet on the output link. As described in section 4.6, it is possible in a multi-hop path for a source's packets to fail to adhere to the source's traffic specification even if the source is well behaved, due to the influence of cross traffic at earlier nodes. The maximum number of packets which could arrive at a node in the path is restricted to the number of packets generated in the given time period by the channels multiplexed on the link plus the packets which could have been held at prior nodes, due to variations in experienced delay (i.e. jitter).

In the worst case, all of the input links simultaneously produce the longest burst possible. To create the longest burst each prior node should delay the first packet as long as possible. Define the maximum delay at node n to be $d_{max,n}$. Assume the first packet arrives at some time which we will consider to be 0, so that it is sent at $d_{max,n}$. Subsequent packets should be sent at the earliest possible time after the first packet. Due to switching overheads there may be a minimum delay, $d_{min,n}$ experienced by all packets. If not, $d_{min,n} = 0$. A subsequent packet arriving at time t_{arr} should therefore be delayed for $d_{min,n}$ if $t_{arr} + d_{min,n} \geq d_{max,n}$ (i.e. a delay of $d_{min,n}$ will not cause the packet to be sent *before* the first packet) or for $d_{max,n} - t_{arr}$ otherwise (i.e. the packet will be ready for transmission at time $d_{max,n}$).

Any packets arriving during the time period $d_{max,n} - d_{min,n}$ will be ready for transmission at the time the first packet must be sent. The maximum number of packets a channel j can generate during this period is:

$$\lceil \frac{d_{max,n} - d_{min,n}}{x_{min,j}} \rceil \tag{25}$$

Given that $d_{max,n} - t_{arr} \geq d_{min,n}$ whenever the minimum delay experienced is greater than $d_{min,n}$, this will represent the worst case of generated packets.

To maximize the number of packets at the kth node. All of the nodes $n = 1 \cdots k-1$ should delay the first packet $d_{max,n}$ and send all other packets with delay $d_{min,n}$. At the kth node, the first packet has been delayed $\sum_{n=1}^{k-1} d_{max,n}$. The other packets have been delayed $\sum_{n=1}^{k-1} d_{min,n}$. Therefore the maximum number of packets which could have arrived from channel j at the kth node after being delayed at previous nodes is

$$delayed_{j,k} = \lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil \tag{26}$$

Having determined the maximum number of packets which could have been held due to jitter, we now determine the number which could be generated during time t. At time t, the maximum number of packets which can be generated by a channel, j, is:

$$gen_j(t) = \lceil \frac{t}{x_{min,j}} \rceil^{+} \tag{27}$$

The maximum number of packets which can arrive by time t on a link due to a single channel is the number of packets generated by that channel during time t plus those generated earlier but delayed at prior nodes. Therefore the maximum number of *bits* arriving at time t from channel j at the kth node is:

$$A_j(t) = (gen_j(t) + delayed_{j,k}) * S_{max,j} \tag{28}$$

The maximum number of bits generated by all sources on a link l at time t is

$$A_{src}(t) = \sum_{j \in l} A_j(t) \tag{29}$$

This is the upper limit on the bits imposed by the sources.

The maximum number of bits available to be sent from a link, l, is the minimum of the limit imposed by the sources and the limit imposed by the link. As described previously, the idle time of the link should be determined before calculating the limit imposed by the link. One maximally sized packet is included since it could be sent at time 0. The maximum number of bits arriving from link l is therefore:

$$A_l(t) = \min(A_{src}(t), (t - idle(t)) * r_l + S_{max,l}) \tag{30}$$

The number of bits generated at any time t for an output link at the nth node by the set of all input links L is therefore:

$$A(t) = \sum_{l \in L} A_l(t) \tag{31}$$

Equation 30 uses the first principle of serialisation to reduce the bound on the maximum number of bits queued for an output link. The second principle is now applied to further reduce this bound.

The maximum delay experienced by a packet is equal to the difference between the latest departure time of the packet and the earliest arrival time of the packet. The worst case departure time on an output link will occur for the last arriving packet at the point of maximum queue backlog, by definition of FIFO queuing. If the maximum backlog occurs at time t, the packet, p_{last}, on incoming link l with the greatest delay will be the packet arriving at or nearest t.

Packets arriving on l will not delay p_{last} if

1. they arrive after t (by definition of FIFO queuing).

2. $r_{out} \geq r_l$ (due to serialisation).

From equation 30, link l will generate a maximum of $A_l(t)$ bits. p_{last} will therefore arrive at time:

$$t_{arr} = \max(\frac{A_l(t)}{r_l}, t) \tag{32}$$

The first factor is the earliest arrival time due to the input link rate since the link rate may be smaller than the peak source rate, the packet may be queued at the previous node. The second factor is the earliest arrival time based on the source generation rates, i.e. we know that the bits will not be generated before time t, since this is the time the packet generation $A_l(t)$ is calculated from. Whichever of these is later, determines the arrival time of p_{last}.

At the arrival time of this packet the output link will have sent $t_{arr} * r_{out}$ bits. The backlog experienced by this packet will therefore be reduced by this amount. The maximum number of bits which can delay any packet on link l at node n at time t is therefore:

$$b_l(t) = A(t) - t_{arr} * r_{out} \tag{33}$$

Since in the worst case the most time sensitive packet will be p_{last}, equation 33 represents the backlog which should be assumed in determining delay guarantees.

Substituting the values given in this section into equation 33, we arrive at[6]:

[6]In the following equations the idle time of the input link has not been included. This is to simplify the equations. Leaving out the idle time will not invalidate the delay bounds but they may be more pessimistic than if the idle time is included.

$$
\begin{aligned}
b_l(t) = \quad & \textstyle\sum_{\forall l' \in L} \min \begin{pmatrix} \sum_{\forall j \in l'} S_{max,j}(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil + \lceil \frac{t}{x_{min,j}} \rceil^{+}), \\ t * r_l^{'} + S_{max,l'} \end{pmatrix} - r_{out} \\
& * \max \begin{pmatrix} \frac{1}{r_l} \min \begin{pmatrix} \sum_{\forall j \in l} S_{max,j}(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil + \lceil \frac{t}{x_{min,j}} \rceil^{+}), \\ t * r_l + S_{max,l} \end{pmatrix}, \\ t \end{pmatrix}
\end{aligned} \tag{34}
$$

The time t at which the maximum backlog can occur is bounded by equation 34. We want to find the time by which the actual backlog is less than or equal to zero. We can find this time by solving for t where $b_l(t) \leq 0$.

We can substitute:

$$\sum_{\forall j} S_{max,j}(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil + \lceil \frac{t}{x_{min,j}} \rceil^{+})$$

for

$$\sum_{\forall l' \in L} \min(\sum_{\forall j \in l'} S_{max,j}(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil + \lceil \frac{t}{x_{min,j}} \rceil^{+}), t * r_{l'} + S_{max,l'})$$

since the first value will be chosen if it is smaller than $t * r_{l'} + S_{max,l'}$ and if it is larger, then solving for t with this larger value will give us a t bound at least as large as the actual t bound. The same argument can, of course, be made for choosing $t * r_{l'} + S_{max,l'}$, but it is atypical for links to be running at or near 100% capacity. During most time instances, the input links will not be at capacity and the sources will bound the packets. Therefore, the substitution selected will be more accurate for the majority of time instances.

Four cases remain, represented by the remaining max and min choices in equation 34. The upper bounds from each of the four cases are detailed in the following Theorems. The name $delayed_{j,k}$ is substituted for $\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{x_{min,j}} \rceil$ in the following theorems to make them easier to read. Since this value is not dependent on t, it can be separated without affecting the proofs.

Theorem 9 *Let there be L links such that link $l \in L$ has J_l channels. All channels are contending for the same output link at node k. Each channel j has the traffic specification of $\{S_{max,j},\ x_{min,j}\}$. Given an output rate of r_{out} where $\sum_{\forall j \in L}(S_{max,j}/x_{min,j}) \leq r_{out}$, if the packets generated on link l is limited by the sources' generation rate and the arrival time of the last generated packet is* not *limited by the input link rate, the maximum delay for link l will occur by time*

$$t = \frac{\sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1)}{r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j}} \tag{35}$$

Proof

This case equates to choosing the first min value and the second max value.

$$\begin{array}{lcr}
b(t) \leq & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \lceil \frac{t}{x_{min,j}} \rceil^{+}) - t * r_{out} & \leq 0 \\
b(t) \leq & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \frac{t}{x_{min,j}} + 1) - t * r_{out} & \leq 0 \\
b(t) \leq & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) + t \sum_{\forall j \in L} S_{max,j}/x_{min,j} - t * r_{out} & \leq 0 \\
b(t) \leq & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) + t(\sum_{\forall j \in L} S_{max,j}/x_{min,j} - r_{out}) & \leq 0 \\
b(t) \leq & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) \leq t(r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j} & \\
 & t \geq \frac{\sum_{\forall j \in L} S_{max,j}(delayed_{j,k}+1)}{r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j}} &
\end{array}$$

□

Theorem 10 *Let there be L links such that link $l \in L$ has J_l channels. All channels are contending for the same output link at node k. Each channel j has the traffic specification of $\{S_{max,j},\ x_{min,j}\}$. Given an output rate of r_{out} where $\sum_{\forall j \in L}(S_{max,j}/x_{min,j}) \leq r_{out}$, if the packets generated on link l is limited by the sources' generation rate and the arrival time of the last generated packet is limited by the input link rate, the maximum delay on link l will occur by time*

$$t = \frac{\sum_{\forall j \in L} S_{max,j} * delayed_{j,k} + \sum_{\forall j \in L} S_{max,j} - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delayed_{j,k}}{\frac{r_{out}}{r_l} \sum_{\forall j \in l} \frac{S_{max,j}}{x_{min,j}} - \sum_{\forall j \in L} \frac{S_{max,j}}{x_{min,j}}} \tag{36}$$

Proof

This case equates to choosing the first min value and the first max value. The proof is similar to theorem 9

$$
\begin{aligned}
b(t) \leq \quad & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \lceil \tfrac{t}{x_{min,j}} \rceil^{+}) \\
& - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j}(delayed_{j,k} + \lceil \tfrac{t}{x_{min,j}} \rceil^{+}) && \leq 0 \\
b(t) \leq \quad & \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \tfrac{t}{x_{min,j}} + 1) \\
& - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j}(delayed_{j,k} + \tfrac{t}{x_{min,j}}) && \leq 0 \\
b(t) \leq \quad & \sum_{\forall j \in L} S_{max,j} * delayed_{j,k} + t \sum_{\forall j \in L} \tfrac{S_{max,j}}{x_{min,j}} + \sum_{\forall j \in L} S_{max,j} \\
& - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delayed_{j,k} - t \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} \tfrac{S_{max,j}}{x_{min,j}}) && \leq 0 \\
b(t) \leq \quad & t(\sum_{\forall j \in L} \tfrac{S_{max,j}}{x_{min,j}} - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} \tfrac{S_{max,j}}{x_{min,j}}) + \sum_{\forall j \in L} S_{max,j} * delayed_{j,k} + \sum_{\forall j \in L} S_{max,j} \\
& - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delayed_{j,k} && \leq 0 \\
b(t) \leq \quad & \sum_{\forall j \in L} S_{max,j} * delayed_{j,k} + \sum_{\forall j \in L} S_{max,j} - \tfrac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delayed_{j,k} \\
& \leq t(\tfrac{r_{out}}{r_l} \sum_{\forall j \in l} \tfrac{S_{max,j}}{x_{min,j}} - \sum_{\forall j \in L} \tfrac{S_{max,j}}{x_{min,j}}) \\
t \geq \quad & \frac{\sum_{\forall j \in L} S_{max,j} * delayed_{j,k} + \sum_{\forall j \in L} S_{max,j} - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delayed_{j,k}}{\frac{r_{out}}{r_l} \sum_{\forall j \in l} \frac{S_{max,j}}{x_{min,j}} - \sum_{\forall j \in L} \frac{S_{max,j}}{x_{min,j}}}
\end{aligned}
$$

□

Theorem 11 *Let there be L links such that link $l \in L$ has J_l channels. All channels are contending for the same output link at node k. Each channel j has the traffic specification of $\{S_{max,j}, x_{min,j}\}$. Given an output rate of r_{out} where $\sum_{\forall j \in L}(S_{max,j}/x_{min,j}) \leq r_{out}$, if the packets generated on link l is limited by the input link rate and the arrival time of the last generated packet is* not *limited by the input link rate, the maximum delay on link l will occur by time*

$$t = \frac{\sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1)}{r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j}} \tag{37}$$

Proof

This case equates to choosing the second min value and the second max value.

$$b(t) \leq \quad \sum_{\forall j} S_{max,j}(delayed_{j,k} + \lceil \frac{t}{x_{min,j}} \rceil^{+}) - t * r_{out} \quad \leq 0$$

Note that this equation is identical to that in the proof of theorem 9. The proof is therefore the same.

Theorem 12 *Let there be L links such that link $l \in L$ has J_l channels. All channels are contending for the same output link at node k. Each channel j has the traffic specification of $\{S_{max,j},\ x_{min,j}\}$. Given an output rate of r_{out} where $\sum_{\forall j \in L}(S_{max,j}/x_{min,j}) \leq r_{out}$, if the packets generated on link l is limited by the input link rate and the arrival time of the last generated packet is limited by the input link rate, the maximum delay on link l will occur by time*

$$t = \frac{\sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) - S_{max,l} * r_{out}/r_l}{r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j}} \tag{38}$$

Proof

This case equates to choosing the second min value and the first max value.

$$\begin{aligned}
b(t) \leq &\quad \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \lceil \frac{t}{x_{min,j}} \rceil^{+}) - (t * r_l + S_{max,l})/r_l * r_{out} &\leq 0 \\
b(t) \leq &\quad \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + \frac{t}{x_{min,j}} + 1) - (t * r_l + S_{max,l})/r_l * r_{out} &\leq 0 \\
b(t) \leq &\quad \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) + t \sum_{\forall j \in L} S_{max,j}/x_{min,j} - (t + S_{max,l}/r_l) * r_{out} &\leq 0 \\
b(t) \leq &\quad \sum_{\forall j \in L} S_{max,j}(delayed_{j,k} + 1) + t(\sum_{\forall j \in L} S_{max,j}/x_{min,j} - r_{out}) - S_{max,l} * r_{out}/r_l &\leq 0 \\
&\quad t \geq \frac{\sum_{\forall j \in L} S_{max,j}(delayed_{j,k}+1) - S_{max,l} * r_{out}/r_l}{r_{out} - \sum_{\forall j \in L} S_{max,j}/x_{min,j}}
\end{aligned}$$

□

Determining which of the four cases applies requires the value of t to be known; but since this is what we are trying to find, we can not determine which case applies in advance. Therefore, the maximum of the bounds can be used as an upper limit on the time at which the maximum backlog occurs (i.e. it may occur earlier; but we will not have to check times beyond t). Note that the bounds calculated in Theorems 9 and 11 are identical and that the bound calculated in Theorem 12 is strictly less than that calculated in Theorem 9. Therefore the maximum of the two bounds in Theorems 9 and 10 is sufficient as an upper bound on the time by which the maximum backlog will have occurred.

4.8 Service Analysis for Combined Peak Channel Rates > Output Link Rate

Bursty traffic often exhibits peak source rates greater than the output link rate. This traffic can be accommodated if sufficient buffering exists and the average rate over time does not exceed the output link rate. The restriction of section 4.7.1 expressed in equation 23 is now removed to accommodate such traffic.

Channels are represented as in [91] with the tuple $\{x_{\min}, S_{\max}, x_{ave}, I\}$. x_{ave} represents the average rate of the channel and I is the time period over which the average holds. The channels are restricted such that the sum of their average rates does not exceed the output link rate.

$$\sum_{j=1}^{J} \frac{S_{\max,j}}{x_{ave,j}} \leq r_{out} \tag{39}$$

The maximum number of bits which can be generated by the sources during time period t is limited by both the peak rate and the average rate. The total time period during which packets that arrive at time t could have been generated by source j can be divided into several periods of length I_j plus a single period of length less than I_j. During each interval I_j, $\lceil \frac{I_j}{x_{ave,j}} \rceil$ packets can arrive from channel j. There can be at most $\lfloor \frac{t}{I_j} \rfloor$ such intervals. Therefore the number of packets generated by source j during the intervals is

$$\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \tag{40}$$

The time remaining between the last interval and the total time during which packets could be generated is

$$t_{rem} = t \bmod I_j \tag{41}$$

During this time, the channel can send $\lceil\frac{t_{rem}}{x_{min,j}}\rceil^+$ packets, unless this exceeds the average rate guarantee, in which case only $\lceil\frac{I_j}{x_{ave,j}}\rceil^+$ packets can be sent. The number of packets generated during t_{rem} will be the minimum of these two bounds. The total number of packets generated will be the sum of those generated during the intervals plus those generated during t_{rem}:

$$gen_j(t) = \lfloor\frac{t}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil + min(\lceil\frac{t_{rem}}{x_{min,j}}\rceil^+, \lceil\frac{I_j}{x_{ave,j}}\rceil^+) \tag{42}$$

As in section 4.7.1, the maximum number of packets that can arrive by time t on a link due to a single channel is the number of packets generated during time t plus those which could be delayed at previous nodes. As expressed in equation 26, the maximum number of delayed packets at node k is the number of packets that can be generated by the source in the time period:

$$t_{del} = \sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \tag{43}$$

Therefore, the maximum number of packets which could have arrived from channel j at node n after being delayed at nodes prior to n is:

$$delayed_{j,n} = gen_j(t_{del}); \tag{44}$$

Having determined $gen_j(t)$ and $delayed_{j,n}$ equations 28 and 29 can now be applied to determine the maximum number of bits backlog generated by the sources on each input link ($A_{src}(t)$). The maximum number of bits generated by the link will again be the maximum of $A_{src}(t)$ and the link constraint $(t - idle(t)) * r_l + S_{max,l}$.

As in section 4.7.1, packets on the same link have a limited effect on the delay of other packets sharing the link due to serialisation. The backlog experienced by the last packet will therefore be reduced by the packets from the same input link, which are sent before it's arrival time. The worst case backlog for packets on link l at time t is therefore the same as equation 33. The only change is the equations for $delayed_{j,n}$ and $gen_j(t)$. The upper bound on the time t when the maximum backlog will occur can be found in the same manner as theorems 9 through 12.

For completeness, the fully expanded backlog equation for the backlog when the sum of the peak input rates exceeds the output link rate and the sum of the input link rates also exceeds the output link rate is given in Appendix A. The values and proofs of the upper bounds on the time t by which the maximum backlog must have been reached are also given. The algorithm for calculating the maximum backlog is given in Appendix B.

4.9 Summary

In a non-rate controlled network, the backlog experienced at a node is dependent on the backlog experienced at prior nodes. This can be clearly seen from equation 26. Due to this dependency, any inaccuracy in the delay bounds will have an effect not only at the node it occurs at but also on subsequent nodes in the path. This leads to a highly pessimistic value for the backlog and the bounds which are dependent on the maximum backlog. This chapter introduced a method for reducing this overhead significantly by providing analysis which more accurately determines the maximum worst case backlog in non-rate controlled FIFO networks, by considering the effects of serialisation.

The maximum backlog calculated with the equations in this chapter will give the minimum bounds which can be guaranteed. If this delay is reserved, further requests by new channels for resources at the node will be rejected, since the addition of more traffic will necessarily increase the maximum possible backlog, and therefore the QoS bounds, seen by the existing channels. The bounds required at the nodes must be relaxed to meet the actual required bounds of the real-time channel. New channels will then only be rejected if their acceptance will cause a node's QoS bounds to exceed that required by an existing channel. The next chapter develops a connection admission control (CAC) scheme based on the analysis in this chapter that defines the information routers need to support QoS guarantees and how this information is obtained. Chapter 6 then evaluates through simulation the accuracy of the QoS bounds.

Chapter 5

Connection Admission Control

5.1 Introduction

The provision of QoS for bounded delay to a channel involves multiple tasks:

- Determine the end-to-end QoS requirements.
- Translate the end-to-end requirements into per-hop requirements.
- Determine if the per-hop requirements can be met.
  - If they can be met, reserve the necessary resources
  - If they can not be met, signal the host requesting the QoS

The equations in Chapter 4 provide a means for determining if a per-hop delay requirement can be met by calculating the smallest queuing delay that can be offered given the traffic of existing channels at the node and the new channel. It is important to note that this is a *minimum* delay that can be offered at the node under the current traffic conditions and may not represent the delay bound actually required by the new or existing channels. However, we can determine if the new channel can be accepted by ensuring that the minimum delay bound at each node along a channel's path is less than both the per-hop delay requirement of the new channel and the delay bound requirements of the existing channels. If the calculated minimum delay exceeds the delay bound requirements of either the new channel or existing channels, then the new channel can not be accepted. The remaining task is to provide a scheme for translating the end-to-end QoS requirements into per-hop requirements and for establishing channels at nodes in a manner that does not introduce deadlock or livelock. These tasks make up the Connection Admission Control (CAC), which is the subject of this chapter.

The CAC introduced in this chapter addresses how routers isolate traffic flows specific to the analysis in Chapter 4. It can be integrated into any general signaling protocol that supports advertisement of minimum delay bounds on a path, specification of desired QoS and specification of slack for distribution.

5.2 Connection Admission Control

The establishment of a channel should be *receiver initiated*. The QoS the receiver may request is limited by the capabilities of the underlying network and the sender, but only the receiver knows the QoS actually required. The receiver's QoS requirements may be based on factors besides those of receiving the best performance that can be offered by the sender and network. Cost, purpose, local hardware/software limitations, etc. could all be factors in deciding the receiver's preferred QoS.

The connection admission control for channels following the admission tests described in Chapter 4 is performed three phases. The first phase (receiver to sender) sends the channel specification and the QoS requirements to the sender. The second phase (sender to receiver) determines if sufficient resources are available at each hop to meet the requirements and a final phase (receiver to sender) reserves the necessary resources for the channel if it can be accepted. This is illustrated in figure 5-1.

In the first phase (channel specification) the destination node determines the required end-to-end delay bounds and jitter bounds (if required). This information along with the specification for the sender's traffic behaviour is forwarded to the first hop of the path. Note that the sender's traffic behaviour is the arrival rate specified by the receiver, not the arrival rate the sender is capable of producing. The sender will

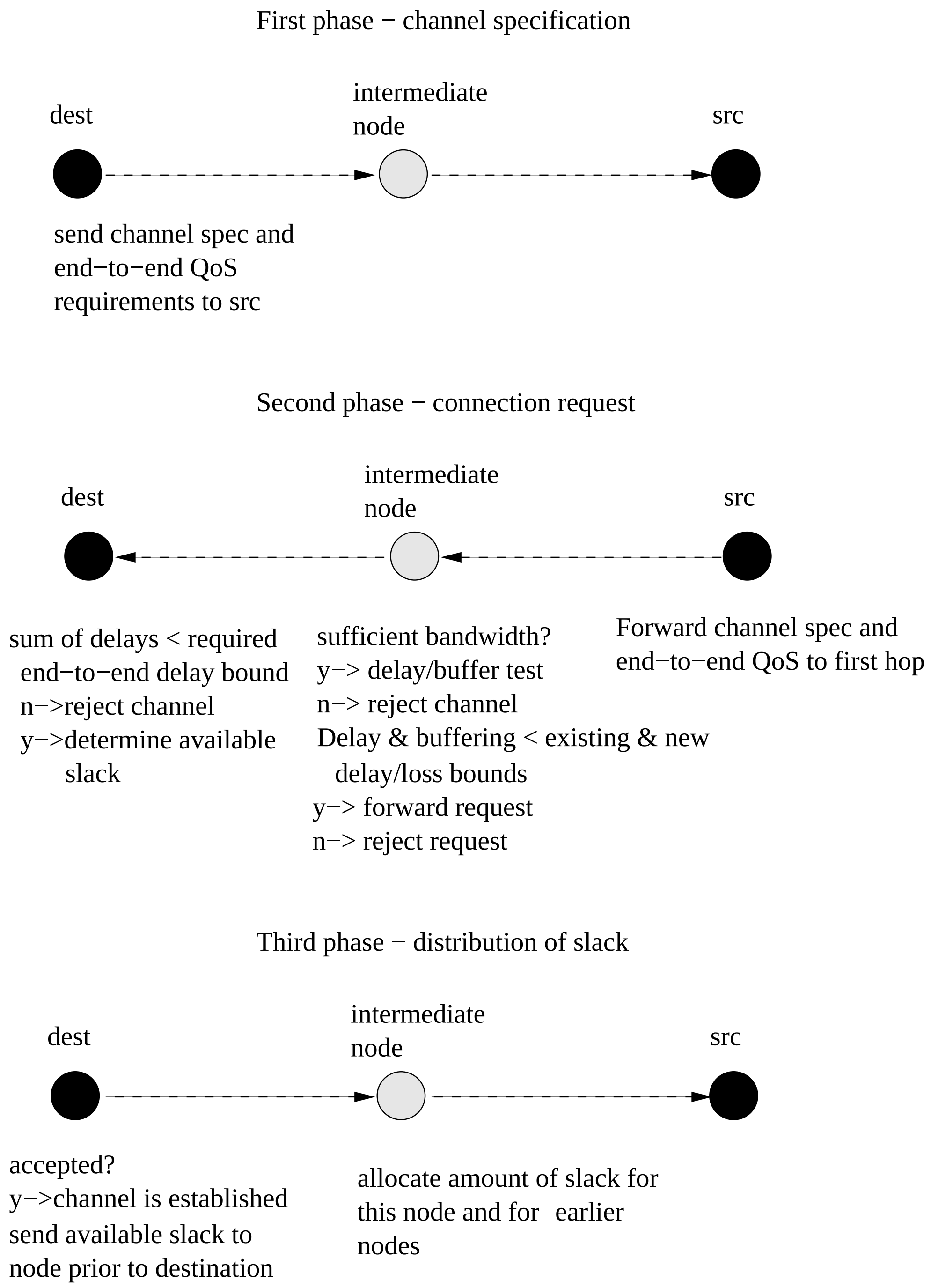

Figure 5-1: 3-phase connection establishment

be required to regulate its traffic to match the specification provided by the receiver. The rate chosen will affect the delay at nodes within the network as higher rates are likely to result in larger worst case queuing and therefore higher delay. Therefore, the receiver should not chose a sending rate greater than the rate the sender is actually capable of sending. Otherwise, the delay bounds calculated will be increased even though the actual traffic could not produce them. The receiver should be aware of the maximum sending rate of the sender. This can be signaled to the receiver before the connection request. In rate controlled networks it is possible to determine the delay bounds based on this traffic specification since the delay at a node is independent of delays at earlier nodes. However, this is not possible if rate control and traffic shaping are removed. Since we do not yet know the delays at the nodes closer to the source, we can not determine the local delay. Therefore the QoS and traffic specification must be forwarded to the source node. Rate controlled networks are able to perform CAC in 2-phases. In such networks this first phase (channel specification) can be combined with the next phase (connection request) since the delay at a node is independent of delay at other nodes.

The second phase (connection request) determines the minimum delay bounds. The source forwards the received QoS and traffic specification to the first node. The delay and jitter bound can be calculated at this node since the jitter at the prior node is known to be zero. The total jitter can be forwarded to the next node so that it also has the jitter information about its earlier nodes needed to calculate local bounds. At the intermediate nodes, the following three tests are performed at each output link along the path:

- Bandwidth test[1]

$$\sum_{j=1}^{J} \frac{S_{\max,j}}{x_{ave,j}} * \frac{1}{r_{out}} \leq 1 \tag{45}$$

- Delay test

$$(\forall l \in L) d_{m,l} = \frac{b_{max,l} + \max \forall j \in l(S_{\max,j})}{r_{out}} \leq \min(\forall j \in l) d_{m,j} \tag{46}$$

[1]Note that for nodes which meet the restrictions of section 4.7.1, $x_{ave,j} = x_{min,j}$.

- Buffer requirement test

$$B_m \geq \max \forall l \in L(b_{max,l}) \tag{47}$$

The bandwidth test is a pre-condition for accepting a new channel. This test checks that the total average incoming traffic rate of all the channels' sharing the same output link does not exceed the total bandwidth of the output link. If this test is not met, the arrival of traffic can continually exceed the sending rate of the output link. This can cause a continually growing backlog of packets, and therefore an unbounded delay. If the bandwidth test fails, there is no reason to continue with further acceptance tests. The channel should be rejected immediately and the node should signal a rejection of the channel back on the path to the sender.

The delay test determines whether the maximum delay, with the new channel added, at the node for the outgoing link L exceeds the local delay bound for any of the existing channels that have delay guarantees. $b_{max,l}$ is the maximum backlog on the outgoing link for channels arriving on link l using the calculations from Chapter 4. $d_{m,j}$ is the upper delay bound acceptable at node m for channel j. One maximum sized packet is added to the maximum backlog to account for a packet currently being sent on the output link (i.e. a packet currently being sent is not preempted). Note that on the forward pass the required per-hop delay bound for the new channel is unknown. However, if at any intermediate node the sum of the minimum delays from each hop exceeds the end-to-end delay bound at any node, then the channel can not be carried and should be rejected.

The buffer requirement test insures that sufficient buffering is available for the output link to hold the maximum possible backlog that can occur at the node. This prevents packet loss due to congestion on guaranteed channels. B_m is the amount of buffer space available for the output link at node m. If input buffering is used then the buffering at the input link l must be at least that of the maximum backlog seen by input link l, $b_{max,l}$

When the destination node is reached the following test is performed:

- Delay bound test

$$D_{sum} = \sum_{m=1}^{M} d_{m,l} \leq D_j \tag{48}$$

The delay bound test sums the per-node delays and tests that this bound is less than the required end-to-end delay bound. A jitter test is not included in detail here; but could be determined by summing the *minimum* delay that could be experienced at each node and testing that the difference between sum of the maximum and the sum of the minimum delays is less than the required jitter bound J_j.

- Jitter test

$$D_{jitter} = D_{sum} - \sum_{m=1}^{M} d_{m,l}^{min} \leq J_j \tag{49}$$

If the bandwidth test or buffer test fails, or the aggregate delay exceeds the maximum end-to-end delay bound at any node or the final end-to-end delay test fails, the channel is rejected.

Note that in the second phase (connection request) the per node delay bound is the upper bound on the delay that could be experienced by packets of the new channel. This bound assumes that only the traffic currently at the node at the time the bound was calculated exists. It does not assume that further traffic might arrive. This delay bound is the smallest delay that can be offered to the new channel by that node given the channels existing at the time of the request. If the new channel requires this delay bound, then no additional channels can be accepted at the node. This occurs since the addition of more traffic will necessarily increase the worst case delay. Due to this effect, the bound on the delay at this node for the new channel should not be set to the current maximum delay at the node. Instead it is necessary to find a value for the channel's delay bound that reflects the actual minimum requirements of the channel rather than the current capability of the network node.

The final phase of the CAC accomplishes this task. When the receiver receives the successful set up request, it compares the total delay, D_{sum} to the requested delay bound, D_j. If D_{sum} is less than D_j, then there is *slack* in the network delay bound, i.e. it is able to more than accommodate the requested delay bound. The slack available in the delay for channel j at the final node f, $\Delta_{f,j}$, is defined as $\Delta_{f,j} = D_j - D_{sum}$. This slack should be redistributed to the nodes along the channel to relax their delay bound for channel j.

In rate controlled networks, it is sufficient to divide the slack over the nodes in any manner such that the sum of all the increases equals $\Delta_{f,j}$. Each node m has a delay bound for channel j equal to $d_{m,l}$ plus the portion of $\Delta_{f,j}$ assigned to node m. However, this is not sufficient for non-rate controlled networks since increasing $d_{m,l}$ increases the jitter at node m. This increase in jitter increases the delay bound not only at node m; but also at nodes between m and the destination. Therefore, the division of the slack must consider the potential delay caused by increased jitter at earlier nodes as well as the increased delay bound at the current node.

Increasing the upper delay bound of $d_{m-1,l}$ by δ_{m-1} will in the worst case increase the backlog at node m by $(\sum_{\forall i \in l} S_{\max,i}/x_{\min,i} * \delta_{m-1} + 1) * (1 - r_{out}/r_l)$ for channels whose peak and average rates are equal or by $(\sum_{\forall i \in l} S_{\max,i}/x_{\min,i} * (\delta_{m-1}/x_{ave,i} + \delta_{m-1}/I_i) + 1) * (1 - r_{out}/r_l)$ for channels whose peak and average rates differ. This can be seen by examining the equations for $b(t)$ in Chapter 4 (equation 34) and Appendix A (equation 75) respectively. The delay at node m will therefore increase by this increased backlog divided by the rate of the output link.

At node $m+1$, $d_{m+1,l}$ will be increased by[2]:

$$
\begin{aligned}
(\textstyle\sum_{\forall i \in l} S_{\max,i}/x_{\min,i} \quad * & \quad [\delta_{m-1} + (\textstyle\sum_{\forall i \in l} S_{\max,i}/x_{\min,i} * \delta_{m-1} + 1) * (1 - r_{out}/r_l)/r_{out}] \\
+ & \quad 1) * (1 - r_{out}/r_l)/r_{out}
\end{aligned}
$$

that is, the increase at node $m+1$ will be the sum of the delay increase at node $m-1$, i.e. δ_{m-1}, and the delay increase at node m due to δ_{m-1}, multiplied by $\sum_{\forall i \in l}(S_{\max,i}/x_{\min,i}) * (1 - r_{out}/r_l)/r_{out}$ In general, the delay increase at some node k will be the sum of the increases at previous nodes multiplied by the constant factor $\sum_{\forall i \in l}(S_{\max}/x_{\min,i}) * (1 - r_{out}/r_l)/r_{out}$. This factor is only constant for a particular input

[2]This is the increase for channels with $x_{ave} = x_{min}$. The delay increase for unequal peak and average rates can be found by substituting $(\delta_{m-1}/x_{ave,i} + \delta_{m-1}/I_i)$ for $\delta_{m-1}/x_{min,i}$. This substitution can be made for the rest of the equations in this chapter.

link and node. To indicate this, we will denote the constant for input link l at node k as $C_{k,l}$. Denote the *total* increase in delay due to slack, δ_k added at the node plus the effect at node k due to changes in the delay at previous nodes as $\Delta_{k,j}$.

At the destination node, f, we can now rewrite the worst case end-to-end delay experienced on channel j as:

$$D_{sum} + \Delta_{f,j} = \sum_{m=1}^{f-1} d_{m,l} + \sum_{k=1}^{f-1} \Delta_{k,j} * C_{f,l} \tag{50}$$

This equation divides the end-to-end delay into two parts: the minimum end-to-end delay (the first summation) and the additional end-to-end delay increase due to the addition of slack (the rest of the equation). Since the amount of slack allocated to the destination node, δ_k is zero (i.e. the destination node does not contribute to queuing delay itself and therefore does not have it's own delay bound), $\Delta_{f,j}$ consists of the summation of the additional delay due to slack at all prior nodes.

Since we allocate some part of the available channel slack to each node, the additional possible delay at some node k for channel j due to the the allocation of slack is the sum of the affect due to allocation of slack at previous nodes plus the slack allocated at node k. We can now write the equation for the additional delay, $\Delta_{k,j}$ more specifically as:

$$\Delta_{k,j} = \delta_k + \sum_{n=1}^{k-1} \Delta_{n,j} * C_{k,l} \tag{51}$$

We use the iterative property of this equation to divide the slack as follows. On the final pass (distribution of slack), at each node k a value for δ_k is chosen. Note that the value of δ_k is bounded by equations 50 and 51. Substituting equation 51 for the value of $\Delta_{f-1,j}$ in equation 50, i.e. the additional delay due to slack at the node immediately prior to the destination, we can rewrite equation 50 as:

$$D_{sum} + \Delta_{f,j} = \sum_{m=1}^{f-1} d_{m,l} + (\delta_{f-1} + \sum_{k=1}^{f-2} \Delta_{k,j} * C_{f-1,l} + \sum_{k=1}^{f-2} \Delta_{k,j}) * C_{f,l} \tag{52}$$

or simplified as:

$$D_{sum} + \Delta_{f,j} = \sum_{m=1}^{f-1} d_{m,l} + (\delta_{f-1} + \sum_{k=1}^{f-2} \Delta_{k,j} * (C_{f-1,l} + 1)) * C_{f,l} \tag{53}$$

$\sum_{k=1}^{f-2} \Delta_{n,j} > 0$ implies $\delta_{f-1} < \Delta_{f,j}/C_{f,l}$. This relationship is intuitive since if we increase the delay at the node prior to the destination by more than the available channel slack, the end-to-end delay can be violated. Once a value for δ_{f-1} has been chosen, the remaining slack available, Δ_{rem}, is $\Delta_{f,j}/C_{f,l} - \delta_{f-1}$. This is the slack that is available that can be attributed to the effect of slack allocation at upstream nodes.

This allocation of slack can continue at the next node, $f-2$. The slack available for the remainder of the nodes is:

$$\sum_{k=1}^{f-2} \Delta_{k,j} = \Delta_{rem}/(C_{f-1,l}+1) \tag{54}$$

Again, we can rewrite the left hand side of the equation as:

$$\Delta_{f-2,j} + \sum_{k=1}^{f-3} \Delta_{k,j} \tag{55}$$

Substituting equation 51 for $\Delta_{f-2,j}$, we arrive at:

$$\delta_{f-2} + \sum_{k=1}^{f-3} \Delta_{k,j} * C_{f-2,l} + \sum_{k=1}^{f-3} \Delta_{k,j} \tag{56}$$

Again, if $\sum_{k=1}^{f-3} \Delta_{k,j} > 0$, then $\delta_{f-2} < \Delta_{rem}/(C_{f-1,l}+1)$. At each node k a value is chosen for δ_k and $\sum_{n=1}^{k-1} \Delta_{n,j}$ that meets the constraints of the remaining slack.

The assigned delay bound for delay due to traffic competing with channel j at each node k is equal to $d_{k,l} + \delta_k$. This is the bound used for scheduling and represents the minimum delay plus slack allocated to node k. Note that this is not the upper bound on the possible experienced delay as the delay from $\sum_{n=1}^{k-1} \Delta_{n,j}$ represents a potential delay increase at node k which the node has no control over (i.e. it can occur by the acceptance of traffic in upstream nodes). The maximum delay that could occur at node k given the reservation is $d_{k,l} + \delta_k + \sum_{n=1}^{k-1} \Delta_{n,j} * C_{k,l}$.

When a new channel request arrives at a node k, the minimum delay bound is calculated as per the equations in Chapter 4. This new minimum delay bound is compared to the assigned delay bound for each channel. If this new value of $d_{k,l}$ is still less than the assigned delay bounds of the other channels which share the link with channel j, then the new channel may be accepted as long as the end-to-end delay requirements are met and the buffer test is still satisfied.

The problem of choosing optimal values for δ_k is an area for further research; but there are several factors to consider in choosing the values. The selection of the individual values of δ_k will effect the ability of the network to carry further traffic. If δ_k is set to zero, no further guaranteed traffic can be accepted at node k. Therefore, it is sensible to choose values of δ_k which are a "fair" proportion of the remaining available slack for the channel so as to leave some slack available to other nodes in the path that have not yet been allocated a portion of the slack. It may also be sensible to choose higher values of δ_n for nodes towards the center of the network with higher numbers of channel requests or for nodes with higher traffic loads.

5.3 Avoiding Deadlock and Livelock

In the 3-phase CAC it is possible for deadlock (a state in which progress in establishing channels is blocked) and livelock (a state in which progress can be made; but channels are never established due to more work being continuously added to establish the channel) to occur. During the second phase of the CAC (connection request), the delay is based on the traffic generated by channels currently at the node plus the new channel (channel A). If the path of a second channel establishment request (channel B) intersects one of the nodes in the path of channel A request, channel B is blocked until the establishment of channel A completes or is rejected. The problem arises because the actual delay requirement of channel A is not yet known since slack has not been distributed. Therefore, we can not allow channel B to be established through the node until the slack for channel A is distributed, or the establishment of channel A is rejected (resulting in a teardown message). Deadlock can occur in at least two ways. The paths of channel A and channel B may share more than one node. It is possible that channel A may request a delay bound first from one of the shared nodes, and channel B make the first request at another shared node. Neither channel A nor B can proceed until the other channel completes phase 3 and allocates its slack, resulting in deadlock. Alternatively, it is possible that the slack distribution or teardown message could be lost, thereby permanently blocking any further channels from being established through nodes in the path of channel A.

Due to the possibility of deadlock, maintaining channel information and reservations in *soft-state* is recommended. Soft-state involves the use of a timeout. If channel reservations are not refreshed (reconfirmed) during this timeout period the state and reservations of the channel are removed from the node. This allows resources to be freed if reservation requests are blocking each other from succeeding. This is the approach proposed for the Internet QoS signaling protocol RSVP [11] [97].

The timeout and re-establishment of channels leads to the possibility of livelock. Both channels' connection requests could time out and they could both again request channel establishment simultaneously. This could repeat indefinitely resulting in livelock. Although this can't be entirely eliminated, the probability of this is likely to be small as the receipt of the rejection of the channels and retry as establishment are unlikely to remain synchronised.

Another option, which is likely to result in fewer retries, is to allow channels that are requesting connections at nodes where a prior request is still awaiting confirmation and slack distribution to complete the second phase conditionally and reject or accept the requests when the prior request completes phase 3 (slack distribution). The new requests can complete the second phase by assuming the prior requests will be accepted and determining the delay for the new request based on the existence of the traffic from the prior requests that are pending. Then if slack distribution indicates that the later request will violate the delay bound of a prior request, the later request will be rejected by the node.

This is clearly pessimistic. If the prior channel is rejected, due to failing end-to-end delay constraints at a subsequent node on the path, it's traffic will still be considered in new requests and may result in a later channel being rejected even if it's delay bounds could be met given the rejection of the prior channel. However, this is likely to result if fewer rejections than the backoff scheme described earlier.

5.4 Conclusions

This chapter presents a connection admission control scheme for the allocation of delay bounds that can be used with the scheduling presented in Chapter 4. Connection admission control in a non-rate controlled channel is more complicated than in rate controlled channels due to the fact that changes in node delay limits can have non-local affects on delay, increasing delay at downstream nodes as well. The scheme presented in this chapter takes these non-local affects into consideration in setting local scheduling delay bounds.

The choice of how the slack is allocated should only affect the network utilisation, since the amount of slack allocated to a node will affect how much additional traffic with delay guarantees it can carry. Optimal selection of per node slack values is an open issue and left as future work, but the purpose of this chapter is to show that it is possible to provide a connection admission control scheme that will work with the analysis presented in Chapter 4.

The primary contribution of this chapter is to define the router admission control and the distribution of slack for the packet based analysis in Chapter 4. The specific signaling presented is just one option and the scheme can be integrated into another signaling protocol that supports determination of path QoS, reservation of desired QoS and slack distribution. RSVP using One Pass with Advertisements (OPWA), to determine the minimum delay bounds on the path, could be used to provide signaling for the described scheme. It is important to note that RSVP has adopted the rate based (σ, ρ) specification for flows, so some modifications in the flow specification of RSVP would be necessary to support packet based flow specification.

Chapter 6

Simulation Results

6.1 Introduction

This chapter compares the queuing delays calculated by the equations in chapter 4 with the queuing delays observed in simulation. The simulations are divided into two groups: single hop simulations and multiple hop simulations. For single hop, the following scenarios are examined:

- Queuing Delay through a Single Hop
  - Total Incoming Channel Peak Rate $\leq$ Output Link Rate
    - Total Input Link Rate $\leq$ Output Link Rate
    - Total Input Link Rate $>$ Output Link Rate
  - Total Incoming Channel Peak Rate $>$ Output Link Rate
    - Total Input Link Rate $\leq$ Output Link Rate
    - Total Input Link Rate $>$ Output Link Rate

In the multiple hop simulations, cases where the total input link rate $\leq$ output link rate are ignored since these cases are trivial extensions of the single hop case. The queuing delay at a hop is not affected by jitter at previous hops in these cases (as discussed in Theorem 4.7.1.1). Therefore, the worst case delay can be determined at each node in isolation. The multiple hop simulations examine different categories of traffic, which fit the various constraints in the multiple hop analysis of chapter 4.

- Queuing Delay through a Multiple Hop Path
  - Constant Bit Rate Traffic:
    - Total Incoming Channel Peak Rate $\leq$ Output Link Rate
    - Total Input Link Rate $>$ Output Link Rate
  - Bursty Traffic:
    - Total Incoming Channel Peak Rate $>$ Output Link Rate
    - Total Input Link Rate $>$ Output Link Rate
  - Variable Bit Rate Traffic:
    - Total Incoming Channel Peak Rate $>$ Output Link Rate
    - Total Input Link Rate $>$ Output Link Rate

The Bursty and Variable Bit Rate Traffic simulations have the same constraints. They differ in that in the Bursty simulations all of the channels send at their maximum rate simultaneously, which represents the worst case traffic arrival. In the Variable Bit Rate simulations the channels can send at any rate up to their maximum specified rate; but they can also send at lower rates and their sending does not have to be correlated. This allows us to compare delays when the worst case traffic arrival patterns are not guaranteed to occur. The results are also compared with rate controlled FIFO delay bounds proposed by Ferrari and Zhang in [91] and non-rate controlled FIFO delay bounds proposed by Pusopa in [70]. The comparisons show that the equations from Chapter 4 compute far more accurate bounds on the worst case delay, even in the absence of rate control.

The REAL network simulator is used for all simulations in this chapter [47][1]. The REAL network simulator is an open source discrete event simulator from Cornell University designed to perform simulations of packet switched networks for various source types and scheduling disciplines.

[1]The REAL simulator is used for exercises in the book [48].

6.2 Single Hop Simulation Results

This section presents simulation studies to support the results of the single node analysis in section 4.4. The single hop simulations simulate multiple channels arriving from multiple input links sharing the same output link at a single node. The simulations are designed to take into consideration the way the REAL simulator represents channels. So that the simulations can be reproduced, the design is described here.

The REAL simulator represents channels as independent source nodes. Since the source nodes are independent in the simulation, we can not directly model the serialisation of packets that would occur for a single source node carrying several channels that originate at that source node (e.g. a workstation with multiple real time streams). Packets from the multiple channels coming from the single source node would be serialised as they competed for the single link connecting the source node to the network. The packets would not arrive over the source nodes link to the network simultaneously. However, if the multiple channels are defined as individual source nodes in the simulator, packets from the different channels can arrive simultaneously at the next node in the simulator.

In order to capture the serialisation affect, the simulation is set up as shown in figure 6-1.

The six edge nodes are source nodes in the simulator and each one represents the source end of a single channel. All of these channels are competing for the same output link from node 2 to node 1. Channels are routed over intermediate nodes to simulate the serialisation of packets. For example, nodes 6 and 7 model two channels. Node 3 models their serialisation at a single source node. Packets from nodes 6 and 7 will not arrive at node 2 simultaneously since serialisation will occur as they pass through node 3. We will refer to nodes that serialise packets from different channels originating from the same source node as serialisation nodes.

The serialisation nodes exist only to allow multiple channels from one source to be accurately modeled. It is important that serialisation nodes do not introduce additional delays to packets. The simulation should behave as if the channels were

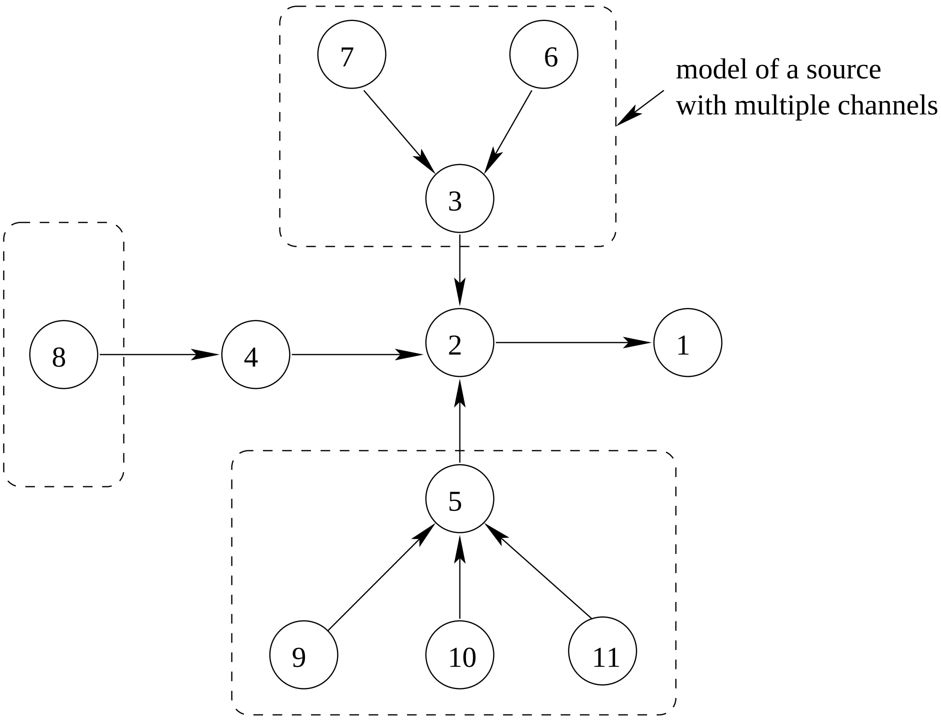

Figure 6-1: Single Hop Simulation

arriving from sources connected directly to node 2. To accomplish this the following conditions were set on the links from the sources to the serialisation nodes:

- The data rate of each link is greater than the rate of the channel it carries. This avoids constraining the packet arrivals from the channels.
- The propagation delay of the link is set to zero.
- The links are all configured to produce the same transmission delay. This ensures that packets sent at the sources of the channels at the same time are available for transmission at nodes 3, 4 and 5 at the same time. This is why node 8 with only one channel also connects through a serialisation node, to ensure that packets from channel 8 will be available for sending at node 4 at the same time as packets from the other channels are available at nodes 3 and 5.

The aim of the simulations is to confirm the accuracy of the equations in Chapter 4. The purpose is not to determine actual delays for particular traffic mixes seen on a particular network. As such the simulation values are chosen to test the equations for different categories of traffic (continuous rate, bursty, etc.) and the number of sources and links is chosen to be sufficient to capture the behaviour that affects the queuing delay predicted by the equations (i.e. serialisation and link rate constraints). The chosen simulation configuration does not imply a limit to the applicability of the equations. The equations and the effects seen in the simulations apply across any point to point topology. However, the absolute values of the queue lengths and the network configurations should not be taken as typical of any particular network (especially the Internet). In the following sections, this configuration, with various link and channel rates is used to test the accuracy of the calculations for each of the single hop cases presented in section 4.4.

6.2.1 Combined Peak Channel Rates Less Than or Equal to the Output Link Rate

Following the order of section 4.4, we begin with simulations that have the constraint that the total incoming traffic rate into node 2 from the sources of the channels must be no greater than the output link rate from node 2 to node 1.

$$\sum_{j=1}^{J} \frac{S_{max,j}}{x_{min,j}} \leq r_{out}$$

The first simulation in this set examines the case of section 4.4.1.1 where the total input link rates are also less than or equal to the output link rate.

$$\sum_{l=1}^{L} r_l \leq r_{out}$$

The simulation is configured as shown in figure 6-2.

The output capacity is divided between the three links entering node 2 and the channel rates are divided so that the sum of the channel rates is no greater than the output link. As discussed in Chapter 4, the input links into node 2 and the channel rates both constrain the maximum number of packets that could arrive at node 2. In order to capture link constraints, nodes 9, 10 and 11 have a combined peak rate

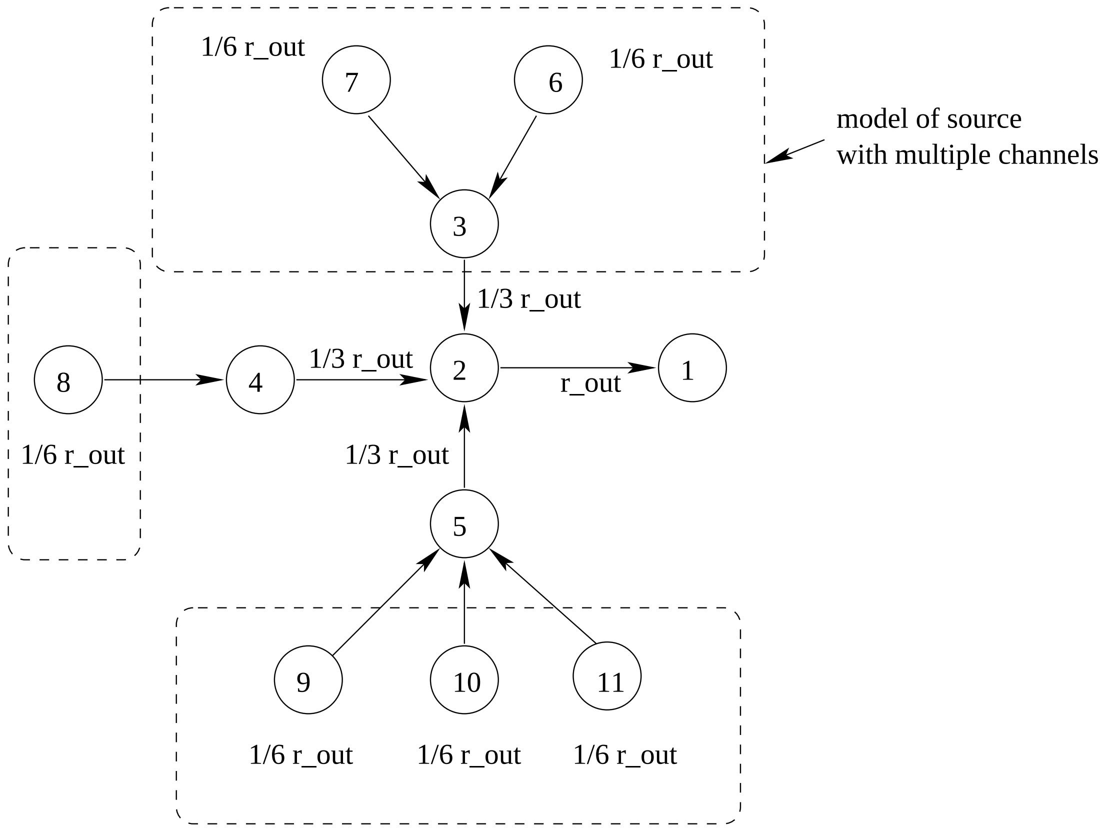

Figure 6-2: Simulation Configuration - Total Channel Rate and Total Input Link Rate Less than Output Link Rate

greater than the rate of the link from node 5 to node 2. In order to capture channel traffic constraints, nodes 7 and 6 have a combined peak rate less than the rate of the link from node 3 to node 2. Node 8 also has a peak rate less than the rate of the link from node 4 to node 2.

Figure 6-3 compares the calculated queue length for Zhang's rate controlled FIFO scheduling (fifo), Pusopa's non-rate controlled FIFO scheduling (nrc-fifo), equation 18 section 4.4.1.1 (eqn) and the observed delay from the simulator (sim) at various load levels.

The calculation from equation 18 predicts a maximum backlog of $\sum_{\forall l} max_{\forall j \in l}(S_{max,j})$. That is, it predicts the worst case is when one maximally sized packet, $S_{max,j}$, arrives simultaneously from each of the three incoming links into node 2. After that point the

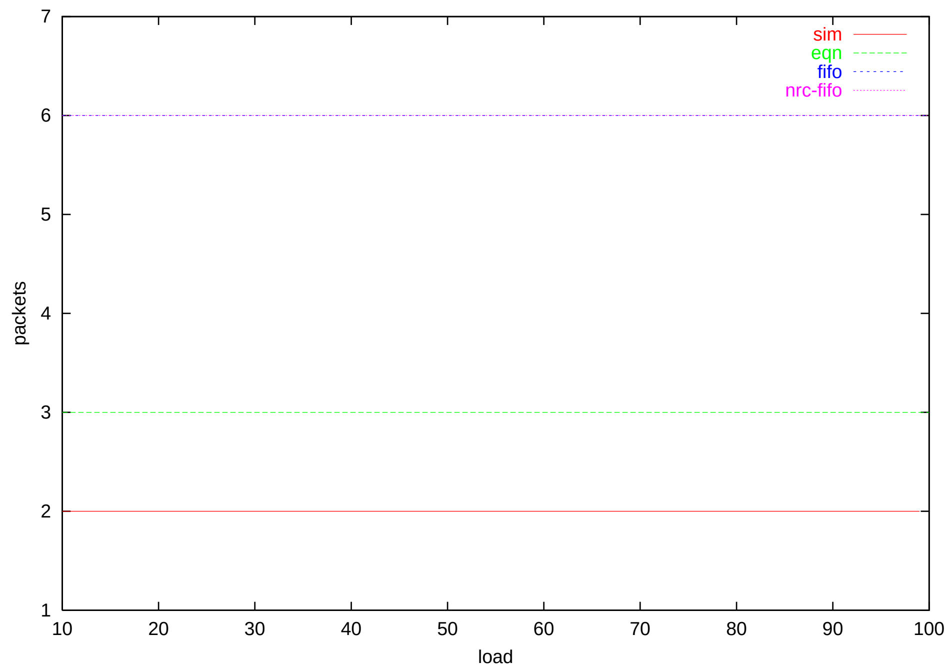

Figure 6-3: Total Channel Rate and Total Input Link Rate no Greater than Output Link Rate

backlog does not increase beyond that value since the total rate of the input links is less than or equal to the output link. Therefore the output queue at node 2 is cleared at a rate greater than or equal to the total input link rates. The comparison with the simulation shows one more packet in the queue in the equation than is seen in the simulation. This occurs because the calculation assumes that the output link may be already busy sending a packet at any time including the worst case arrival time of the packets. In the simulation the output link is idle when the worst case arrival occurs, so one of the three packets is sent immediately leaving a queue of two packets.

The equations of Zhang [91] and Pusopa [70] both calculate backlogs of six packets. The assumption made in Zhang's rate controlled FIFO and Pusopa's non-rate controlled FIFO calculations that each channel can contribute a packet to the queue in the worst case scenario is clearly overly pessimistic and does not consider the effects of serialisation which will prevent packets from arriving simultaneously from all the channels at node 2. Equation 18 accurately predicts the worst case packet arrival as seen in the simulation. Since delay is proportional to backlog, our equations predict a delay which is half of that predicted by Pusopa and Zhang. In addition, the more accurate backlog bound reduces the buffering required. In this example only half of the buffer space would be needed for a router using the equations in this thesis. Although the difference is small in this particular example, in later sections we will see that this difference becomes more significant under traffic where the total channel peak rates can exceed the output link rate.

We now look at the case described in section 4.4.1.2 where the sum of the input link rates exceeds the output link rate.

$$\sum_{l=1}^{L} r_l > r_{out}$$

In Chapter 4, two observations are used to develop equations that more accurately determine delay bounds. The first is serialisation across a link and the second is that the input link rate limits packet arrivals. The previous simulation showed the effect of serialisation with the packet arrival rates determined by the number of input links. Since the output queue never grows above one packet per input link this simulations doesn't clearly show that the minimum of the channel rates or the link rates may also limit the arrival rate of packets at node 2 as given in equation 19. In this simulation we remove the serialisation effect to focus on the effect of the channel and link rates.

The simulation configuration is shown in figure 6-4.

This differs from the previous simulation in that there are no parallel channels from the same source node. Note that intermediate nodes (3, 4 and 5) still exist. This allows us to define channel rates that exceed the input link rates into node 2. The simulator does not allow the peak source node rate to exceed the link rate, even when the average source node rate is less than or equal to the link rate.

The sum of the peak rates of the channels originating at nodes 6, 7 and 8 is equal to the output link rate, r_{out}, from node 2 to node 1. Each of the three channels sends at a rate of $1/3\ r_{out}$. The capacity of the output link from node 2 to node 1 is therefore divided evenly between the three channels. This represents 100% load on the shared output link. The three input links into node 2 are configured such that while each

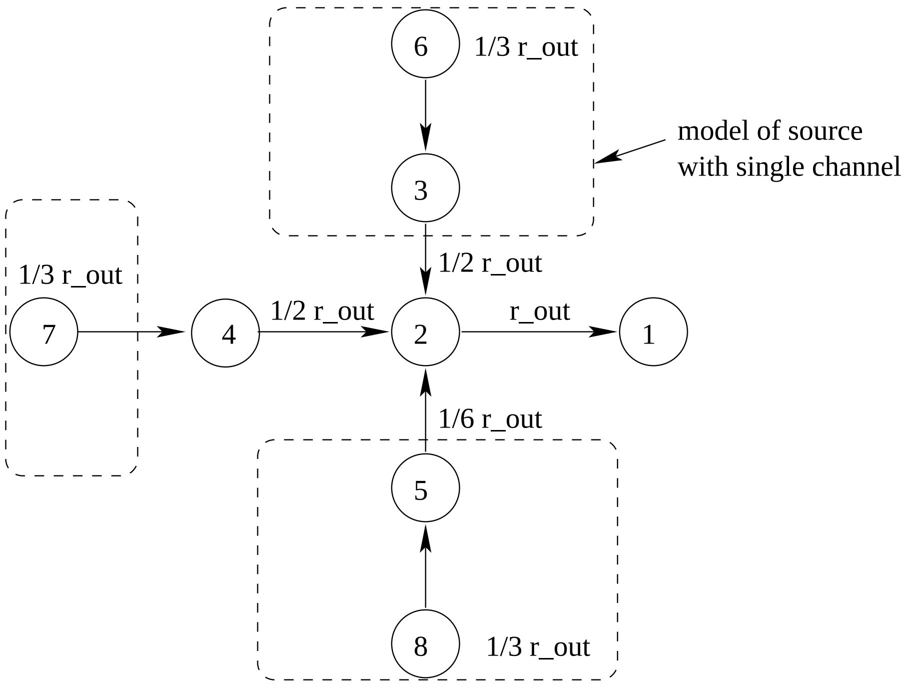

Figure 6-4: Simulation Configuration - Total Channel Rate no Greater than Output Link Rate, Total Input Link Rate Greater than Output Link Rate

input link has a capacity less than r_{out} the sum of their capacities exceeds r_{out}. This allows us to examine the case where the total channel rate is no greater than the output link rate, but the total input link rate is greater than the output link rate. The input links from nodes 3 and 4 to node 2 are 1/2 the output link capacity and the link from node 5 to node 2 is set to equal 1/6 r_{out}. Note that the link capacity from node 5 to node 2 is less than the rate of the channel accessing this link from node 8.

This configuration of the input links allows us to study both the case when the input link limits the packet arrivals to node 2 and the case when the channel rate limits the packet arrivals to node 2. At 100% load, as shown in figure 6-4, the input link rate from node 5 to node 2 is less than the channel rate from node 8 and therefore the link limits the arrival of packets to node 2. The link rates from the other source nodes (3 and 4) are greater than the channel rates from channels 6 and 7, so the channel rates limit the packet arrivals from these sources to node 2.

If we halve all of the channel rates to 1/6 r_{out}, thereby reducing the total load to 50% of the rate of the output link from node 2 to node 1, the channel rate from node 8 and the link rate from node 5 to node 2 become equal and therefore both are placing the same limits on the traffic arrival. At loads less than 50%, the channel rate from node 8 will be less than 1/6 r_{out} and therefore will be less than the link rate from node 5 to node 2. The channel rate of node 8 becomes the limiting factor on packets arriving to node 2. We therefore simulate two cases: 10% load, where packets from node 8 are channel rate limited and 100% load, where packets from node 8 are link rate limited.

The graphs 6-5 and 6-6 show the output queue length at node 2 over time. The total of the channel rates is 10% of the output link capacity in graph 6-5 and 99% of output link capacity in graph 6-6[2]. The graphs 6-5 and 6-6 therefore show the two cases of interest: first where the channel rate limits the input to node 2 and second where the link rate limits the input to node 2.

The effect of the link and channel limitations for the packets arriving on the link from node 5 to node 2 can be seen in both graphs. In the case of 10% load, the link from node 5 to node 2 does not limit the packet arrival rate but it has a greater transmission delay than the other two input links into node 2. This causes the packet from node 5 to arrive slightly after the arrival of the packets from nodes 3 and 4. Therefore, the worst case backlog does not occur with the arrival of the first packets. However, at some point in the future the arrival of a packet from node 5 will coincide with the arrival of packets from nodes 3 and 4. This will happen whenever the least common multiples of the minimum time between packets (i.e. x_{min}) coincide for continuous rate channels. The queue will increase to two packets at these points as seen in the graph. Since the load is less than 100%, the backlog of 2 packets will only occur 10% of the time. In this simulation, the source rates constrain the traffic arrival.

In the case of 99% load, the queue fluctuates between zero and one packet while it is only receiving packets from nodes 3 and 4. Since the link between nodes 5 and 2 has

[2]the simulator experiences problems with the precision of numbers at 100% load so 99% load is used instead.

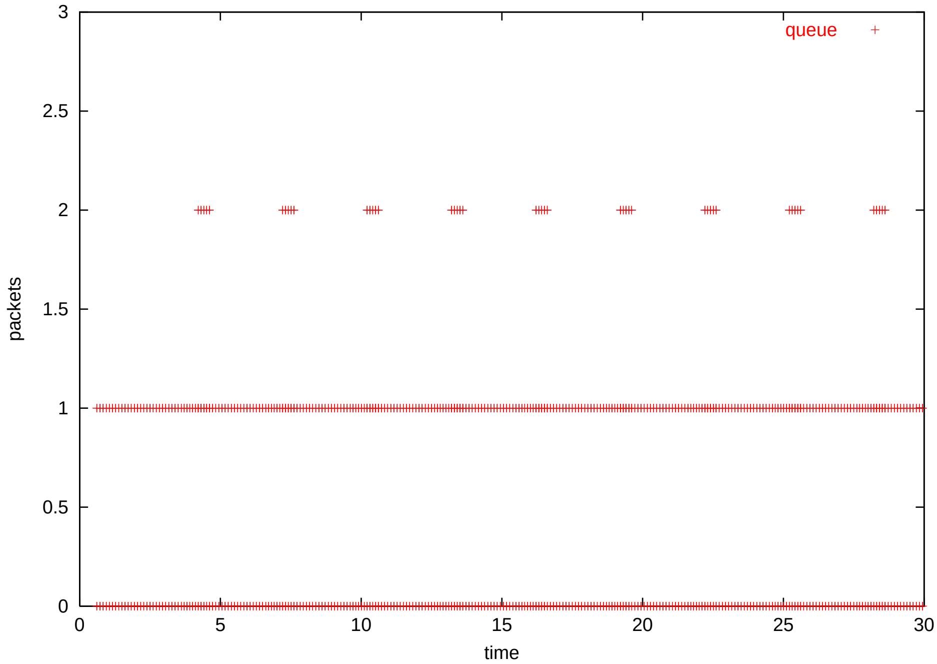

Figure 6-5: Simulation Output -Total Channel Rate Less than Output Link Rate - 10% load

a rate less than its channel source, once the first packet arrives the link will continually produce packets, so there will always be a packet from node 5 at future times. The link from node 5 to node 2 restricts the total incoming traffic rate into node 2 to be less than the output link rate of node 2. Therefore the queue will fluctuate between having zero, one or two packets queued, even though the total incoming source rate is equal to the output link rate.

In both simulations, the worst case number of packets backlogged is the arrival of one packet from each channel. In this case, where no serialisation of channels takes place, Zhang's rate controlled FIFO, Pusopa's non-rate controlled FIFO, the calculations based on the arrival prediction in equation 19 and the simulations all produce the same upper bound of three packets arrived, two packets queued. Although the link rate clearly has an affect on the packet arrival rate in the simulation, it does not affect the worst case backlog in this instance because the combined channel rates are less than the output link rate. Therefore once packets begin to arrive, they will

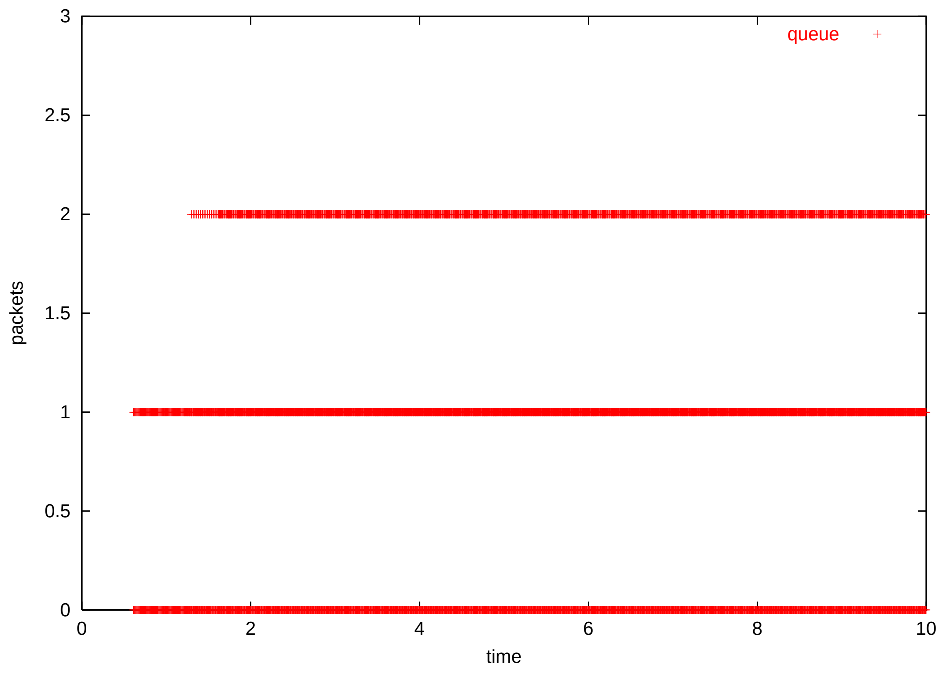

Figure 6-6: Simulation Output - Total Channel Rate Less than Output Link Rate - 99% load

be cleared at a rate as fast or faster than they are generated. However, it is clear that the link rate can change the timing of packet arrivals. In the next section, the effect of this change in timing on the worst case delay is shown.

6.2.2 Combined Peak Channel Rates Greater Than the Output Link Rate

The restriction that the total channel rate is no greater than the output rate is now removed allowing channels to burst at higher peak rates as long as their combined average rate does not exceed the output link rate. The simulations in this section therefore meet the criteria from section 4.4.2 that:

$$\sum_{l=1}^{L}\sum_{j=1}^{J_l}(S_{max,j}/x_{min,j}) > r_{out}$$

$$\sum_{l=1}^{L}\sum_{j=1}^{J_l}(S_{max,j}/x_{ave,j}) \leq r_{out}$$

When the total input link rates are less than the output link rate,

$$\sum_{l=1}^{L} r_l \leq r_{out}$$

the increased burstiness of the channels does not affect the queuing as the links limit the packet arrival rate to be no greater than the output link rate. One packet from each link can arrive simultaneously and after that the queue does not grow. As predicted, the queuing in the simulation and the equations is identical to that shown in the single hop simulation in figure 6-3.

The case where the combined input link rates exceeds the output link rate

$$\sum_{l=1}^{L} r_l > r_{out}$$

shows a greater improvement in accuracy between the equations in this thesis over those existing in the literature. Three studies are presented:

1. A comparison of calculated packet backlog for Zhang's rate controlled FIFO scheduling, Pusopa's non-rate controlled FIFO scheduling, the scheduling method described in section 4.4, and the observed backlog from the simulator at various average load levels. This also shows the effect of different averaging intervals on the calculations.

2. A comparison of the calculated packet backlog and observed backlog versus time under worst case conditions.

3. A comparison of the calculated packet backlog versus time under non worst-case conditions.

The goal is to show the effect of channel and link constraints on packet arrivals. To avoid serialisation influencing the comparison between the the equations from Chapter 4 and those of Zhang and Pusopa, the simulation configuration shown in figure 6-4 is used for these simulations.

For bursty channels the worst case scenario occurs when the channels simultaneously burst for as long as possible (up to the limit of their average rate) and then stop sending until the next interval. This creates the maximum size burst of traffic. The simulations in this section are all configured to produce this behaviour. For the worst case to occur it is also necessary for the packets from the channels to arrive at node 2 simultaneously. To guarantee this the total transmission delays from the source of the channels to node 2 are configured to be equal. To include the effects of both link and channel limitations on packet arrivals, nodes 7 and 8 have rates that are constrained by the channel rates (the links between them and node 2 have a capacity greater than the peak channel generation rate). Node 6 is link constrained. The link from node 3 to node 2 has a rate less than the peak channel generation rate of node 6, so the link will constrain the arrival during bursts. This will spread the burst seen from node 6 over a longer period of time than if it were not link constrained.

6.2.3 Packet Backlog versus Load

The first simulation (figure 6-7) compares the maximum backlog at various average load levels. Equation 21 from chapter 4 (eqn) is compared with the simulator output (sim), those calculated by the equations proposed by Zhang in [91] for rate controlled FIFO networks (fifo) and finally those calculated using the equations proposed by Pusopa in [70] [71] for non-rate controlled FIFO networks (nrc-fifo).

The backlog determined by equation 21 is identical to that seen by the simulator except at two points where the queue length is calculated to be one packet less than that seen by the simulator. An examination of the simulator queues showed that this was due to a rounding error in the program used to calculate the queued packets. A smaller scale is shown in figure 6-8 to show the simulator and equation comparison more clearly.

The simulation shows that the equation is very accurate under worst case conditions. Since the equations proposed in chapter 4 aim to give an *upper bound* on the delay given a particular traffic specification this is the expected and desired result.

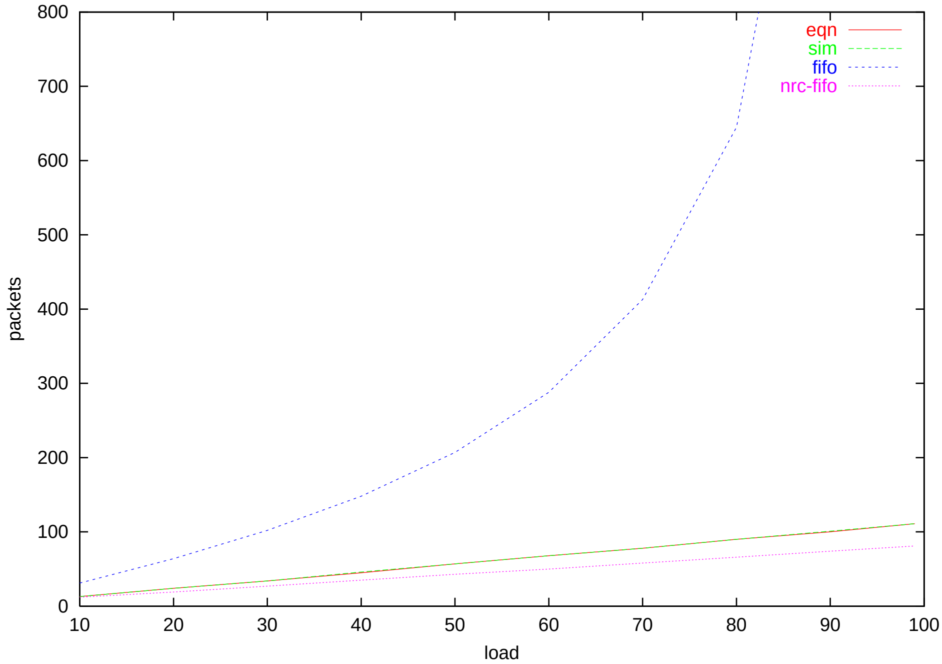

Figure 6-7: Packet Backlog vs Load

Pusopa's non-rate controlled FIFO (nrc-fifo) and Zhang's rate controlled FIFO (fifo) calculations vary significantly from the worst case packet backlog seen in simulation. Pusopa's non-rate controlled FIFO model becomes increasingly overly *optimistic* as the load increases and Zhang's rate controlled FIFO model is overly pessimistic.

The error in Pusopa's non-rate controlled FIFO model occurs because it assumes that the maximum backlog is the summation of the initial bursts minus the packets sent by the output link at the time of the largest burst arrival.

$$B_{max} = \sum_{j=1}^{J} BurstSize_j - t_{arr,maxburst} * r_{out}$$

This assumption is true when *all* channels have a maximum rate that exceeds the output link rate. In cases, such as in the simulation, where only the *total* peak channel rate exceeds the output link rate but the sum of the peak rates of a *subset* of the channels is less than the output link rate, the maximum backlog may earlier than the end of the last burst.

For example, consider figure 6-9 which shows the backlog

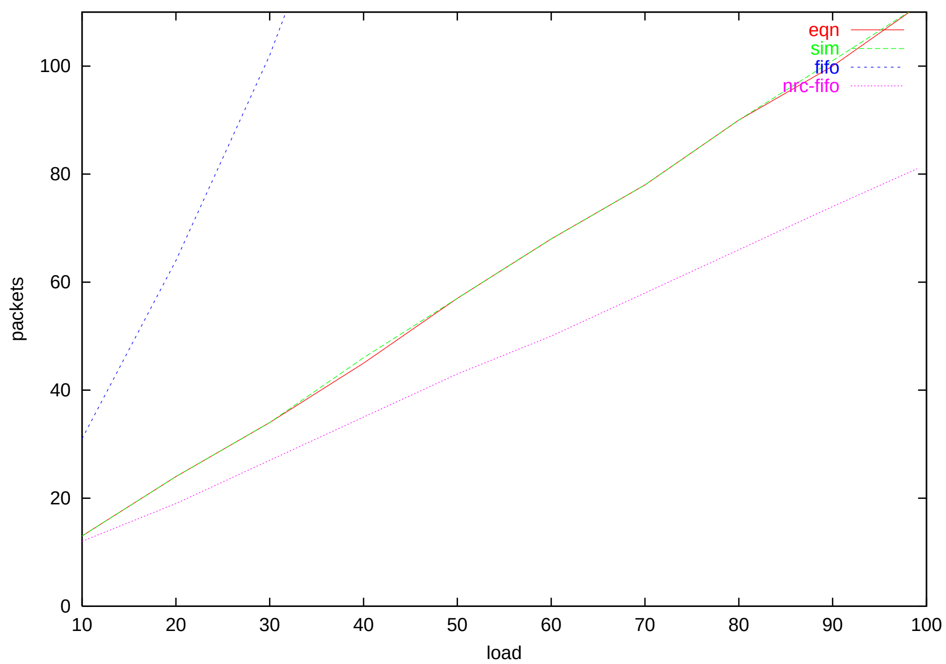

Figure 6-8: Packet Backlog vs Load - Closer View

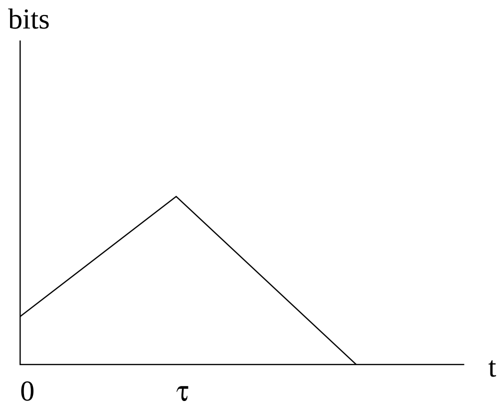

Figure 6-9: Backlog and Channel Bursts

on an output link shared by four channels. Each channel has a peak rate of 1/3 r_{out} so their combined peak rate of 4/3 r_{out} exceeds the output link rate. Call the channel with the longest burst j_B. At time $t = 0$ packets begin arriving from all of the channels and the backlog increases linearly by 1/3 r_{out}. At some time τ two of the channels finish their burst. Neither of these channels is j_B by definition (since their bursts are obviously not the longest). The total input rate is now 2/3 r_{out} and the backlog will begin to *decrease*. Making the assumption that the maximum backlog occurs at the end of the longest burst, which occurs some time after τ, will result in a backlog smaller than the backlog at τ. This is not the worst case backlog and is the reason why the non-rate controlled calculations result in optimistic delay bounds (lower than actually seen in the simulator) in the figures 6-7 and 6-8.

In the case where each individual channel *does* have a peak rate equal or greater than the output link rate, Pusopa's non-rate controlled FIFO model will provide a correct upper bound on the backlog; but the backlog will be overly pessimistic since it fails to consider relative input and output link speeds. It is pessimistic to assume that the bursts will arrive at the channel rate since the input link may also limit the arrival rate of the burst.

Zhang's rate controlled FIFO model also fails to consider relative input and output link rates. However, it is more general in its applicability than the bounds given in Pusopa's non-rate controlled FIFO work as it does not make the assumption that the maximum backlog occurs at the end of the longest burst. As well as accounting for the possible reduction in backlog at the end of a shorter burst as discussed above, Zhang's rate controlled FIFO model also allows for the possibility that one of the channels with a shorter burst duration could burst multiple times within the time of the longest burst. This implies that the maximum backlog is not related solely to the sum of the burst sizes of the channels. A single channel may have more than one burst contributing the maximum backlog.

Consider, for example, three channels with the traffic specifications: $\{x_{min}, x_{ave}, S_{max}, 2I\}$, $\{x_{min}, x_{ave}, S_{max}, 2I\}$, and $\{x_{min}, x_{ave}, S_{max}, 0.5I\}$, whose packet arrival pattern is shown in figure 6-10. All of the channels send packets of maximum size S_{max} at a maximum rate with spacing x_{min} between packets. The first and second channel can send at most $2I/x_{ave}$ packets in the interval $2I$. The third channel can send at most $0.5I/x_{ave}$ packets in the time interval $0.5I$. So channels one and two can send up to four times as many packets during their bursts as channel three can. Given that under worst case conditions all channels try to send as large a burst as possible, the traffic generated by the channels will resemble that shown in figure 6-10. Channels one and two can burst every $2I$ time intervals and channel three can burst every $0.5I$ time intervals. It is clear that when maximising burstiness channel three is not guaranteed to burst just once during the arrival time of the largest burst. In this particular example it will burst three times.

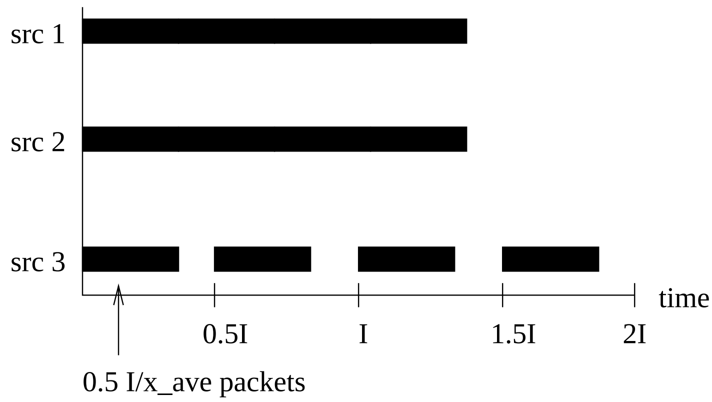

Figure 6-10: Arrivals From Channels with Different Averaging Intervals

This possibility is not considered in Pusopa's non-rate controlled FIFO work, so it is limited to scenarios in which the averaging interval (the interval over which the average rate is guaranteed to hold) of the shorter bursts is greater than or equal to the averaging interval of the longest burst. In the previous example channel 3 would have to have an averaging interval of at least $2I$. This guarantees that the shorter burst channel source will not burst again before the longest burst is finished. The possibility of multiple bursts is included in Zhang's rate controlled FIFO model and contributes to the greater pessimism of the upper bound on the backlog.

Both the Zhang's rate controlled FIFO and Pusopa's non-rate controlled FIFO models use less accurate calculations on the number of packets generated than that used in chapter 4 and do not consider link constraints which compounds the pessimism in the upper bounds on the worst case backlog. As well as producing overly pessimistic worst case delay bounds, the increased worst case backlog increases the buffer space predicted to be necessary to prevent packet loss. This would lead to a larger buffering capacity requirement in routers based on Zhang's or Pusopa's models. Given the limited cases in which Pusopa's non-rate controlled fifo is valid and the exponential growth of the bounds from Zhang's rate controlled FIFO, the equations presented in this thesis are a significant step forward towards providing more accurate worst case backlog predictions and therefore more accurate delay guarantees and reduced buffering requirements.

6.2.4 Packet Backlog versus Time

The simulations thus far have examined maximum backlog versus load. We now look at simulations showing the backlog versus time to show how accurately the equations determine the backlog at a given time under worst case traffic arrival conditions. Since the equations given by Zhang and Pusopa calculate upper bounds and not the backlog at a given time, the plots show only equation 21 and the simulation. The upper delay bounds calculated by Zhang and Pusopa are compared with the maximum delay seen by the simulator in the discussion.

The first two graphs compare the total packet backlog versus time from equation 21 with that seen by the simulator for the simulation at the start of this chapter (figure 6-7). As well as showing the accuracy of the equation, they also show the effect of load on the bounding of the time by which the maximum backlog occurs. Two load levels are examined. The first simulation is run at 99% load (figure 6-11). The backlog closely tracks that of the simulation. In this instance the analysis from chapter 4 determines that the maximum backlog is reached by time one (the least common multiple of the averaging intervals in this scenario) and calculations stop at this point.

At lower loads the upper bound on the time by which the maximum backlog occurs is typically smaller, as shown in figure 6-12.

In this scenario the upper bound on the time when the maximum backlog occurs is that calculated by the bound in Equation 22 in Chapter 4. Such bounds are typically

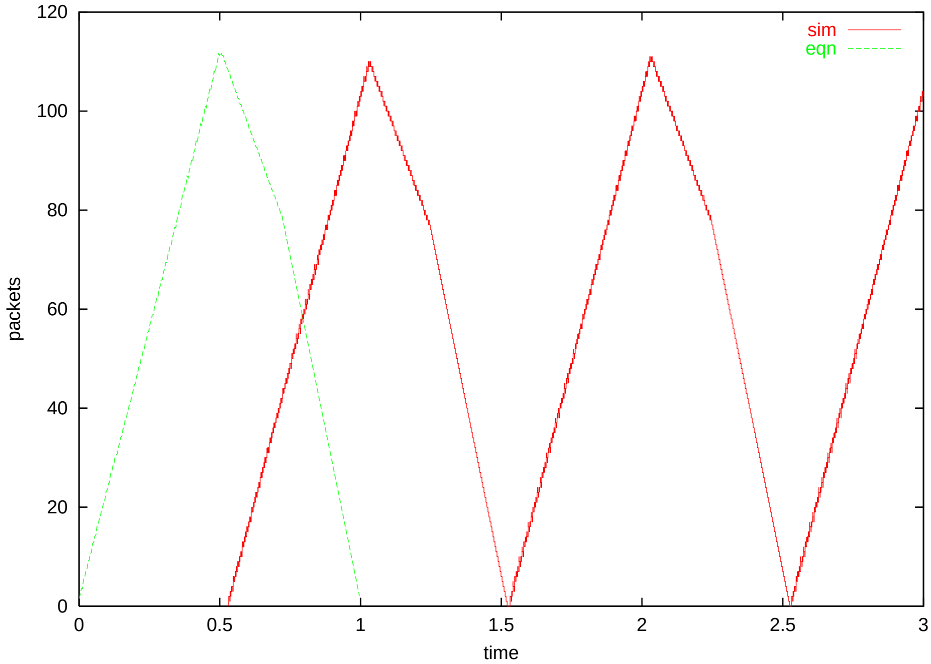

Figure 6-11: Backlog vs Time at 99% load

significantly less than the least common multiple of the averaging interval. When this bound can be used the maximum backlogs can be determined with fewer time points examined.

The next simulation examines the accuracy of equation 21 in calculating the packet backlog over time when channels are able to burst multiple times before the maximum backlog is reached. This simulation follows the example of figure 6-10. This results in a more complex arrival pattern than the previous simulation. In this simulation the channels have different averaging intervals. The averaging interval of channel 7 is set to 0.5 and channels 6 and 8 have intervals of 2.0. This corresponds to the case discussed in the last section and illustrated in figure 6-10.

Figure 6-13 plots the packet backlog versus time calculated by equation 21(eqn) and seen by the simulator (sim). The simulation uses the network configuration shown in figure 6-4. Channel 7 is able to burst three times within the single bursts of channels 6 and 8. The combined peak rate of channels 6 and 8 is less than the output link rate, so when channel 7 finishes its burst, the backlog begins to fall, but it climbs again when

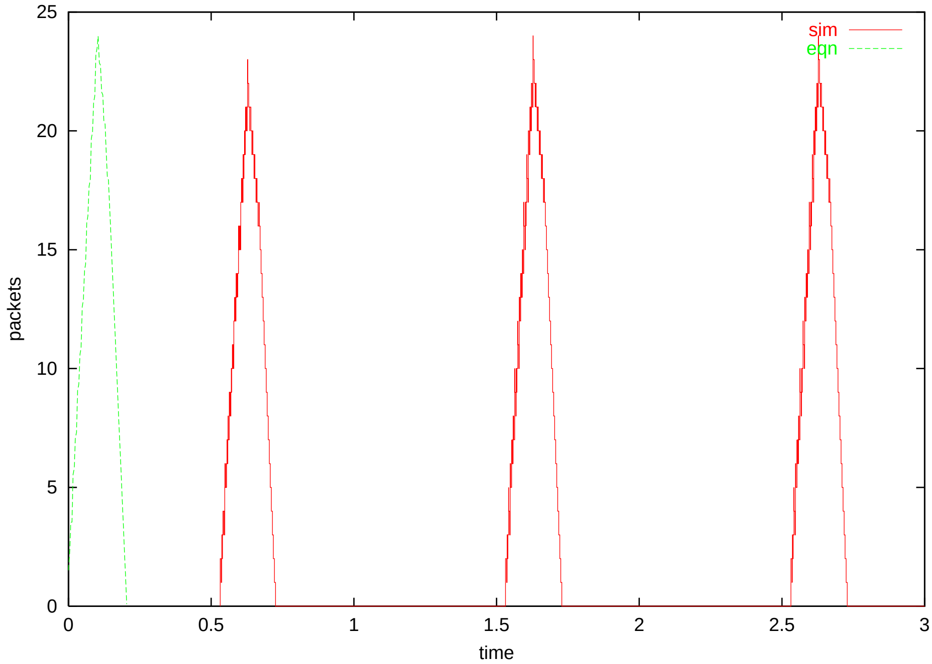

Figure 6-12: Backlog vs Time at 20% Load

channel 7 begins another burst. The calculation from chapter 4 over time closely tracks the packet backlog seen in the simulator. The offset of the simulation values occurs because of the transmission delay from the channels to the intermediate nodes (nodes 3, 4 and 5) and a delayed start time of the source nodes (required by the simulator). Zhang's work calculates a maximum backlog of 2,432 packets for this scenario which is clearly pessimistic. Pusopa's work calculates a negative backlog as this is a case in which the maximum backlog does not occur at the end of the longest burst with one burst from each channel. In this particular simulation, the longest burst finishes at time 1.44. It is clear that this is not the point of maximum backlog and also that several bursts from node 7 will have occurred before this point.

All of the simulations in this section indicate that under worst case conditions, equation 21 is very accurate for determining observed loads and changes in the rate of backlog growth or decay.

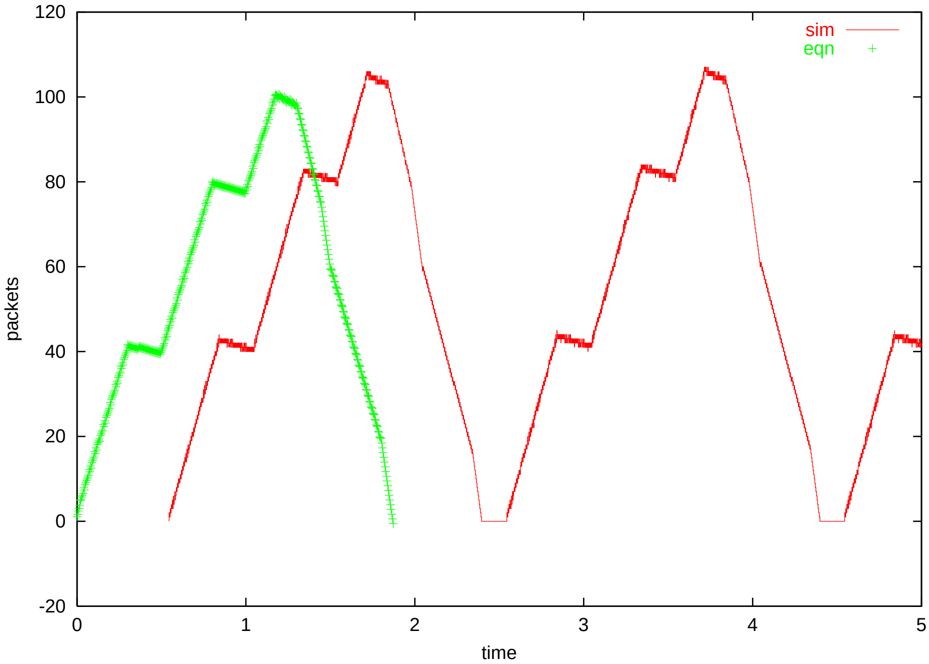

Figure 6-13: Channels with Different Averaging Intervals

6.2.5 Backlog Under Non-Worst Case Conditions

It is more probable that the maximum link bursts will not be synchronized. Under these conditions, the backlog will be lower than in the worst case. Although the aim of this thesis is to provide upper bounds on the worst case, it is interesting to compare how much worse the worst case is to a more average case. Figure 6-14 compares the maximum backlog under worst case traffic arrival calculated by equation 21 versus the maximum backlog seen by the simulator for average traffic arrival under the same load densities. In the simulation, channels send at the average rate over the simulation time and burst up to the peak rate. The sending rate at any particular time is generated randomly between these values. Each load level was run ten times and the largest backlog seen in any of the ten simulations is shown in the figure. As expected the predicted worst case backlog is greater than the backlog seen in the simulator, with the difference growing as the load increases. This indicates that as loads increase, it may be very difficult to accept any new real-time channels even when actual delays experienced are not outside the requirements of the channels. The upper bounds are

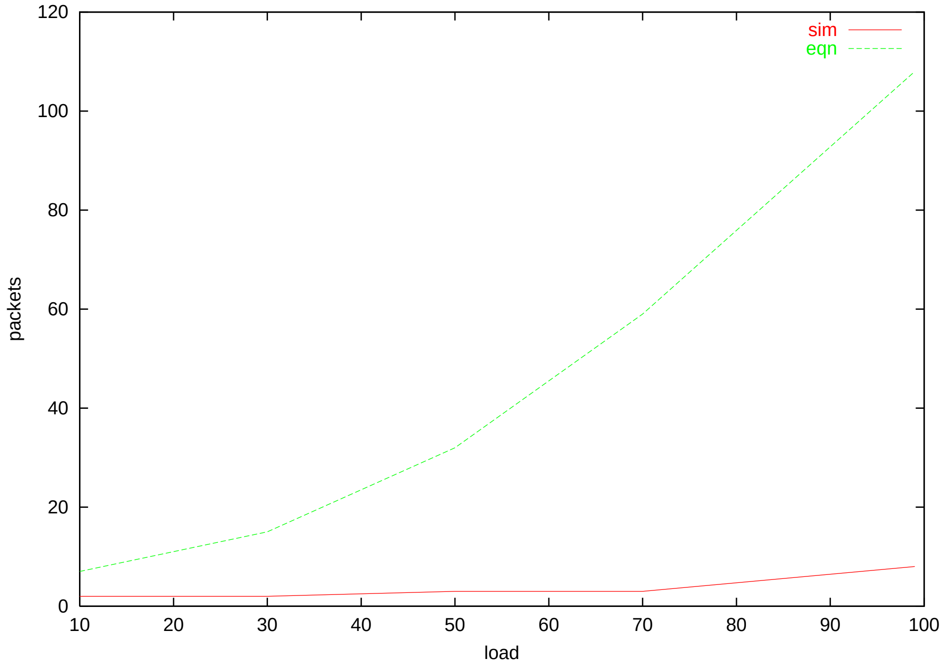

Figure 6-14: Backlog under Non-worst case traffic

still accurate. They reflect the bound assuming a worst case arrival pattern, which obviously is not occurring in the average case scenario.

Currently non-real-time traffic far exceeds real-time traffic. It is likely that non-real-time traffic (such as e-mail and file transfers) will continue to make up a significant portion of the traffic mix on wide area networks. The loads that affect real-time traffic only include other real-time traffic. It is assumed that non real-time traffic has lower priority. Unless the traffic balance changes significantly to favour real-time traffic, we are unlikely to see networks running with a real-time traffic load near 100%. Therefore the large divergence at higher loads between the worst case predicted and actual delays seen is unlikely to occur in practice.

6.3 Simulation of a Multihop Flow

In the worst case scenario of channels bursting for as long as possible and the bursts arriving simultaneously for the same output link, the single hop simulations show a very accurate correlation to the equations in chapter 4. In this section the simulations are run over multiple hops and the end-to-end queuing delay is compared.

The multi-hop simulations are performed over a ten hop network shown in figure 6-15[3].

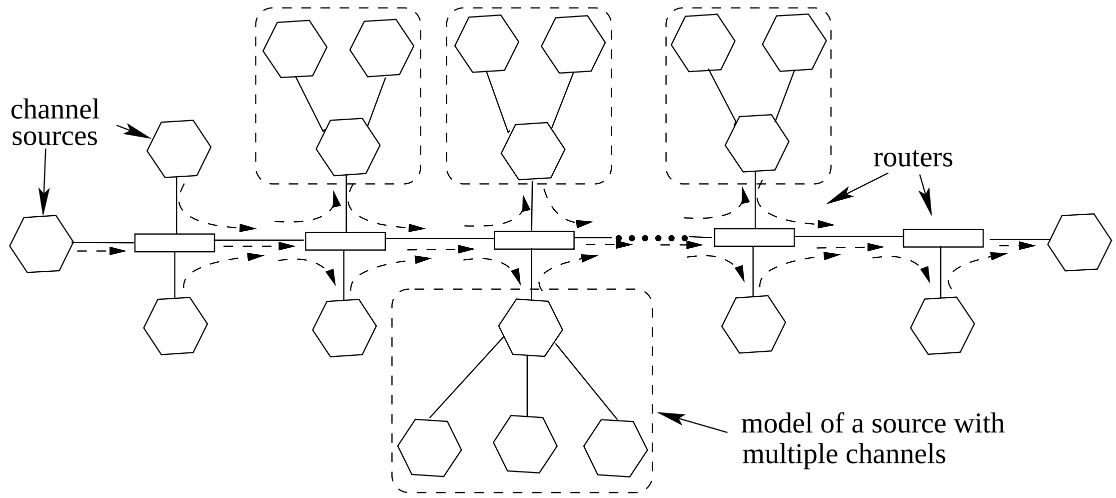

Figure 6-15: Multiple Hop Simulation

A ten hop case is chosen as it is sufficient to show the accumulation of jitter effects and is a reasonable number of hops for a traffic flow in a wide area network. The behaviour of the traffic originating at the three leftmost nodes and destined for the rightmost node is monitored.

At the intermediate hops, additional real-time cross traffic is generated. The cross traffic is generated as one or more real-time channels multiplexed on each input link. This method is used rather than a more conventional model of combined cross traffic (i.e. Poisson) for two reasons. First, accurate models of combined real-time cross traffic on non-rate controlled wide area networks (such as the Internet) are still an active area of research and subject to change as different applications become more prominent on the network. Some models have been proposed, but no single model has been widely accepted at this time. Prior to 1993, Poisson models were typically used to characterise cross traffic combinations for wide area traffic. This tended to smooth

[3]The configuration of the third hop is repeated for hops four through eight.

the typically bursty nature of the cross traffic. This model has since been replaced due to the work of Paxson and Floyd[65] which showed that the combination of such traffic does not tend to smooth but instead exhibits long range dependencies or self-similar characteristics. Some work has supported independent identical distribution of packet arrivals in a given time interval, but simulations have only shown some correlation with compressed video sources[14][13]. As such, many experiments use actual or created sources matching the above models at the edge of the network to generate cross traffic as well as the specification of source traffic. To simulate bursty cross traffic, controlled rate on-off sources are used as cross traffic in the simulations.

The second reason for using controlled rate sources as cross traffic is due to the use of per channel based scheduling. The scheduler requires specific channel information about the arriving traffic. The scheduler must have access to each individual channel specification or alternatively a specification of the combined channels in order to perform schedulability analysis. Although defining combined flow specifications is a possibility, it is likely to to be more pessimistic than scheduling the individual channel flows. This assumption of bursty cross traffic does not affect the comparison of the calculations with the simulation as the calculations are based on the the cross traffic model actually generated in the simulations, so both the simulation and the calculations are making the same assumptions, but as discussed earlier the actual delays seen may not be indicative of a particular network and should only be taken as a comparison of the accuracy of the equations in determining the delay bounds.

It is not practical to create the worst case scenario within the simulator for the multi-hop case. To do so would require significant modification of the simulator to allow control over packet arrival patterns. The worst case arrival pattern of the source traffic would always have to arrive at each hop at the same time as the worst case arrival pattern of the cross traffic. Given that the worst case delay at earlier nodes may be experienced by packets of either the cross traffic or the source traffic, we cannot predict the exact arrival time of the source traffic at the next node. Choosing a correct start time for the cross traffic at the next node such that it would coincide with the arrival of the source traffic packets from the previous node would not be possible unless we control the order of queuing of packets. Manipulating the queuing from straight FIFO would bring into question the validity of the results. In addition to queuing, variation in input link rates will also cause packets to arrive at different times even if the channels start sending packets at the same time. As such, the simulations are not guaranteed to create the worst case scenario and reflect more closely the average case for the various traffic types studied.

Simulations in this section are run for continuous bit rate traffic, bursty traffic and variable traffic (i.e. average and peak rate adhered to but the instantaneous rate is chosen randomly between the average and peak). These correspond respectively to the cases in Section 4.7 as follows:

- Constant Bit Rate Traffic:
  - Total Incoming Channel Peak Rate $\leq$ Output Link Rate
- Bursty Traffic:
  - Total Incoming Channel Peak Rate $>$ Output Link Rate
- Variable Bit Rate Traffic:
  - Total Incoming Channel Peak Rate $>$ Output Link Rate

The variable rate traffic has the same traffic characteristics as the bursty traffic; but does not guarantee to produce the worst case traffic arrival, even at the sources. To show the cumulative effect of jitter, the total calculated and simulation delays are shown at each of the hops along the path. The propagation and transmission delays have been removed, so the delays shown in the graphs are solely cumulative queuing delays. In order to isolate the end-to-end effects on the source traffic, the cross traffic generated at a node is destined for the next node so it only has an effect at the one hop.

6.3.1 Continuous Bit Rate Traffic

The first set of comparisons (figure 6-16) shows the cumulative delay at each hop for continuous bit rate source traffic. In these sources the minimum packet interarrival time, x_{min}, and the average packet interarrival time, x_{ave}, are equal. The sources will send packets continuously with a spacing of x_{ave} between packets. Since in all cases the sum of S_{max}/x_{ave} of all the multiplexed sources must be less than the output link rate, this represents the case of equation 34. The simulation is run and plotted at three load levels where the total input rate of the sources is 10%, 50% and 99% of the output link rate. The graph shows the cumulative queuing delay for different length paths from 1 to 10 hops.

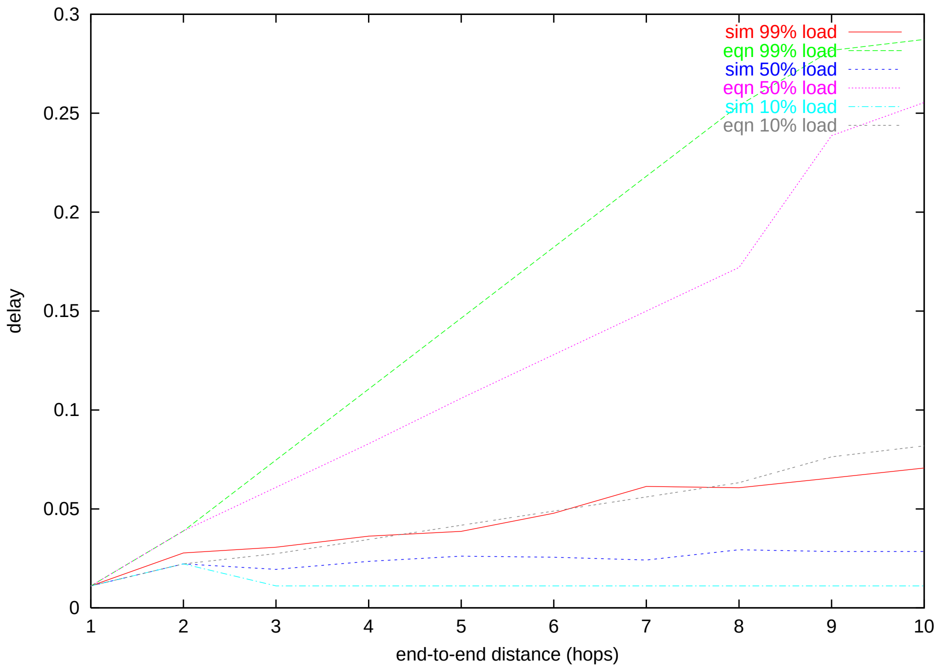

Figure 6-16: Continuous Bit Rate Traffic - End-to-End Delay by Path Length

For a single hop path the calculation and the simulation are the same. This is consistent with the results from the single-hop simulation. For longer distances the delay bound is pessimistic. This is due to two assumptions of the equations that are not actually realised in the simulations. The first assumption is that packets from all sources have experienced the worst case delay. This is obviously pessimistic since of the three sources multiplexed onto the first hop, only one of these could possibly have experienced the worst case delay. We could assume that whichever source has the highest rate suffers the worst case delay as this would maximize the additional packets due to jitter in equation 34. However, this will only be valid if the sources all follow the same path. If the sources follow different paths, it is necessary to assume that any of them could have suffered the worst case delay to correctly determine their potential contribution to the maximum traffic arrival at later nodes. To keep the calculation of delay separate from routing information, we assume that each of the sources experiences the worst case delay. Since the traffic arrival at nodes after the first hop is pessimistic the delay will be higher than actually seen. This increase in the delay calculation at the node will increase the total worst case jitter assumed at the next node, again increasing the assumed worst case number of packet arrivals and the calculated delay. This effect accounts for some of the pessimism in the calculated versus the simulation delays.

The second reason for the pessimistic results is the assumption that the worst case traffic arrival pattern occurs at each hop. As discussed in the introduction, it is not possible to create this effect in the simulator beyond the single hop, so at most hops the bursts becomes statistically multiplexed in the simulation and do not arrive simultaneously. This is very apparent in the lower load levels where the worst case delay is experienced at the first hop; but at subsequent hops the probability of contention is very low and no additional queuing delay is seen. This is why the end-to-end queuing delay for all path lengths in the 10% load case is equal to the first hop delay and the 50% case only shows small increases over the first hop delay. At the higher load (99%) the probability of contention increases and we see higher additional queuing delay at nodes after the first hop in the simulation as well as the equation.

Note that in the simulations the end-to-end delay measurements sometime decrease. This is due to the simulation design. The simulator does not provide information on per hop queuing delay, so the simulations were run independently as networks with one hop, two hops, three hops, etc, and the worst case total end-to-end queuing delay recorded for each simulation. Since these simulations were run independently, occasionally the queuing seen in one path length was larger than runs with a greater number of hops. This is indicative that we are not seeing the worst case scenarios in the simulations.

Figure 6-17 shows the benefit of considering serialization of packets on the same link. In the non-rate controlled case (nrc-fifo), all packets are assumed to arrive again simultaneously at the next hop, which results in pessimistic delay bounds. Since rate control is not used, it is also possible for packets which were delayed at previous nodes to arrive during the same time period, as explained in chapter 4. Since the delays are pessimistic, the number of additional packets arriving after being delayed at previous nodes is pessimistically high. This leads to an even larger overassumption of the delay at the next node. This cumulative effect results in exponential growth of the delay bound.

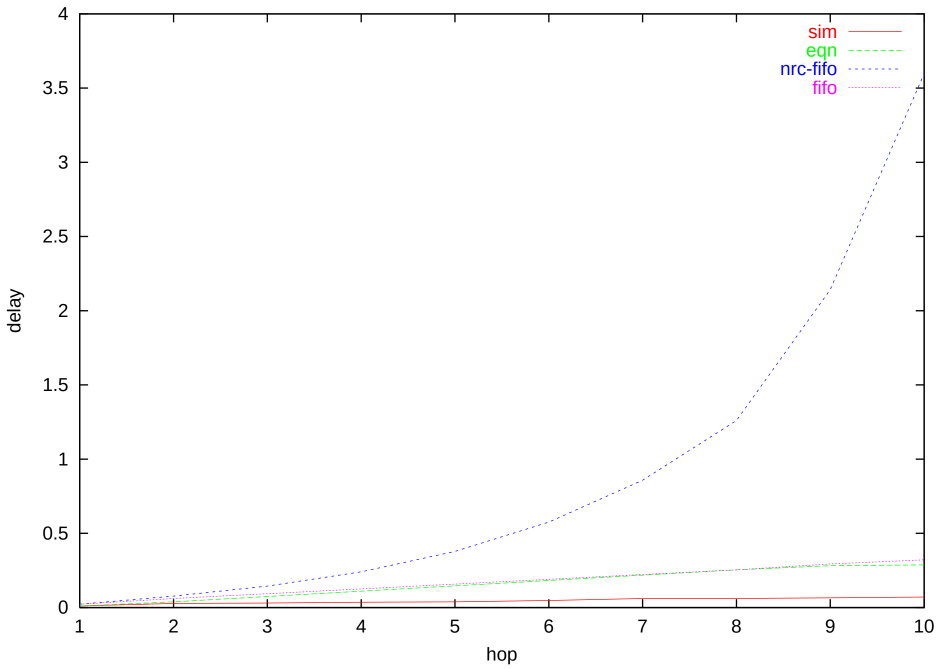

Figure 6-17: Comparison of Models - Continuous Traffic - 99% Load

It is interesting to note that the end-to-end delay calculated by equation 34 is very close to that calculated by Zhang's rate controlled FIFO model (fifo). The two models take quite different approaches since equation 34 takes into account previous delays and assumes that packets can not be delayed again by the packets that delayed them at earlier nodes (i.e. serialization) whereas Zhang's rate controlled FIFO model assumes the arrival of packets from all sources at each node, but does not assume that the source rates could be higher than specified due to delayed packets at earlier nodes. There is no obvious reason that they should be correlated; but it is probably worth exploring this effect in further research to see if under continuous bit rate arrivals the worst case delay calculated by the equations in this thesis is always identical to those calculated by Zhang's equations.

6.3.2 Bursty Traffic

The next set of simulations study bursty traffic where the peak rate (S_{max}/x_{min}) of sources can exceed the output link rate as long as the sum of their average rates (S_{max}/x_{ave}) is no greater than the output link rate over the defined averaging interval I. The simulation is compared to the delay calculated by the equation 75 in Appendix A. Comparisons are plotted in figure 6-18 where the sum of the average source rates is equal to 10%, 50% and 99% of the output link rate. In these simulations all sources send a burst of I/x_{ave} packets with a spacing of x_{min} between packets. This causes the sources to send as many packets at the highest rate possible within their traffic specification and then stop sending until the next interval I. This maximises the burstiness of the traffic.

As expected, the worst case backlog both calculated and seen in the simulation is greater than with continuous bit rate sources. The larger bursts allow for more packets to be queued at a node before the output link is able to clear them. The backlog calculated by the equation is still approximately six times more pessimistic than the maximum backlog actually seen in the simulation, but this is again due to the fact that the bursts do not arrive simultaneously at each hop and the assumption

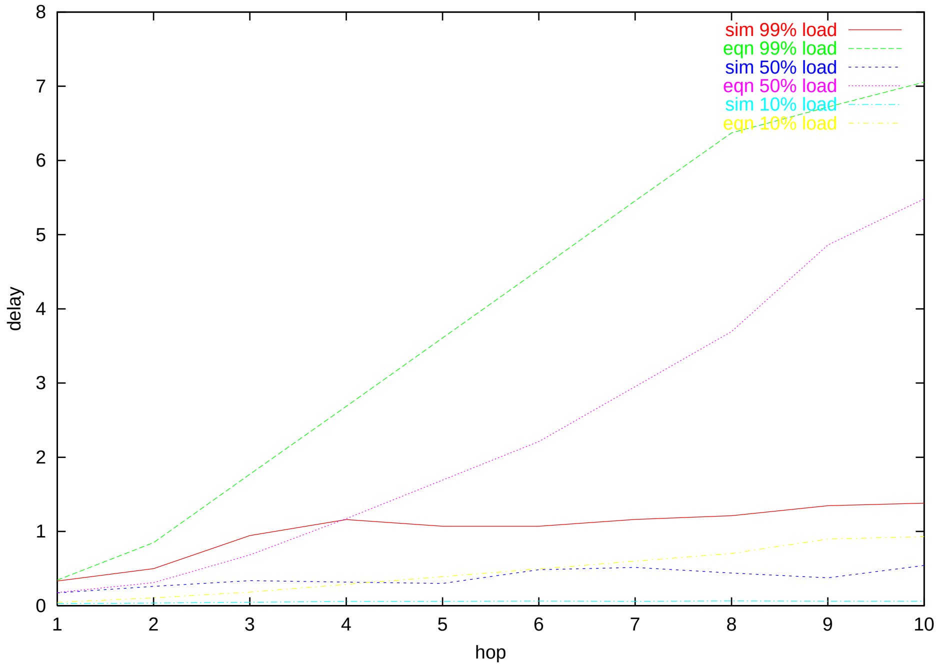

Figure 6-18: Bursty Traffic - Total Delay at each Hop

that every channel experiences the worst case delay at each node in its path. In the case of the 10% load, the burst at nodes after the first is gone by the time the packets traveling end-to-end arrive and additional bursts do not coincide with the source traffic arrival so no queuing delay is seen after the first hop.

Figure 6-19 compares the simulation and the delay bounds from equation 75 with Zhang and Pusopa's delay bound calculations. In this graph the scale is set to include the maximum bound calculated by Zhang's model at the tenth hop. The simulation and equation bounds are at the very bottom of the graph. Figure 6-20 shows the same plots with the scale quartered so that the simulator and equation values are more visible.

Unlike the continuous bit rate case, Zhang's bounds are far more pessimistic for the bursty traffic. The assumption that all sources are able to produce bursts simultaneously at each hop, results in bounds nearly 100 times more pessimistic than those of the equation 75. Another limitation of Zhang's bounds is that the equation

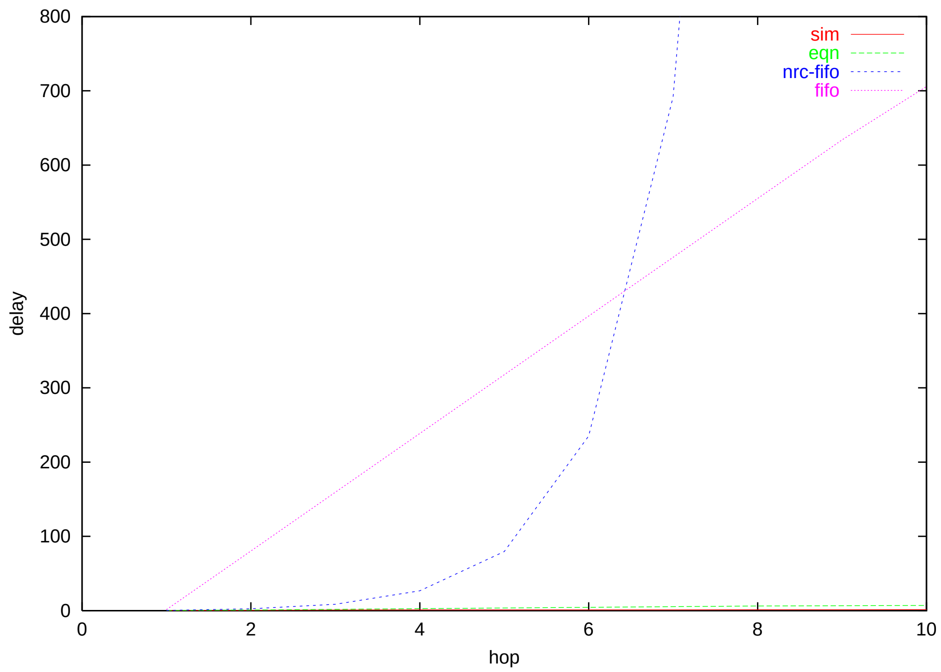

Figure 6-19: Comparison of Models - Bursty Traffic - 99% Load

results in a division by 0 when $\sum_{\forall j} S_{max,j}/x_{ave,j} = r_{out}$. So it is not possible to calculate bounds for 100% average load. Equation 75 has no such restrictions.

The non-rate controlled bounds show an even greater exponential increase with the bursty traffic. Since these bounds are overly pessimistic in the single hop case, especially for bursty traffic, this pessimism is carried over into the calculations of arrivals for the next hop. This causes the rapid exponential growth of the assumed worst case arrival pattern and a resulting exponential growth of the worst case delay calculation.

6.3.3 Variable Rate Traffic

The final set of simulations is a variation on the bursty traffic. Instead of sending the maximum burst initially, the sources are allowed to vary their rates. The sources send at their average rate over time but are allowed to burst at rates between their specified average and peak rates. The traffic specifications have not changed, just the arrival patterns. Therefore the equations in section 4.8 still apply and the bounds calculated

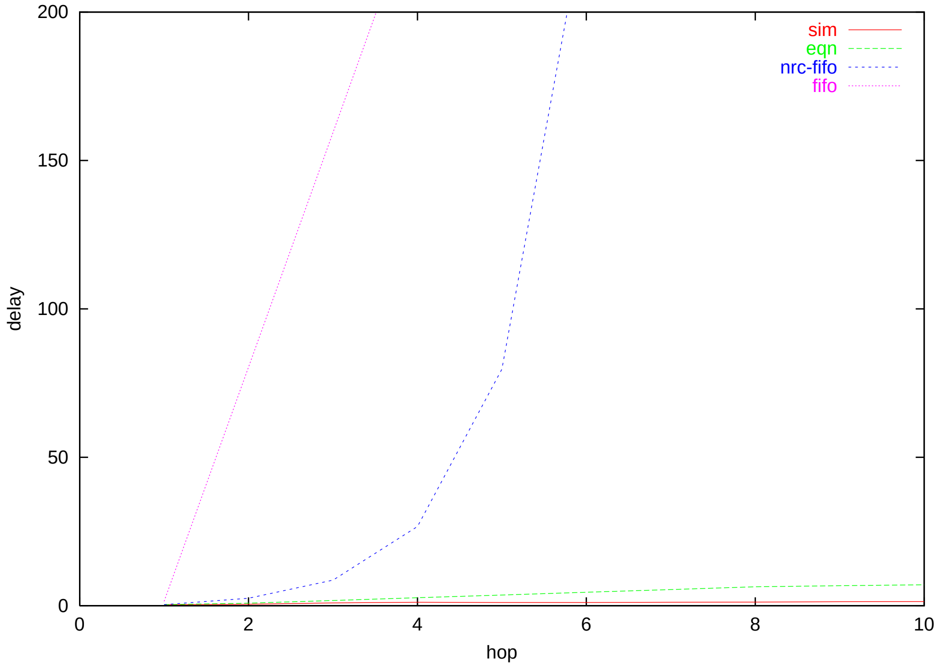

Figure 6-20: Comparison of Models - Bursty Traffic - 99% Load - Closer View

are the same. Figure 6-21 shows the values calculated by the equations and the delay seen in the simulations. As before, average loads of 10%, 50% and 99% are plotted.

The behaviour is very similar to the bursty case, although the variable rate simulation does show slightly lower delay at high loads. This is consistent with a further reduction in the probability of two or more bursts arriving at a node simultaneously. The one anomaly in the graph of the 99% load simulation at node two occurs because the simulator does not use an averaging interval in determining the average rate when generating variable traffic from the sources. In the simulation with highest delay at node 2 the generated traffic was higher than the stated average over the averaging interval used in the calculations. In the other simulations, the averaging interval was consistent with the traffic generated in the simulation. For interest, the bounds were recalculated with an averaging interval that matched the generated traffic. This resulted in a calculated upper delay bound of 1.511 for node 2 compared with 1.429 seen in the simulation.

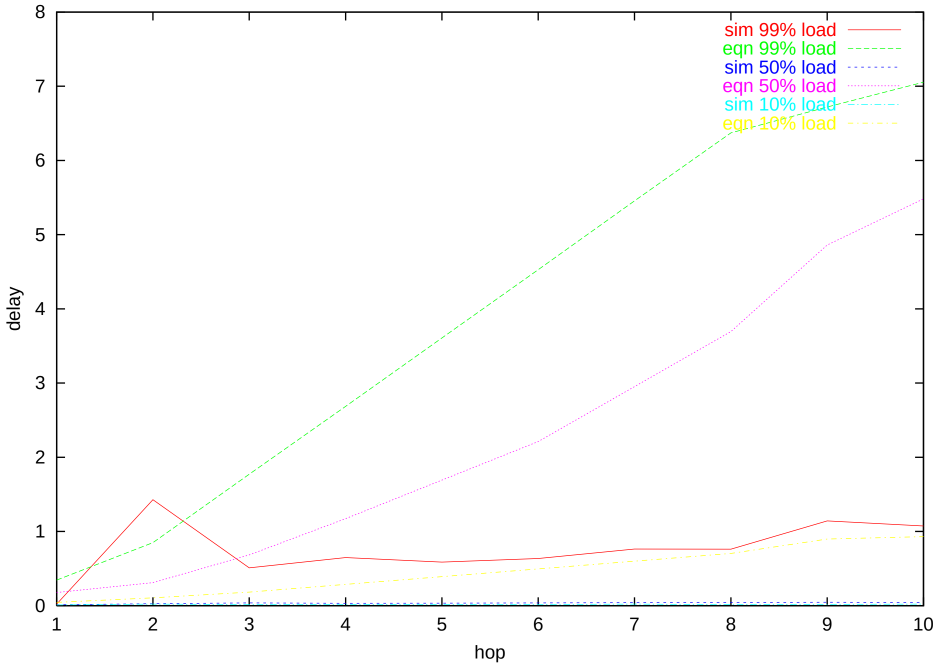

Figure 6-21: Variable Rate Traffic - Total Delay at each Hop

The final graph (figure 6-22) compares the delay bounds of the rate controlled and non-rate controlled models with the simulation and the equations from section 4.8.

The scale of the delay in the graph is half that of figure 6-19 so that the simulation and thesis plots can be differentiated. As before it is clear that the equations from section 4.8 are far more accurate than the existing work of Zhang and Pusopa in predicting the actual delays seen in the simulation.

6.4 Discussion

The simulations in this chapter clearly show that the delay bounds presented in chapter 4 are significantly more accurate than those of existing related work in the literature. This accuracy is important for scheduler based quality of service guarantees. Pessimistic bounds can lead to an application deciding that its delay requirements can not be met by the network even when they can. Given the significant difference in delays between the work of Zhang and Pusopa and the work presented here, multimedia

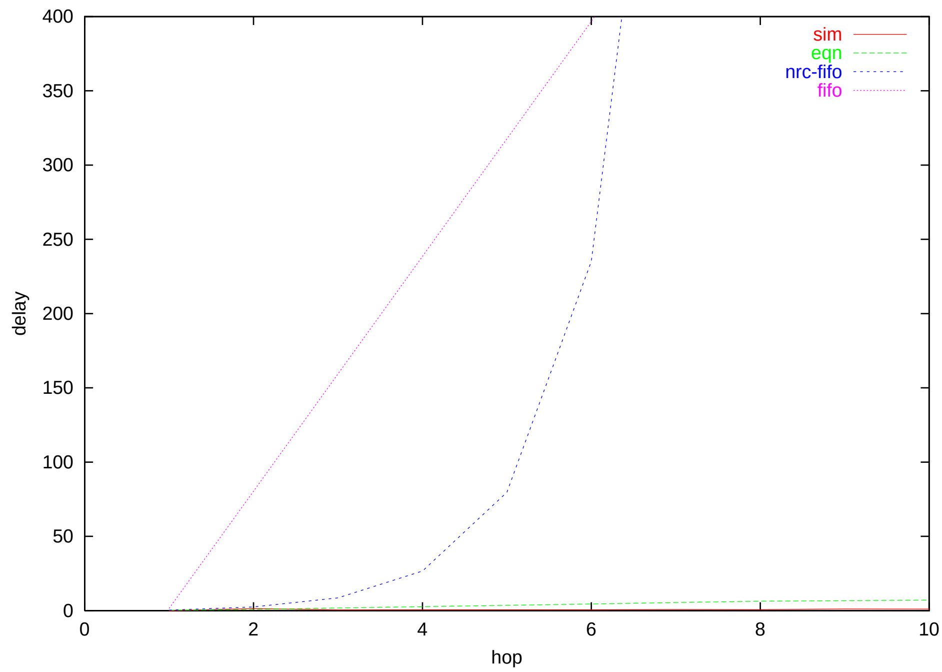

Figure 6-22: Comparison of Models - Variable Rate Traffic - 99% Load

and other time sensitive applications will be far more likely to be offered delay bounds that they can accept. Even if the applications could accept the more pessimistic bounds, each source and hop added will increase the pessimism significantly. The delay bounds calculated will limit the total number of new channels that can be accepted without violating existing delay bound guarantees. Since the contribution of new channels to the delay bounds is highly pessimistic, the total number of applications that will be able to be accepted for guaranteed quality of service will be smaller.

Although Zhang's bounds for the rate controlled case may be no more pessimistic than the bounds presented in this thesis for continuous bit rate traffic, they are significantly more pessimistic when examined across other traffic types. Since most real-time data traffic on wide area networks is bursty rather than continuous in nature this difference is important. It is also important to emphasize that the bounds presented by Zhang assume that rate controllers exist at each hop to reshape traffic to its original specification. The bounds presented in this thesis do not assume the use of rate controllers. Hence we can achieve the same or smaller end-to-end delay bounds without the use of potentially expensive rate controllers within the network.

Pusopa's model is based on Zhang's work and extends the work to a non-rate controlled network. The primary difference was to add in the possibility of arrival of packets queued at earlier nodes along with the arrival of packets from all of the sources. This results in bounds that are even more pessimistic than Zhang's for more than a few node hops. Pusopa's model is also limited in the case of non-continuous bit rate traffic to sets of channels which cannot burst multiple times within the arrival of the largest burst in the set. In other cases the bound is actually optimistic and can claim to guarantee a delay bound that the network may not meet.

The main limitations in the published equations is that there is no consideration of the effects of serialization and input link rates. This results in a much higher traffic arrival assumptions than can actually occur. The equations in this thesis are able to capture both of these effects in the equations and therefore lead to much tighter delay bounds. The simulations in this chapter show that considering serialization and link rates makes a significant difference to the delay bounds. The difference is significant enough to allow for the removal of per node rate controllers and still allow for a tighter upper bound on packet backlog and delay.

The backlogs seen in the simulations of the multi-hop case are significantly lower than those calculated by the equations. Most of this difference can be accounted for by the fact that the worst case traffic conditions can not be forced to occur over multiple hops in the simulator. However, there are still some characteristics of multiple channels sharing a link that are not captured by the equations in this thesis. For instance, if several channels are sharing a link and produce a worst case traffic arrival pattern by bursting as long as possible, only one of the channels will suffer the worst case delay (the one whose packet is at the tail of the queue at the point of maximum backlog). The equations for determining multi-hop delay bounds assume that all of the channels experience this worst case delay bound. This does accurately bound the worst case; but is clearly pessimistic since a higher delay bound leads to a larger number of packets assumed to arrive at the next node. By capturing which channel experiences the worst case delay, a more accurate worst case backlog could be predicted for multi-hop channels. There are likely to be other aspects of traffic behaviour that could be captured by more complex equations to further improve the accuracy of the bounds on worst case packet backlog. The insights and equations in this thesis have significantly improved on the current bounds in the literature; but there are still further improvements that can be made in future work.

Chapter 7

Fault Tolerance in Real-Time Packet Switched Networks

7.1 Introduction

Applications which require quality of service often also desire or require reliability of that service. Faults in the network should not necessarily imply the loss of an application's connectivity and performance guarantees. Faults reduce the resources of the network and the task of fault tolerance can be compared to that of establishing real-time channels. Both tasks involve preserving the performance of a channel from the effects of concurrent processes competing for network resources: either other traffic, which gracefully requests resources, or faults, which abruptly remove resources. In best-effort packet switched networks, such as the Internet, the availability of redundant paths is used to recover from the loss of links or nodes. Distributed routing algorithms detect the failure, typically due to a timeout in the response from a control message, and update routing tables to route over alternative paths. This approach presents several problems for real-time channels with delay and loss guarantees.

First there is no guarantee that an alternative path exists that can support the performance provided by the original path. In a failure situation, the load on alternative paths is likely to increase thereby increasing queuing delays and packet loss. Unless the resources are reserved a priori to the occurrence of the fault, it is impossible to guarantee that the channels will be able to be scheduled on alternative paths without degradation in their performance.

Another problem is that fault recovery in itself implies a delay in the receipt of packets. When a fault occurs time is needed to detect and recover from that fault. For applications requiring reliable guaranteed service, the properties of the fault recovery must be considered when determining the performance bounds offered by the connection admission control protocol.

This chapter proposes methods that address these two problems in providing fault tolerance to real-time channels with delay and loss guarantees. The first contribution is a method for reducing the overhead of the resources needed to guarantee recovery on an alternative path with the same performance guarantees as the original path. The second is an algorithm for choosing the node from which to build the recovery path and a method for accounting for fault recovery time within the delay guarantees of the real-time channels.

7.2 Methods of Fault Tolerance

There are several approaches taken to providing fault tolerance and recovery for channels with QoS guarantees. These approaches can be classified into two main categories: *proactive* and *reactive* [7]. The proactive approach can be further divided into *active* and *passive* approaches. Solutions from each of these categories have been proposed in the literature: [8](active) [101](passive).

Proactive fault tolerance techniques use additional network resources to support fault recovery. The additional resources are reserved for the channel and are not available for providing guarantees to other channels. These channels may be in active use as proposed by Banerjea [6] where the data is split amongst several channels with redundant information sent on additional channels to allow for error correction when a fault occurs. Alternatively the channels may be passive where the resource reservations are in place, but the channel capacity is not used unless a fault occurs [100]. In the case of passive fault tolerance, the reserved resources can be used by non-real-time traffic when not being used to recover from a fault.

Reactive fault tolerance attempts to recover as quickly as possible from a fault, but does not reserve resources a priori. Reactive fault tolerance does not guarantee that recovery will be possible or that if it is possible that the same performance guarantees can be met. However, it does provide a lighter weight approach to fault recovery as additional reservations are not required. This approach is likely to be useful to applications which can tolerate a significant recovery time and possibly reduced performance. The characteristics of this approach are studied by Banerjea in [7].

The focus of this thesis is on applications with guaranteed network performance requirements. Therefore the work proposed in this chapter follows the proactive model. In wide area networks the majority of applications are best effort and do not require QoS guarantees. Due to this characteristic, the methods proposed in this chapter are passive to allow non-real-time traffic to make use of the extra resources reserved for the real-time traffic until a fault occurs.

7.3 Towards an Efficient Proactive Fault Recovery Technique

Providing *guaranteed timely* delivery of real-time messages in point-to-point networks requires the reservation of network resources such as buffer space and bandwidth based on worst case traffic characteristics (e.g. maximum source rate, maximum transmission delay). As shown in chapter 6, these bounds are based on the worst case traffic arrival patterns and are therefore pessimistic on average and tend to reserve more resources than are actually needed in most cases. If fault-tolerance of a real-time connection is required, then additional backup resources must be reserved. Current fault-tolerant real-time communication protocols require an overhead that may be 100% or higher of the resources required by the primary channel, even for the simple case of tolerating single node/link failures. If a high level of fault tolerance is required, the extra resources will impact on the ability of the network to accept primary guaranteed channels even though the additional resources for fault tolerance are used only rarely.

In some cases the failure assumptions for the network are such that some failure conditions are considered impossible. For example, if it is expected that no more than a certain number f of links will fail at any one time, then it is unnecessary to reserve resources for the simultaneous failure of more than f links. In other cases, when the fault-tolerant protocol can only tolerate a certain number of faults per channel, say f, then the resource dependencies between different channels may be such that some of the back-up resources for the two channels may never be required simultaneously. In these and in similar cases there is strong motivation to optimise the reservation of back-up resources. If these failure assumptions for the network and/or dependencies between the resources required by different channels can be taken into account, this will, in general, reduce the overall resource requirements for the network, allowing more real-time traffic to be guaranteed and reducing the minimum guaranteeable delays that the network can offer.

This section presents an approach to extending existing fault-tolerant protocols by using more sophisticated and optimal allocation of network resources. Resources that are never expected to be used simultaneously are used in a *interleaved* manner. Since this idea was introduced by the author in [68], related work has supported the benefits of this approach for scheduling real-time channels [39].

7.3.1 Establishing Real-Time Channels

The protocol in this section is based on the establishment of real-time channels, a concept introduced by Ferrari and Verma [23]. real-time channels were discussed in Chapter 2 and used for the analysis in Chapter 4. This section extends the introduction of Chapter 2 by specifying algorithms and problems that provide a background for adding fault tolerance to real-time channels.

A real-time channel τ_i running through a link l_j will be described as a tuple $(x_{min,i}, P_i^j, d_i^j)$, where P_i^j is the maximum packet transmission time over the link, and d_i^j is the packet delivery bound assigned to the link. We may omit the superscript j when the link number is irrelevant to the discussion. Note that this tuple is related to the tuple $(x_{min,i}, S_{max})$ introduced in Chapter 2 and used in Chapter 4 as follows: $P_i^j = S_{max}/r_{l_j}$

A set of n channels $\mathcal{T}^n = \{\tau_i = (x_{min,i}, P_i, d_i) \mid 1 \leq i \leq n\}$ is said to be *schedulable* over a link if for all i the maximum queuing delay experienced by all packets on channel τ_i over the link is not greater than the requested delay bound d_i. Obviously, a new channel can be successfully established through a link only if all existing channels through the link together with the new one are still schedulable.

The basic algorithm [103] for establishing a real-time channel with end-to-end delay D_i is as follows.

Algorithm 1:

Step 1 Set up a minimum-hop basic circuit $C_b = v_0 l_1 v_1 \ldots l_k v_k$ from the source to the destination nodes, with nodes v_j and links l_j.

Step 2 For all nodes $v_j \in C_b$, determine the minimum delay $d_{min,i}^j$ on l_j.

Step 3 If $D_i \geq \sum_{j=1}^n d_{min,i}^j$ then the channel can be established. Assign the link delay over l_j to be $d_i^j = d_{min,i}^j + \sigma_i/k$, where $\sigma_i = D_i - \sum_{j=1}^n d_{min,i}^j$ [1].

Otherwise the establishment request fails.

Hidden in Step 2 above is the problem of computing the minimum guaranteed delay over a link.

Minimum Guaranteed Delay Problem: Suppose a set $\mathcal{T}^{n-1}$ of $n-1$ channels are schedulable over a link, where:

$$\mathcal{T}^{n-1} = \{\tau_i = (x_{min,i}, P_i, d_i) \mid 1 \leq i \leq n-1\}$$

[1]Note that the value σ_i/k represents an even distribution of the the available slack in the delay among the k hops. This implies the assumption of rate control at each node. The bounds in this chapter all assume rate controlled networks as does the original work that they extend.

> Given a new channel τ_n, with $x_{min,n}$ and P_n, what is the minimum value $d_{min,n}$ such that all n channels $\mathcal{T}^n = \{\tau_i = (x_{min,i}, P_i, d_i) : 1 \leq i \leq n\}$ are still schedulable?

Zheng and Shin [102] describe an algorithm for computing the minimum guaranteed delay bound, and a proof of its correctness. The proof attempts to also serve as a description of some factors the authors felt were contributing towards the delay bound. I have developed alternative correct algorithms and proofs emphasizing the timing behaviour which facilitates the further work described in this chapter. The bounds are the same as those in [102], although some symbols have been changed to coincide with the symbols used in the other chapters of this thesis. I describe the algorithm for computing the minimum delay bound below as it is essential for the rest of this section.

7.3.1.1 Minimum Guaranteed Delay Bound

Let $f(t, d_n) = \sum_{i=1}^{n} \lceil (t - d_i)/x_{min,i} \rceil^+ P_i$ [2] be the total time needed by all n channels to transmit all packets generated by time t with deadlines before t.

The minimum delay bound for channel τ_n can be no smaller than the transmission time P_n and is exactly that if all n channels are schedulable with $d_n = P_n$, i.e if for all times t we have $f(t, P_n) \leq t$. Otherwise, we need to find a $d_{min,n}$ such that at all times t we have $f(t, d_{min,n}) \leq t$. In that case $d_{min,n} = \max\{d^t : f(t, P_n) > t\}$, where d^t is the minimum delay that can be met at time t.

Figure 7-1 illustrates how d^t is formed. At time t, the schedulability condition $f(t, P_n) < t$ is violated, $k + 1$ packets have arrived on τ_n, and the last two packets, numbered k and $k + 1$ have not been fully transmitted. Extra time $E(t) = f(t, P_n) - t$ is required to transmit these two packets and the greatest delay will be experienced by the earlier packet, numbered k. That delay is the difference between when it arrived, i.e. $kx_{min,n}$, and the time when it would be fully transmitted, i.e. $t + e_k(t)$.

[2] $\lceil x \rceil^+ = n$ if $n - 1 \leq x < n$; $= 0$ if $x < 0$

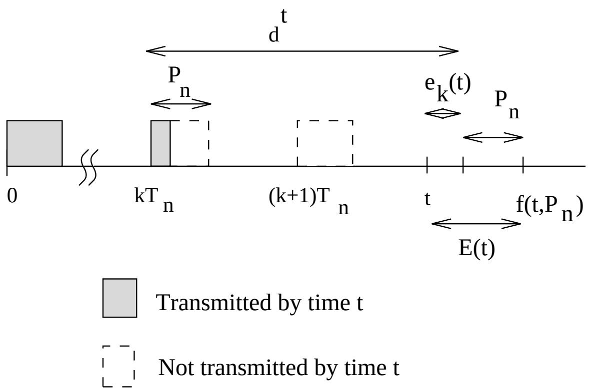

Figure 7-1: Determining d^t

In general, the extra time needed by the earliest packet to complete transmission is:

$$e_k(t) = (f(t, P_n) - t) - (\lceil (f(t, P_n) - t)/P_n \rceil - 1)P_n$$

That is the time to send all untransmitted packets generated by time t minus the time to send the untransmitted packets arriving after the k'th packet.

The index of the earliest packet in τ_n that needs further transmission at t can be determined by subtracting the number of packets needing transmission after t from the total number of packets generated by time t.

$$k = \lceil (t - P_n)/x_{min,n} \rceil^+ - \lceil (f(t, P_n) - t)/P_n \rceil$$

With equations for $e_k(t)$ and k, we can now write equations for d^t and $d_{min,n}$.

$$d^t = t + e_k(t) - kx_{min,n} \tag{57}$$

$$d_{min,n} = \max\{\{P_n\} \cup \{d^t \ : \ f(t, P_n) > t\}\} \tag{58}$$

In the expression for $d_{min,n}$ above, the values for d^t are computed over $\{t \ : \ f(t, P_n) > t\}$, which in general is innumerable and unbounded. Fortunately, it can be shown that a bounded and discrete set of points G is sufficient.

- Let $S_i = \{d_i + mx_{min,i} \ : \ 0 \leq m \leq \lfloor (t_{max} - d_i)/x_{min,i} \rfloor\}$. S_i contains the points t at which packet deadlines expire on channel τ_i.

- Let $S = \cup_{i=1}^{n} S_i$. S contains the set of "interesting" points t at which $f(t, P_n)$ changes and schedulability needs to be checked.
- $t_{max} = \max\{d_1, \ldots, d_n, L\}$, t_{max} is an upper bound on when $f(t, P_n)$ can be $> t$[3]
- $L = [\sum_{i=1}^{n}(1 - d_i/x_{min,i})P_i]/(1 - \sum_{i=1}^{n} P_i/x_{min,i})$
- $G = S \cap \{t \,:\, f(t, P_n) > t\}$

The complexity of the computation for the minimum guaranteed delay bound is therefore of the order of the size of S, which is reasonably small when the link usage $\sum_{i=1}^{n} P_i/x_{min,i}$ is not too close to one. However, as the link utilisation approaches one, S could become very large as the example at the end of this section will demonstrate.

7.3.2 Fault Tolerant Real-Time Channels

An attractive feature of point-to-point communication networks is the potential for high reliability. High reliability comes from the multiple communication nodes and interconnecting channels existing in the network that provide natural redundancy. This natural redundancy provides multiple communication paths between any two nodes in the network. Some of these paths may be minimum-hop paths and others may be of arbitrary length. The appearance of faults in the network may remove some of the available alternatives; but in principle two nodes can continue communicating as long as there remains at least one connecting path in the network. Non-real-time networks and non-real-time traffic in a real-time network usually make use of some or all of the available alternative paths, either for the sake of increased performance through adapting to traffic conditions or for the sake of fault-tolerance.

The real-time channels discussed above cannot take advantage of these possibilities. All packets of a real-time channel are transmitted along a single pre-determined path and a single failure on that path will cause the entire channel to fail. An obvious way of increasing the tolerance of the channel to failures is to expand the channel with extra

[3] See [101] for the proof of this bound.

links and nodes that can be used to re-route packets around faulty components on the original channel. Each of the alternative paths provided in the extended channel must guarantee the channel delay bounds and, as mentioned above, this can only be achieved through resource reservation. Therefore the cost of a fault-tolerant real-time channel is proportional to the number of additional links and nodes involved. These reservations are idle under normal (non-failure) operation, yet they reduce the networks capacity to carry additional primary (e.g. active) channels.

The higher the degree of fault-tolerance, in terms of the number of faults that can be tolerated before the entire channel is disabled, the higher the cost of the channel. In any particular case a trade-off between cost and degree of fault-tolerance will have to be made. Zheng and Shin [103] introduce *single-failure-immune (SFI) channels* to guarantee timely delivery of a packet as long as the packet encounters no more than one link/node failure on its way to the destination node. We will now consider SFI channels in some detail as this section will critically evaluate some of the results and will propose an alternative approach to fault-tolerance in real-time networks.

7.3.2.1 Single Failure Immune (SFI) Real-Time Channels

First we introduce some useful definitions in a manner similar to [102]. A communication network is modeled as a directed graph $N = (V, L)$, where V is the set of nodes and L is the set of directed links. A *basic circuit* C_b from node v_0 to node v_k is the network $C_b = (V_b, L_b)$ that comprises the connecting path $c_b = v_0 l_1 v_1 l_2 \ldots l_k v_k$, where $v_i \in V_b$ are nodes and $l_i = \overrightarrow{v_{i-1} v_i}$ is a directed link from node v_{i-1} to node v_i. An *SFI circuit* C_s from node v_0 to node v_k is defined as the network $C_s = (V_s, L_s)$ that comprises a basic circuit C_b from v_0 to v_k augmented with some extra nodes and links, which are called *detours* of the basic circuit, such that there exists a basic circuit from v_0 to v_k in C_s if no more than one node/link (except the source and the destination nodes v_0 and v_k) are removed from C_b. Notice that the removal of a node includes the removal of all links incident to/from it.

An SFI real-time channel τ_s is defined as the basic channel τ_b and the detour channels τ_d. τ_s is established by the following algorithm:

Algorithm 2 (SFI):

Step 1 Establish a basic real-time channel τ_b over a minimum-hop basic circuit C_b with connecting path $c_b = v_0 l_1 v_1 \ldots l_k v_k$ from the source to the destination nodes.

Step 2 Let the network of extra nodes and links $C_d = (\emptyset, \emptyset)$. For $i = 1, \ldots, k$, do the following:

Step 2.1 Remove node v_i if $i \neq k$, and link l_k if $i = k$, from the basic circuit C_b.

Step 2.2 Establish a detour real-time channel τ_i over a minimum-hop circuit C_i. If there are more than one such circuits choose the one that results in an *optimal* circuit in the sense of [102].

Step 2.3 $C_d = C_d \cup C_i$.

Step 3 If the algorithm fails at Step 2, there does not exist an SFI circuit from v_0 to v_k. Otherwise, $C_s = C_d \cup C_b$ is an SFI circuit connecting v_0 and v_k with C_b as its basic circuit, and $\tau_s = \tau_d \cup \tau_b$ is an SFI channel over C_s with τ_b as its basic channel and τ_d as its back-up detour channels.

Algorithm 1 is used to establish a channel τ_i over a circuit C_i, where τ_i can be either a basic channel or a detour channel. At each node v_j in C_i the minimum guaranteed delay for the channel over link l_i must be computed.

Minimum guaranteed delay in SFI: Let C_i be a circuit with a connecting path $c_i = v_0 l_1 v_1 \ldots l_k v_k$ from the source node v_0 to the destination node v_k. The circuit can either basic, i.e. $C_i = C_b$, or detour, i.e. $C_i \subseteq C_d$.

For every link l_j in C_i let $\tau_{i,j}$ represent the portion of the channel τ_i to be established through l_j. Find the minimum guaranteed delay $d^j_{min,i}$ of $\tau_{i,j}$ schedulable on l_j over the set of channels $S_{n,j}$, which comprises *all basic and detour channels* existing through that link but excluding those that are from τ_s.

The minimum guaranteed delay $d^j_{min,i}$ is computed over the set of all channels running through that link but excluding those that are in τ_s. This prevents the scheduling of more than one basic or detour channel in τ_s at any time. The failure assumption of SFI channels is that at any time at most one node in C_s may fail and therefore at most one connecting channel in τ_s will be active at any time.

Step 2.2 of algorithm 2, elaborated in [103], ensures that a detour route C_i is as close to the existing routes C_s as possible and is optimal, i.e. it does not contain any redundant links. This minimises the overhead of back-up resources in the final C_s.

It is difficult to obtain a general expression for the overhead of SFI channels in an arbitrary topology network, as different choices of the basic circuit and detours could result in different numbers of extra nodes/links. However, it can be noticed that the size of the detour circuit will be no less than that of the basic circuit, as for each node/link we have to find another connecting path of at least the same length as that of the basic circuit, which must contain, as a minimum, one node/link not in the basic circuit. As a crude estimate then we can assume that the overhead of SFI channels is 100% or more. For example, an SFI channel on a planar mesh topology requires backup overhead of exactly 100%, as shown in [67][103].

To extend SFI to tolerate up to n faults over a channel (i.e. the basic channel and its detours), algorithm 2 must be extended and applied iteratively n times. Since the faults may now occur not only in the basic circuits, but also in the detour circuits, a new detour channel must be established for each combination of n failed node/link pairs. The overhead of such a channel (which we will name a Multiple Fault Immune (MFI) channel) will be a minimum of $N\times$ 100%. The overhead will also be affected by the connectivity of the network. For example in a 4-connected mesh, there are only 2 possible sets of minimum-hop paths out of a node. An MFI channel with fault-tolerance $n > 2$ will have detours that are longer than the basic circuit and the overhead factor for each detour circuit for $n > 2$ will be greater than 100%. This is just to illustrate that this approach to fault-tolerance is very resource consuming and may become prohibitively expensive if applied to all real-time channels in a network.

7.3.2.2 Using Network Characteristics to Reduce Resources for Back Up Channels

An SFI channel τ_s guarantees timely delivery of all packets as long as there is only one failed node/link in the SFI circuit C_s over which the channel is routed. The guarantees on τ_s are not affected by the state of the network outside the circuit C_s, which may indeed suffer any number of failures. A separation of concerns is achieved and an SFI channel can be considered in isolation from the rest of the network. However, a disadvantage of ignoring the failure behaviour and reliability of the network is the overhead of detour resources that may never actually be needed.

Consider a network N where a link l_j may have detour channels running through l_j that cover the failures of a set of nodes F_j. In principle, all nodes in F_j may fail and the link l_j will still guarantee that all deadlines of all channels through l_j are met. Furthermore, any SFI channel that includes at most one node from F_j will meet its end-to-end deadlines as well. In general, all nodes $v_i \in F_j$ will be within some close proximity of the link l_j as the SFI algorithm for setting up detour circuits guarantees that they will be as close to the basic circuit as possible. It then makes sense to define the failure behaviour of the network in terms of the number of failed components within a certain area at any time. In most practical cases this number is bounded and known a priori as faults tend to be cataclysmic as in the case of natural disasters, in which case there is a reasonable likelihood that no recovery will be possible for channels requiring passage through a small geographical area, or the faults tend to have a low probability and multiple faults an even lower probability.

If a bound f_j can be put on the number of nodes in F_j that can be expected to fail at any time, then we only need to reserve detour resources over l_j for the worst case failure set of up to f_j nodes. By a worst case failure set of nodes for l_j we mean the one that imposes the most stringent detour resource requirements on l_j. The link guarantees all deadlines in case at most f_j nodes in F_j fail. No guarantees are given for all cases when more than f_j nodes fail. If f_j is the same as or greater than the size of the set F_j, i.e. $f_j \geq [F_j]$, then the reliability and the overheads are the same as those of SFI.

However, if $f_j < [F_j]$ then the network can reduce the resources needed to achieve the required reliability. In the simplest case when $f_j = 1$, we only need to be able at any time to successfully schedule over l_j all basic channels and the detours of the worst case node in F_j. In the case of a uniform homogeneous network, we can choose all f_j to be the same and equal to some constant f, i.e. $f_j = f$, where f reflects the failure assumptions across the whole network. Alternatively, in the case of a heterogeneous wide-area network the bounds f_j may be different to allow for different failure assumptions across the network. For simplicity we will begin with the assumption that all bounds are equal to 1, i.e. $f_j = 1$.

Because in a sense a node/link can at any one time back-up only a subset of all nodes/links in its failure set and covers the whole set by interleaving the various subsets over time, we have called this approach to fault-tolerance *Interleaved Back-up Resources* or *IBR*. The algorithm for setting up a fault-tolerant routing circuit C_s for a fault-tolerant real-time channel τ_s in an IBR network is essentially the same as algorithm 2 for setting up an SFI circuit. A circuit $C_s = C_b \cup C_d$ is composed of a basic circuit C_b and its detour circuits C_d. A channel $\tau_s = \tau_b \cup \tau_d$ is composed of a basic channel τ_b and a collection of detour channels τ_d. The difference is in computing the minimum guaranteed delay bound $d^j_{min,i}$ for the portion $\tau_{i,j}$ of the channel over circuit C_i running through link l_j.

For clarity we will repeat the algorithm first and will then formulate the minimum guaranteed delay problem in IBR networks.

Algorithm 3 (IBR/SFI):

Step 1 Establish a basic real-time channel τ_b over a minimum-hop basic circuit C_b with connecting path $c_b = v_0 l_1 v_1 \ldots l_k v_k$ from the source to the destination nodes.

Step 2 Let the network of extra nodes and links $C_d = (\emptyset, \emptyset)$. For $i = 1, \ldots, k$, do the following:

Step 2.1 Remove node v_i if $i < k$, and link l_k if $i = k$, from the basic circuit C_b.

Step 2.2 Establish a detour real-time channel τ_i over a minimum-hop circuit C_i. If there are more than one such circuits choose the one that results in an *optimal* circuit in the sense of [102].

Step 2.3 $C_d = C_d \cup C_i$.

Step 3 If the algorithm fails at Step 2, there does not exist an SFI circuit from v_0 to v_k. Otherwise, $C_s = C_b \cup C_d$ is an SFI circuit connecting v_0 and v_k with C_b as its basic circuit, and $\tau_s = \tau_d \cup \tau_b$ is an SFI channel over C_s with τ_b as its basic channel and τ_d as its back-up detour channels.

Algorithm 1 is used to establish a channel τ_i over a circuit C_i, where τ_i can be either a basic channel or a detour channel. At each link l_j in C_i the minimum guaranteed delay bound $d^j_{min,i}$ for the portion τ^j_i of the channel over circuit C_i running through link l_j is computed. The computation depends on whether the channel is basic, i.e if $\tau_i = \tau_b$, or detour, i.e. if $\tau_i \in \tau_d$. In the former case, the minimum delay bound is such that all existing basic channels, the new basic channel τ^j_b, and all detour channels of the worst case node from the failure set F_j for the link l_j must be schedulable over l_j. In the later case, the minimum delay bound is such that all existing basic channels, all existing detour channels for node v_i, and the new detour channel τ^i_j for the same node v_i must be schedulable.

The informal description above assumes prior knowledge of the worst case node w from the failure set F_j which with its set of detours channels $\mathcal{T}_w$ imposes the worst scheduling requirements on the link and results in the greatest $d^j_{min,n}$. However, since the computation for $d^j_{min,i}$ changes with the choice of set G, and G is affected by the new channel τ_n. there appears to be no easy way of determining w. Instead we can organise the computation of $d^j_{min,n}$ in a way that produces the same result as if w were known in advance, but avoids the costs of computing all possible combinations.

At any time t, we select the set $\mathcal{T}^t_w$ that makes the greatest contribution towards d^t and use that result for d^t. Since $d^j_{min,n}$ is defined as the maximum of all such

$\{d^t : t \in G\}$, the final result will be same as if we used $\mathcal{T}_w$ at all times. The set $\mathcal{T}_w$ is the set with the largest scheduling requirements $f_w(t)$ at time t. Given a set of $n-1$ channels $\mathcal{T}_v$, the time needed to transmit on time all of the packets that arrive by some time t is:

$$f_v(t) = \sum_{i=1}^{n} \lceil (t - d_i)/x_{min,i} \rceil^+ P_i$$

All contributions to the sum are positive. If d_i is greater than t, the contribution to the sum by the channel τ_i is 0. For the channels to be schedulable, the time needed must be less than or equal to t.

When a new channel τ_n (either basic or detour) is added to the set, the time needed by τ_n is added to that already required by the existing channels.

$$\sum_{i=1}^{n-1} \lceil (t - d_i)/x_{min,i} \rceil^+ P_i + \lceil (t - d_n)/x_{min,n} \rceil^+ P_n$$

It can be seen from the equation, that the larger the time requirements of the existing channels in $\mathcal{T}_v$, the smaller the requirements of the new channel must be. The only way to reduce the requirements of the new channel is to increase its delay bound d_τ. Therefore, at time t the worst case set of channels $\mathcal{T}_w^t$ for that time has the largest $f_w(t)$. This set(s) of channels will maximize d_τ at time t.

Because the set of channels that actually determines the value of d^t changes with t it is convenient to consider the expression for d^t defined above as a function D^t over a set of channels $\mathcal{T}$, i.e. $d^t = D^t(\mathcal{T})$. Similarly, the computation for t_{max}, which determines the size of the domain S, can be considered a function Z over a set of channels $\mathcal{T}$, i.e. $t_{max} = Z(\mathcal{T})$.

Minimum guaranteed delay in IBR ($f = 1$): Let τ_n be a real-time channel that is being established over C_i and τ_n^j is the portion of τ_n running through link l_j in C_i. Let $\mathcal{T}_b$ be the set of all basic channels on l_j.

Case 1: $\tau_n \in \mathcal{T}_b$ **is basic** $d_{min,n} = \max\{\{P_n\} \cup \{d^t : t \in G\}\}$ where

$d^t = D^t(\mathcal{T}_b \cup \mathcal{T}_w^t \cup \tau_n)$

$G = S \cap \{t : f_b(t) + f_w(t) + f_{\tau_n}(t) > t\}$

$t_{max} = \max\{Z(\mathcal{T}_b \cup \mathcal{T}_v \cup \tau_n) : v \in F_j\}$

Case 2: $\tau_n \in \mathcal{T}_v$ **is detour** and $v \in F_j$ $d_{min,n} = \max\{\{P_n\} \cup \{d^t\,:\, t \in G\}\}$ where

$$d^t = D^t(\mathcal{T}_b \cup \mathcal{T}_v \cup \tau_n)$$
$$G = S \cap \{t\,:\, f_b(t) + f_v(t) + f_{\tau_n}(t) > t\}$$
$$t_{max} = Z(\mathcal{T}_b \cup \mathcal{T}_v \cup \tau_n)$$

In either case the schedulability computation is over a subset of the channels that are used for computing the minimum delay bound in SFI. This can significantly reduce the overhead due to the interleaving as is shown in Appendix C. It can also be shown that t_{max} can be chosen without computing it over all combinations, in a manner similar to the selection of $\mathcal{T}_w$, reducing the computation even further. In an IBR network, we are able to schedule many more channels, both basic and detour, successfully and to improve the resource utilisation of the network. The restriction of $f = 1$ can be removed by selecting the additional $f - 1$ next worst case channels.

7.4 IBR vs SFI: an example

In this section we give an example that shows the benefits of using IBR. We wish to establish a real-time channel, τ_n over a link which is currently carrying one basic and three detour channels. The channels have the following characteristics (minimum interarrival time, transmission time, delay bound):

$$\tau_{v1} = (20, 2, 5)$$
$$\tau_{v2} = (15, 3, 6)$$
$$\tau_{v3} = (10, 3, 8)$$
$$\tau_b = (20, 4, 12)$$
$$\tau_n = (15, 2, d_n)$$

To establish an SFI channel, we must calculate the minimum delay bound of τ_n over all existing channels on the link. t_{max} is 109 and the $S = S_1 \cup S_2 \cup S_3 \cup S_b \cup S_n$ where:

$S_1 = \{5, 25, 45, 65, 85, 105\}$

$S_2 = \{6, 21, 36, 51, 66, 81, 96\}$

$S_3 = \{8, 18, 28, 38, 48, 58, 68, 78, 88, 98, 108\}$

$S_b = \{12, 32, 52, 72, 92\}$

$S_n = \{2, 17, 32, 47, 62, 77, 92, 107\}$

$\forall t \in S$, $f(t, P_n) \leq t$, except the times $t \in G$, where $G = \{6, 8, 12, 18, 21, 32, 38, 52\}$. $\forall t \in G$, $\max(d^t) = 14$. This is the minimum delay bound, $d_{min,n}$ for the new channel τ_n.

An IBR channel is established differently depending on whether the new channel is a detour channel or a basic channel. If τ_n is a detour channel due to the failure of link l_i (or node n), the minimum delay bound, $d_{min,n}$ is calculated over the set $\tau_n \cup \tau_{l_i} \cup \tau_b$. Let us presume that τ_{v1} and τ_{v2} are also detours due to the failure of link l_i and τ_{v3} is a detour due to a different link l_k. $d_{min,n}$ is calculated over the channels $\tau_b, \tau_{v1}, \tau_{v2}$ and τ_n. In this case, $t_{max} = 19$, $S_1 = \{5\}$; $S_2 = \{6\}$; $S_b = \{12\}$; $S_n = \{2, 17\}$; $G = \{6\}$. At time $t = 6$, $d^t = 7$. As expected, a tighter delay bound is possible, t_{max} and the size of S are smaller; and the complexity of determining $d_{min,n}$ is reduced.

When τ_n is a basic channel, the worst case set of detour channels must be used to determine schedulability. For the set of channels $\mathcal{T}_l = \{\tau_{v1}, \tau_{v2}, \tau_b, \tau_n\}$, $t_{max,l} = 19$. For the set of channels $\mathcal{T}_k = \{\tau_{v3}, \tau_b, \tau_n\}$, $t_{max,k} = 13$. We choose the worst case $t_{max} = 19$. The contributions to the set S are $S_1 = \{5\}$; $S_2 = \{6\}$; $S_3 = \{8, 18\}$; $S_b = \{12\}$ and $S_n = \{2, 17\}$. Let $\forall t \in S$,

$$u_i(t) = \sum_{\forall \tau_x \in \mathcal{T}_i} \lceil (t - d_x)/\mathcal{T}_x \rceil^+ P_x.$$

The sums at each t for $\mathcal{T}_k$ and $\mathcal{T}_l$ are:

t	$\mathcal{T}_k$	$\mathcal{T}_l$	$f(t, P_n)$
2	0	0	2
5	3	2	5
6	3	5	7
8	3	5	7

12	3	5	11
17	3	5	13
18	6	5	14

The set of channels $\mathcal{T}_l$ are the worst case at all times except $t = 0$ (where the channels are equal) and $\{t = 5, t = 18\}$ where $\mathcal{T}_k$ is worst.

At each time $t \in S$, we check the schedulability of $\mathcal{T}_w^t \cup \mathcal{T}_b \cup \tau_n$. The sets are schedulable (i.e. $f(t, P_n) \leq t$) at all times except $t = 6$. At time 6, $d^t = 7$ which is the minimum delay bound, $d_{min,n}$ for the new channel. If the channels were unschedulable at other times, the maximum d^t would be chosen for $d_{min,n}$.

As is shown in this example, IBR can significantly reduce the number of time points which must be examined in finding the minimum delay bound as well as reducing the minimum delay bound that can be offered to a new channel. These benefits will be realised whenever back up channels for different link failures can be grouped and the number of different groups that can fail simultaneously is bounded.

7.4.1 Summary of the Cost and Benefits of IBR

Interleaved Backup Resources (IBR) provides an algorithm which allows a tighter bound on the resource overhead needed for providing fault tolerance in real-time channels. By considering network characteristics, interleaving can be used to reduce idle periods for resources under both normal and faulty conditions. Minimizing the resource overhead is critical to connection based protocols, such as real-time channels which use reservations to guarantee quality of service since all such reservations effectively reduce the overall networks capacity for carrying such traffic. Backup (detour) channels are particularly wasteful, since under normal operation they are idle and can not be used to carry any real-time traffic. IBR offers a method for reducing this overhead.

The only additional information which must be known over SFI in order to provide an IBR channel is the link or node which the backup channel is providing recovery for.

This does not represent a significant amount of additional state to be stored, relative to the channel tuples.

Interleaving can also be used for basic channels if a relationship exists between the channels such that only a known subset of the channels may be sending packets at any time t. This extension is a possible area for further work.

7.5 Timing Effects of Switching from Primary to Backup Channels

Even when resources are available for establishing backup paths, faults due to node or link failures can lead to very costly, unbounded recovery time to re-establish the connection [64]. To *guarantee* recovery in bounded time for critical communication, proactive protocols, which reserve additional network resources to be used in the event of failure, have been proposed.

Such protocols are expensive in terms of network resources (i.e. buffer space and bandwidth) [67] [68]. Although active backups can reduce the amount of resources which are unavailable to real-time traffic as shown in [8], the backup resources are also unavailable to non-real-time traffic. Resources reserved for passive backups only affect real-time traffic, and may still be used by non-real-time traffic. Since non-real-time traffic (i.e. data transfer, e-mail, etc.) is expected to be a significant proportion of general network traffic, passive backup will be more suitable for many general purpose networks.

Current published schemes for establishing passive backup channels, including the IBR scheme introduced in the previous section, consider only the delays on the backup channel when determining packet delays during faults. The timing of the recovery itself is not considered or is assumed to be negligible. In wide area networks, the recovery time can be quite large relative to the packet deadlines over a link. If the time to detect and react to failure is not taken into consideration when determining the total delay to packets, deadlines may be missed. This section formally defines the additional timing and resource requirements needed for recovery of real-time channels under local (backup path established from node immediately prior to the failed node/link), source (backup established from the source) and recursive nearest node recovery (backup can be established from any node along the channel's primary path) schemes. An algorithm is presented for selecting a recovery node which minimizes resource requirements while guaranteeing no more than a specified number of packets will miss deadlines during recovery due to being dropped or delayed. This section also discusses the need for recovery management to relax resource constraints and teardown unused channels.

7.5.1 Recovery Timing Effects on End to End Delay

In this section, a real-time channel τ is defined by the minimum interarrival time between packets, $x_{min,\tau}$, the maximum packet size S_τ, and the maximum tolerable end-to-end delay D_τ. The maximum acceptable packet loss, $P_{max,\tau}$ is also defined for fault recovery purposes.

The maximum time required to detect and react to a fault is called the worst case recovery time (WCRT). The worst case delay experienced by a packet in the event of network failure can therefore be defined as the sum of the k node and link delays on the *backup* channel, the WCRT, and the worst case delay between the point of recovery, r, and the failed node, f, or failed link out of f.

$$D = \sum_{i=1}^{k}(d_i + l_i) + WCRT + \sum_{i=r}^{n}(d_i + l_i) \quad n = \begin{cases} f & \text{for link failures} \\ f-1 & \text{for node failures} \end{cases} \tag{59}$$

D_e is used in this section to represent the final term in the above equation. D_e is the worst case extra delay experienced by packets that have already passed the recovery node but have not passed the node that fails. This extra delay is in addition to the delay on the backup channel. d_i is the transmission and queuing delay experienced by packets on τ at link i. l_i is the propagation delay over link i.

WCRT is dependent on the recovery scheme used. The details of fault detection and handling are not explicitly stated in [103] [101] but the point of recovery is defined;

and reasonable assumptions can be made based on this as to the methods which could be used for recovery.

Three points of recovery are examined: local recovery from the node immediately prior to the failed node or link, source recovery from the source node and recursive nearest node recovery from any node. Equations bounding WCRT, additional packet delay, and resource (buffers and bandwidth) overhead associated with the recovery are given for each method.

7.5.2 Local Fault Recovery

Local recovery is used in [101]. Recovering from the node immediately prior to the failed node or link allows for fast reaction to faults. The WCRT is bounded by the acknowledgment time-out and the time to update the routing table locally. Additional buffer space, B_{loc} is needed to hold packets until the packets are known to have passed the failed node if a timeout period has not occurred.

$$B_{loc} = \lceil \frac{\min(timeout, D_e)}{x_{min,\tau}} \rceil - P_{max,\tau} \qquad (60)$$

The timeout period may be unrelated to acknowledgment of data packets and may be associated with a control response to an "are you there" request. Arrival at the next node does not guarantee successful transmission since the node may fail while the packet is queued. If an acknowledgment is sent upon transmission from the next node, the acknowledgment will be delayed by the delay at the next node.

The extra delay, D_e, is equal to the sum of the propagation delay over the link in the case of link failure and is equal to the propagation delay plus the queuing and transmission delay at the failed node in the case of node failure. In real-time applications where the delay at the failed node is high relative to the end-to-end delay, the acknowledgment of packets is of little benefit, since it is unlikely that a resent packet would arrive within its deadline. In these cases, a smaller timeout value could be used.

Using local recovery has the advantage of minimizing the cost to detect failure and removing the need to communicate the detection of a failure to the source node in bounded time. However, unless the path is allowed to cycle, the number of possible routes is reduced and a longer path may have to be chosen or no path may exist. Figure 7-2 shows an example of a path where local recovery would be required to cycle in order to find a back up path.

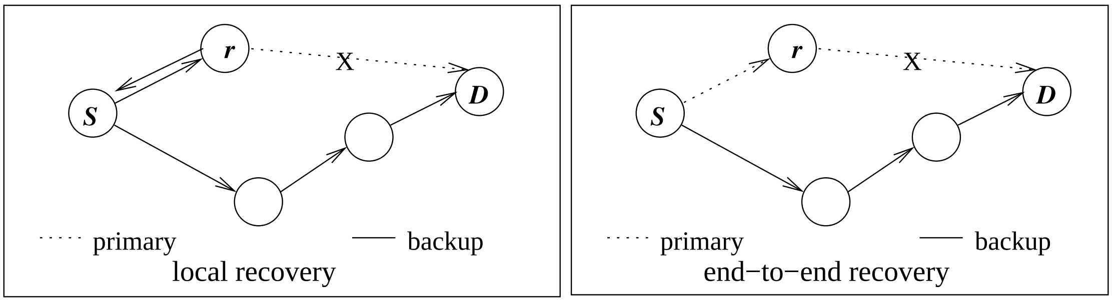

Figure 7-2: Routing Fault Recovery

If cycles are allowed, bandwidth and buffer space must be reserved over the distance of the cycle and the delay over the cycle must be considered. The total delay consists of the delay from the source to the recovery node, the delay over the cycle and the delay from the end of the cycle to the destination, e.g.:

$$D = \sum_{i=1}^{n}(d_i + l_i) + \sum_{i=n}^{c}(d_i + l_i) + \sum_{i=c}^{k}(d_i + l_i) \tag{61}$$

Where n is defined as in equation 59, c is the end point of the cycle and k is the sink.

7.5.3 Source Fault Recovery

Recovery from the source node gives the most efficient use of resources since the backup path is chosen over all possible paths between source and destination with no cycles. In [103], the backup paths are established in this manner. The backup channel does not become active until the source detects the fault. To bound the fault detection time, a second real-time control channel, ρ, is established between the destination and source to transmit fault notification packets. The recovery time is bounded by the transmission time on the real-time channel used to transmit fault notification.

$$WCRT = \sum_{i=n}^{1}(d_{i,\rho} + l_i) \tag{62}$$

In cases where the fault occurs close to the destination, the time to notify the source node of the fault may be large relative to the time between packets, thus causing one or more packets to be sent on the failed channel. These packets will need to be resent and will experience an additional delay equal to the time they have already spent in the network. In the worst case this delay is:

$$D_e = \sum_{i=1}^{n}(l_i + d_{i,\tau}) \tag{63}$$

Significant buffering may be needed to buffer packets for resend at the source in the case where low (or zero) packet loss has been specified in the channel establishment request.

$$B_{e\Leftrightarrow e} = \lceil \frac{\sum_{i=1}^{n}(l_i + d_{i,\tau} + d_{i,\rho} + l_i)}{x_{min,\tau}} \rceil - P_{max,\tau} \tag{64}$$

Source recovery insures that the least cost path is chosen for recovery; but has the worst case additional delay and packets resent. Since recovery must be guaranteed even from failure at the node/link nearest the destination, recovery by this method can not succeed unless D_τ is greater than triple the propagation delay between the source and destination to allow for sending the packet to the last link, which fails, notifying the source of the failed link and resending the packet from the source on the backup channel. The high delay makes source recovery unlikely to succeed for many wide area real-time communication applications.

7.5.4 Recursive Nearest Node Recovery

Combined methods exist which try to recover from the node immediately prior to the fault; and then upon failure to find a suitable channel, try to establish the channel from the source. This approach shares the problems of both local and source recovery (i.e. large failure detection times, sub-optimal paths, and high resource overhead). A protocol which allows for recovery from any node in the channel's path, overcomes these problems. The protocol is more complex, but most of the computation required occurs during channel establishment and does not affect the delay of packets while they are being transmitted. The recovery node is chosen based on the channel requirements of acceptable packet loss and maximum packet delay, and the network state. The algorithm proceeds as follows:

Given the maximum acceptable packet loss, $P_{max,\tau}$; minimum interarrival time between packets, $x_{min,\tau}$; and the available buffer space at link j, B_j

Step 1: For each link, j along the channel τ's path starting from the source until a recovery node r is found do the following:

Step 1.1: Find the packet loss, $P_{act,j,\tau}$, at link j.

$$P_{act,j,\tau} = \lceil \frac{D_e + WCRT}{x_{min,\tau}} \rceil \tag{65}$$

Step 1.2: if $P_{act,j,\tau} \leq P_{max,\tau} + B_j$ let r be the node with output link j. Establish a least cost path channel, τ_r from r to the destination node.

Step 1.3: If the algorithm fails at step 1.2, a backup channel can not be established at j.

Step 2: Let T_r be the set of links in τ_r and T_j be the set of links in the primary channel between link j and the failed link/node.

Step 2.1: if $T = T_r \cap T_j \neq \emptyset$, then for all links $k \in T$ beginning at the link closest to the destination do the following:

Step 2.1.1: Find the packet loss, $P_{act,k,\tau}$, at link k as in Step 1.1.

Step 2.1.2: if $P_{act,k,\tau} \leq P_{max,\tau} + B_k$ the node associated with link l becomes the recovery node.

Step 1 of the algorithm finds the first node which has sufficient buffering and recovery time to handle a fault. Step 2 then moves the recovery node as close to the destination as possible to minimize the number of packets which must be buffered. This algorithm is guaranteed to find a path if either local or source recovery are able to find a path. The chosen path is the least cost path which can still guarantee the recovery within the specified delay and packet loss requirements. The proof of these characteristics are given below:

Theorem 1 *If either local or source recovery are able to find a backup path, then the recursive nearest node algorithm will also find a path.*

Proof: Assume source recovery is able to find a recovery path τ_e. Then at the link from the source node S, $P_{act,S,\tau} \leq P_{max,\tau} + B_S$ must be true (otherwise no guarantee can be given that $P_{max,\tau}$ will not be violated). The recursive nearest node algorithm will attempt to establish a least cost path, τ_r, from the source to the destination given these conditions (Step 1.2). Therefore, $\tau_e == \tau_r$ and if τ_e satisfies the source requirements τ_r must also.

Assume local recovery is able to find a recovery path τ_l. Let r be the node prior to the failed node/link on τ. τ_l is made up of the least cost path from the source to node r and the least cost path from node r to the destination which does not pass through the failed node/link. The recursive nearest node algorithm will attempt to establish a least cost path at some node, n, on the path of τ between the source and node r. The recursive nearest node recovery path τ_r will be made up of the least cost path from the source to n and the least cost path from n to the destination.

If τ_l is the least cost path between the source and destination then $\tau_r == \tau_l$ since both paths consist of the least cost path from the source to n, the least cost path from n to r and the least cost path from r to the destination. If a lower cost path exists between n and the destination which does not include r, then the original channel τ can not be a least cost path, which is a contradiction.

If τ_l is not the least cost path between the source and destination, then the cost of $\tau_r < \tau_l$, since τ_r would choose τ_l as its path if a lower cost path were not available. Since the cost of the path $\tau_r \leq \tau_l$ if τ_l is a valid path then so is τ_r.[4]

Theorem 2 *The path chosen by the recursive nearest node algorithm is the least cost path which can guarantee recovery.*

Proof: The node chosen for recovery by any algorithm must be a member of the nodes in τ since only these nodes are carrying packets for channel τ. No node prior to the node, n, chosen in Step 1.2 is capable of recovery since earlier nodes cannot guarantee that $P_{act,j,\tau} \leq B_j + P_{max,\tau}$ (i.e. maximum packet loss will not be exceeded). As shown in the last two paragraphs of the above proof, any node after n will either create an equivalent recovery path or one of greater cost. Therefore the path chosen by step 1 of the recursive nearest node algorithm will be the least cost path which can guarantee recovery. Step 2 of the algorithm does not change the path (only the node to be notified of the failure). Any node selected in step 2 must also satisfy the constraint of $P_{max,\tau}$ (step 2.1.2). Therefore the least cost property obtained in Step 1 holds.

In addition to these two characteristics, recursive nearest node recovery will tend to use a smaller or equal number of resources than local or source recovery and be more likely to succeed in establishing the backup channel. Although it is possible for local recovery to succeed on a non-least cost path and therefore require fewer additional buffer spaces to be reserved, the path will be on more heavily loaded nodes and links. Recursive nearest node recovery will spread the load of real-time channels more evenly through the network.

For recursive nearest node recovery, the extra delay, D_e consists of the delay between the recovery node, r, and the failed node or link. Similarly the WCRT is the

[4]This assumes that the least cost metric for choosing the recovery path is weighted by the ability of the path to carry additional traffic.

delay on the failure notification channel between the detecting node and the recovery node plus the time to detect the failure. The detection time is assumed to be the worst case delay at the failed node or link in the equation, but it could be a timeout value as discussed in local recovery.

$$D_e = \sum_{i=r}^{n}(d_{i,\tau} + l_i) \tag{66}$$

$$WCRT = \sum_{i=n}^{r}(d_{i,\rho} + l_i) \tag{67}$$

Buffer space must be reserved at r for resending packets. The buffering required is:

$$B_h = \lceil \frac{\sum_{i=r}^{n}(d_{i,\tau} + d_{i,\rho} + 2(l_i))}{x_{min,\tau}} \rceil - P_{max,\tau} \tag{68}$$

7.5.5 Management of Channel Recovery

If any packets must be resent by the recovery node, then the delay and buffering requirements of the backup channel between the recovery node and the destination will be greater than that required by the source under normal flow conditions. As shown in the previous section, the additional delay and buffering required can be calculated based on the recovery scheme used. When $P_{max,\tau} = 0$, the packets which arrive from the source to the recovery node will also see the same delay as the resent packets since they will effectively be blocked (and buffered) by the resent packets to preserve the packet ordering. The buffer in the worst case (when the time between packet arrivals is exactly $x_{min,\tau}$) will be filled by incoming packets from the source at the same rate as it is emptied.

All packets which arrive from the source after the recovery will also experience a constant delay. As a general equation, the buffering needed for all recovery types is:

$$B = \lceil \frac{D_e + WCRT}{x_{min,\tau}} \rceil - P_{max,\tau} \tag{69}$$

The delay for buffered (and resent) packets is:

$$D_{resent} = \sum_{i=1}^{r-1}(d_i + l_i) + \sum_{i=r}^{k}(d_i + l_i) + D_e + WCRT \tag{70}$$

where k is the destination node.

Packets which arrive from the source must wait for the resent packets to be cleared before they can be sent from the recovery node. The recovery node must maintain the specified minimum interarrival time of packets on τ, so these packets experience a delay of:

$$D_{\neg resent} = \sum_{i=1}^{r-1}(d_i + l_i) + \sum_{i=r}^{k}(d_i + l_i) + B * x_{min,\tau} \tag{71}$$

By replacing B in equation 71 with the value in equation 69 the values of D_{resent} and $D_{\neg resent}$ can be shown to be equal where no packet loss can be tolerated. Otherwise, $D_{\neg resent} = D_{resent} - x_{min,\tau} * P_{max,\tau}$.

This effect causes the extra network resources to be reserved for the life of the channel. Where sufficient network and destination resources exist, the channel can relax these initial requirements. A scheme for dynamic management of real-time channels was introduced in [12], however the scheme introduced assumes traffic is being sent on both channels as the transfer of traffic occurs. This is clearly not the case for fault recovery so different requirements apply. One method for relaxing the resource constraints after failure is described below.

Once the channel is being routed along its backup path, the recovery node sends a request along the backup path (between r and the destination) to reduce $x_{min,\tau}$. Each node n along the path from the recovery node to the destination determines the minimum $x_{min,\tau}$ it can allow without affecting the traffic guarantees of any channel using the same output link. This can be calculated as follows (L_l is the transmission rate of link l): [5]

$$x_{min,\tau,l} = \frac{\frac{S_n}{L_l}(t - d_n)}{t - \sum_{i=0}^{n-1}\lceil \frac{t-d_i}{x_{min,i}} \rceil^{+} \frac{S_i}{L_l} + \frac{S_n}{L_l}} \tag{72}$$

[5] $\lceil x \rceil^{+} = n$ if $n - 1 \leq x < n$; $= 0$ if $x < 0$

This equation is derived from the schedulability test of preemptive earliest deadline first scheduling. The reduction of $x_{min,\tau}$ increases the number of packets which may need scheduling at some time t. At time t the total transmission time t_{sched} required by all n channels is:

$$t_{sched,l} = \sum_{i=1}^{n} \lceil \frac{t - d_i}{x_{min,i}} \rceil + \frac{S_i}{L_l} \tag{73}$$

If $t_{sched} \leq t$ then the channels are schedulable. A proof of this property and a bound on the set of times, t, for which the inequality must hold is given in [102]

In addition to the scheduling test, adequate buffering must be available to handle the increased rate. The buffer must hold the packets which can arrive during the delay of the channel $d_{l,\tau}$: $B_l \geq \lceil \frac{d_{l,\tau}}{x_{min,\tau,l}} \rceil$. Therefore:

$$x_{min,B_l} \leq \frac{d_{l,\tau}}{B_l - 1} \tag{74}$$

Since the destination k only receives packets, $x_{min,k}$ will be determined by the buffering constraints at k.

The values of $x_{min,\tau,l}$ are sent to the recovery node, which selects the maximum of the values (i.e. the smallest x_{min} that all l links can accept). A commit request is sent from the recovery node to the destination to reserve the necessary resources for the new value of x_{min}. When acknowledgment is received the recovery node begins sending packets with the minimum interarrival time of x_{min}. If $x_{min} < x_{min,\tau}$, the buffer space reserved for recovery will empty. Once the buffer is empty, the recovery node sends a request to return the minimum interarrival time to $x_{min,\tau}$ and increases the delay from the recovery node to the destination to $D_\tau - D_s$ where D_s is the delay from the source to the recovery node.

All of the management protocol occurs independent of the traffic on τ and therefore is not required to happen in bounded time. The traffic will continue to flow and meet its delay and packet loss requirements regardless of whether the protocol succeeds to clear the buffer or not. In addition to management of active network resources, the management protocol should also send a request to tear down remaining connections which are no longer in use in the failed channel.

7.5.6 Summary

The addition of real-time constraints to wide area communication creates different criteria for fault recovery than those used in non-real-time distributed systems. Dynamic local recovery provides simple, fast recovery from failures in distributed systems but is unable to provide guarantees that packets will still meet their delay requirements. Recovery from other nodes requires the ability of intermediate nodes to receive and act upon fault notification. Whereas this added complexity is not of great benefit to communication with no delay requirements, this section shows it can be essential to successfully re-routing real-time communication.

The end-to-end delay of packets, buffer and bandwidth requirements are all dependent on the method of fault recovery chosen. This section formally defines the additional timing and resource considerations needed for recovery of real-time channels under local, source and recursive nearest node recovery schemes and introduces an algorithm for selecting the recovery point which minimizes resource requirements while guaranteeing no more than a specified number of packets will miss deadlines during recovery due to being dropped or delayed. A management scheme is also introduced for relaxing resource constraints and removing unused channels after recovery has occurred without affecting existing traffic guarantees.

Throughout this section the idea of a least cost path has been used. Some proposals for metrics defining the least cost path have been proposed; but they have not been shown to be optimal [63]. Finding optimal least cost paths is an area of potential future research.

7.6 Conclusions

Supporting real-time channels is a task of resource allocation and management of concurrent processes competing for limited network resources. Chapter 4 dealt with processes which gracefully request resources (i.e. other channels). This chapter has dealt with processes which abruptly take resources (i.e. network faults). When allocating resources for channels, we were able to ignore network traffic that did not impact on the performance guarantees, such as best effort traffic. Similarly, we are able to ignore network traffic that does not impact on the allocation of network resources for back up channels. Knowledge of faults that are independent and will not occur simultaneously has allowed us to introduce a method of providing fault tolerance that significantly reduces both the network resources required to provide fault tolerance and the upper bound on delay for channels. This chapter also formalised the additional delays which occur during fault recovery for different fault recovery protocols and detailed the management of network resources during the recovery process.

This chapter has assumed that the network is rate-controlled and that serialisation does not occur. All sources can send packets simultaneously regardless of shared links. These assumptions were made since the work which this chapter extends also made these assumptions. As shown in chapter 4 and 6 the estimated worst case delay bounds can be significantly improved by considering the affects of serialization and input link constraints. IBR channels and the choice of node recovery are still applicable to the bounds in chapter 4 that include serialisation and link constraints provided that the worst case recovery time (WCRT) calculations and the delay bounds in IBR are modified to follow the calculations in chapter 4. These extensions are an area of future work.

Chapter 8

Conclusions

Applications with timing requirements, such as multimedia, are becoming more prevalent in wide area networks. The popularity of the Internet is leading to an ever increasing number of such real-time applications. Tele-medicine and remote education are often predicted to be potential future applications for the Internet. Both of these applications require at least a minimal level of guaranteed performance to succeed since they involve on-line "live" transmissions which can not be downloaded and played back after download. Best-effort packet transmission can not provide performance guarantees to such applications and can not protect non-real-time applications from the typically high resource demands of these applications.

Overprovisioning in terms of bandwidth is often promoted as the solution to providing QoS [58] [43], and indeed there is evidence that in a network with excess bandwidth resources the queuing delay effects are minimal and all traffic receives very low delay and loss. The Internet 2 project [44] is a good example of the performance that can be achieved in an overprovisioned network. However, the networks that are in general use, rather than limited to a small community, are not being significantly overprovisioned and delay and packet loss are the norm. It is possible that the implementation cost of differentiating traffic may be as high as overprovisioning; but there is no conclusive evidence yet which supports one or the other as the more cost effective approach.

In addition, using overprovisioning as an approach to guaranteed delay requires the available network resources to always be ahead of increasing traffic demands.

This thesis has followed the approach of differentiating between real-time and non-real-time applications and using schedulability analysis to predict worst case delay bounds of traffic. The approach assumes the first come first serve (FCFS) queuing discipline which is currently in use in the Internet. The delay analysis does not require each node to have rate controllers or traffic shapers, thereby reducing the additional hardware/software required to implement the approach. The delay bound computations are mathematically provable, and can be achieved for both constant bit rate and bursty traffic even when the burst rates exceed the link speed.

Unlike previous publications on bounding the maximum delay of FCFS networks, this thesis considers the effect of links on traffic arrival patterns rather than treating all channels as independent and parallel. The idea of *serialisation* is introduced to capture the fact that packets arriving on a single link from different channels form an ordering on that link and do not arrive simultaneously at the next node. This potentially reduces the number of packets that can arrive at a given time, especially in the case of bursty traffic. The relative input and output link rates are also considered since a lower input link rate can also reduce the maximum number of packets arriving at a given time. The inclusion of the combination of these two characteristics in determining packet arrivals gives a much more accurate prediction of actual worst case packet arrivals.

The simulations strongly support the equations. In the worst case single hop scenarios, when packets from all sources begin arriving simultaneously and are as bursty as possible, the backlog calculated by the equations match the simulation within a rounding error of one packet. The equations also result in a lower bound than currently published delay bounds for multi-hop paths. Reducing the bounds as much as possible is critical, since the bounds will affect the amount of guaranteed traffic that can be carried. The work presented in this thesis provides a significant reduction in these bounds. In non-worst case scenarios, the calculations are pessimistic, as one would expect. However, the equations are far less pessimistic than the existing published maximum delay bounds. In the multi-hop simulations of bursty sources the bounds were two orders of magnitude less than those calculated by the FCFS bounds published in [91] and exhibit linear behaviour with respect to traffic load instead of exponential compared to the bounds published in [70].

In addition to providing a far more accurate determination of packet arrivals, rate controllers to reshape the traffic into its original traffic specification at each node are not assumed by the equations. Instead, the effects of *traffic distortion*, where the actual traffic may be made burstier than the original traffic specification due to variable delays at nodes, are included in the equations. The work published in [91] assumes rate control at each hop. It is significant that the delay bounds predicted by the equations without rate control in this thesis are lower than those published assuming rate control, given the assumption of rate control would reduce the worst case burstiness and therefore the worst case delay of the traffic.

There is still room for further improvement and study. The delay bounds for the multi-hop paths could be further reduced by taking into account additional constraints on traffic. For example, only a single packet of one channel will experience the worst case delay at a node during a given worst case traffic arrival pattern. The equations in this thesis assume that all of the channels have experienced the worst case delay at each node, leading to a potentially higher bound on traffic arrival at later nodes in the path. This results in a higher end-to-end bound on traffic arrival than is likely to actually occur in multi-hop channels. So although the bound on traffic arrival is an accurate upper bound, it can be reduced further. More complex theory and equations will be required to capture these effects.

Other areas for improvement exist in the amount of memory needed for storing state information and the cost of computation. The delay bound calculations assume that accurate channel specifications are available at each node. This required state information is likely to cause a memory storage problem at central network nodes. As such, the work presented in this thesis is more appropriate at the edges of the network or within small networks. It is more applicable to the integrated services model rather than the differentiated services model. The problem of storing large amounts of state information can be overcome by treating sections of the network as a single link with a known performance as is done by the differentiated services model. Even with this approach it is worth exploring further whether other source models could be used which retain the lower bounds on traffic arrivals with a reduction in the amount of channel information needing to be stored.

The calculation of the delay bounds, as given in this thesis, grows as $O(L^2)$ in the number of input links, L, into a node and linearly in the number of channels. The calculation time also grows linearly with respect to the number of time points which must be checked for the maximum backlog. The number of links entering a node is likely to be small and static for a node in a wide area network, so this growth rate should not be of great concern. The number of channels at the edges of a network is also likely to be bounded to a relatively small number. The available bandwidth will always limit the number of real-time channels. On a Pentium II 266 Mhz calculating the delay at a node with 50 channels and 50 input links took a total time of .4 seconds for execution. A 10-hop path with nodes of similar computing power would therefore require approximately 4 seconds to calculate minimum end-to-end delay bounds. Although this is probably an acceptable amount of time to wait for the establishment of long lived channels and the processing speed available at the routers is likely to be better than the Pentium II, further study into reducing the number of time points that must be checked and reducing the upper time bound by which the worst case backlog occurs, are likely to lead to improvements in the computation time. This would be important for the establishment of on demand channels (i.e. Internet telephony).

The effects of serialisation and link speed are not limited to FCFS queuing. The strongest contender to FCFS for providing guaranteed delay bounds is weighted fair queuing using the bounds presented by Parekh and Gallagher in [61] based on packet by packet generalised processor sharing (PGPS). This model is supported by the IETF integrated services working group in the guaranteed quality of service network element definition [73]. The work previously published does not take the link effects into account and there is potential to reduce the published bounds by applying the methods developed here. FCFS was chosen as the queuing discipline in this thesis because it was the simplest discipline to examine the ideas on and because bounds based purely on traffic channel characteristics were published. The bounds in PGPS are based on allocation of bandwidth, whereas the FCFS work decouples bandwidth and delay bounds. Since the link effects are closely related to the source characteristics, FCFS is the most direct way of studying these effects. The extension of the link effects to PGPS is an area for further research.

Another contribution of this thesis is a Connection Admission Control (CAC) protocol to establish channels using the equations in this thesis. CACs for many work conserving flow models, such as FCFS without rate control, have not been published with the delay bounds, often because it is not clear how to design the CAC. We know how to determine the minimum delay that a node can guarantee. If this delay is reserved as an upper bound, however, further requests by new channels for resources at the node are likely to be rejected, since the addition of more traffic will increase the minimum delay that could be seen by the existing channels and violate existing guarantees. If the delay reservation is relaxed to allow for additional traffic, this affects not only the delay at the local node but at other nodes downstream by increasing the difference between the minimum and maximum delay (i.e. jitter). This thesis describes a method for distributing slack back to the nodes. The method considers the non-local effects when slack is allocated to a node and determines the amount of slack which can be allocated to the remaining nodes.

Although the CAC introduced solves the problems of determining the capabilities of the network and selecting values for the reservation of resources (in this case the per hop delay to reserve based on the distribution of the slack to the nodes in the path), there are still general open issues that must be dealt with. One known problem is that if the network routing algorithm allows cycles to occur in channel paths, a feedback effect can lead to instability in a network without rate control. One would hope that routing cycles did not occur in a network; but given they can, explicit prevention of cycles in a channel's path may need to be part of the protocol.

Another open issue is the "optimal" distribution of slack. Slack distributed to a node has a multiplying affect on the delay bounds of all nodes in the remainder of the path to the destination. So greater amounts of slack can be distributed at nodes closer to the destination than to the source. However, factors such as the current load on nodes and the fact that central nodes are likely to have heavier loads in general than edge nodes, will also influence how slack should be distributed to best allow the network to carry additional traffic. It is also possible that slack may need to be redistributed as loads change to optimally maximise the utilisation of the network resources. The cost/utilisation trade-offs of methods for selecting the optimal slack allocation is left as an area for further research.

The problems solved above are of the kind where service is guaranteed in the conditions of random traffic requests on shared communication resources. A complementary set of related problems is guaranteeing service in conditions of random failure of shared communication resources. This thesis presents solutions to two problems experienced in providing fault tolerance to channels with guaranteed delay bounds. The first problem is the extra resources which must be reserved along backup channels which are only used in the case of failure. This thesis has proposed the idea of interleaving the backup channels so that resources can be shared when possible. In networks where knowledge about failure rates are known, interleaving backup resources (IBR) can significantly reduce the overhead of providing fault tolerance and allow more real-time traffic to be scheduled. The algorithms and examples of IBR in this thesis assume the presence of rate controllers. This assumption was made because it allowed comparison with published methods which also assumed rate control. Rate control also isolates nodes from effects due to jitter at other nodes, leading to simpler traffic behaviour. Further work can be done to extend the algorithms of IBR to the more complicated case without rate control allowing IBR to be applied more generally.

The second problem of fault tolerance addressed by this thesis is the consideration of the delay and packet loss during recovery in determining whether or not an application can benefit from fault recovery or whether the recovery will be too disruptive to the application itself to warrant fault tolerance. Equations for determining worst case delay in the event of network failure are presented for several network recovery schemes. As with IBR, these equations assume a rate controlled network and further work is needed to extend these equations to be applicable to non-rate controlled networks.

Overall the work presented in this thesis provides an option for supporting integrated service which requires no change to the current queuing model, no rate control requirement, a separation of delay from direct dependence on bandwidth allocation, lower delay bounds and reduced bandwidth and buffering requirements for basic and fault tolerant channels. The ideas on more accurately modeling traffic arrivals are applicable to other queuing models and can be used to extend existing analysis to improve the delay bound guarantees that can be offered by other queuing disciplines.

Appendix A

Additional Equations and Proofs for Chapter 4

Backlog on link l at node k at time t where combined peak channel rate is greater than the output link rate. Combined average channel rates is less than or equal to output link rate.

$$
\begin{aligned}
b_l(t) = \quad & \textstyle\sum_{\forall l' \in L} \min(\sum_{\forall j \in l'} S_{max,j}(\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)), \\
& t * r_l + S_{max,l}) \\
& - \max(\frac{1}{r_l} \min(\textstyle\sum_{\forall j \in l} S_{max,j}(\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)), \\
& t * r_l + S_{max,l}), t) * r_{out}
\end{aligned} \tag{75}
$$

The four bounds on t by which the maximum backlog will have occurred are derived below.

As in Chapter 4 we begin by substituting

$$
\begin{aligned}
\textstyle\sum_{\forall j} S_{max,j} \quad & (\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil))
\end{aligned}
$$

for

$$\begin{aligned}\sum_{\forall l' \in L} \min(\quad & \sum_{\forall j \in l'} S_{max,j}(\lfloor \tfrac{t}{I_j} \rfloor \lceil \tfrac{I_j}{x_{ave,j}} \rceil + \min(\lceil \tfrac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \tfrac{I_j}{x_{ave,j}} \rceil^+) \\ & + \lfloor \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \tfrac{I_j}{x_{ave,j}} \rceil \\ & + \min(\lceil \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \tfrac{I_j}{x_{ave,j}} \rceil)), \\ & t * r_l + S_{max,l})\end{aligned}$$

The argument is that the bits generated by the sources will, in most time intervals, be less than the bits that the link could have sent. So this substitution is likely to be accurate more often. It will always be greater than or equal to the value chosen, so in the cases where the link is limiting the bits generated, we will have a larger bound on t than necessary. A larger t bound is still valid as an upper bound.

Theorem 13 *Let there be L links with J_l channels contending for the same output link at node k. Each channel has the traffic specification of $\{x_{min,j}, S_{max,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out} where $\sum_{j=0}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, if the packets generated on link l is limited by the sources' generation rate and the arrival time of the last generated packet is* not *limited by the link rate, the maximum delay for link l will occur by time:*

$$\begin{aligned} t \geq \quad & (\sum_{\forall j} S_{max,j} \lceil \tfrac{I_j}{x_{ave,j}} \rceil^+ + \sum_{\forall j} S_{max,j}(\lfloor \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \tfrac{I_j}{x_{ave,j}} \rceil \\ & + \min(\lceil \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \tfrac{I_j}{x_{ave,j}} \rceil))) \\ & /(r_{out} - \sum_{\forall j} \tfrac{S_{max,j}}{I_j} \lceil \tfrac{I_j}{x_{ave,j}} \rceil)\end{aligned} \tag{76}$$

Proof

This case equates to choosing the first min value and the second max value.

$$\begin{aligned} b(t) \leq \quad & \sum_{\forall j} S_{max,j}(\lfloor \tfrac{t}{I_j} \rfloor \lceil \tfrac{I_j}{x_{ave,j}} \rceil + \min(\lceil \tfrac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \tfrac{I_j}{x_{ave,j}} \rceil^+) \\ & + \lfloor \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \tfrac{I_j}{x_{ave,j}} \rceil \\ & + \min(\lceil \tfrac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \tfrac{I_j}{x_{ave,j}} \rceil)) - t * r_{out} \quad \leq 0\end{aligned}$$

In the choice $\min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+)$, substitute $\lceil \frac{I_j}{x_{ave,j}} \rceil^+$. Similar to the earlier substitution, this value will always be greater or equal to the actual value chosen. So it may calculate a later bound for t; but this later bound is still a valid upper bound.

$$
\begin{aligned}
b(t) \leq \; & \textstyle\sum_{\forall j} S_{max,j}\big(\frac{t}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil + \lceil\frac{I_j}{x_{ave,j}}\rceil^{+} \\
& +\lfloor\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n})}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil \\
& +\min(\lceil\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n}) \bmod I_j}{x_{min,j}}\rceil, \lceil\frac{I_j}{x_{ave,j}}\rceil)\big) - t * r_{out} \quad \leq 0 \\
b(t) \leq \; & t\textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil + \sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+} \\
& +\textstyle\sum_{\forall j} S_{max,j}\big(\lfloor\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n})}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil \\
& +\min(\lceil\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n}) \bmod I_j}{x_{min,j}}\rceil, \lceil\frac{I_j}{x_{ave,j}}\rceil)\big) - t * r_{out} \quad \leq 0 \\
b(t) \leq \; & t\big(\textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil - r_{out}\big) + \sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+} \\
& +\textstyle\sum_{\forall j} S_{max,j}\big(\lfloor\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n})}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil \\
& +\min(\lceil\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n}) \bmod I_j}{x_{min,j}}\rceil, \lceil\frac{I_j}{x_{ave,j}}\rceil)\big) \quad \leq 0
\end{aligned}
$$

$$
\begin{aligned}
t\big(r_{out} - \textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil\big) \geq \; & \textstyle\sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+} \\
& +\textstyle\sum_{\forall j} S_{max,j}\big(\lfloor\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n})}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil \\
& +\min(\lceil\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n}) \bmod I_j}{x_{min,j}}\rceil, \lceil\frac{I_j}{x_{ave,j}}\rceil)\big)
\end{aligned}
$$

$$
\begin{aligned}
t \geq \; & \big(\textstyle\sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+} + \sum_{\forall j} S_{max,j}\big(\lfloor\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n})}{I_j}\rfloor\lceil\frac{I_j}{x_{ave,j}}\rceil \\
& +\min(\lceil\frac{\sum_{n=1}^{k-1}(d_{max,n}-d_{min,n}) \bmod I_j}{x_{min,j}}\rceil, \lceil\frac{I_j}{x_{ave,j}}\rceil)\big)\big) \\
& /\big(r_{out} - \textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil\big)
\end{aligned}
$$

Theorem 14 *Let there be L links with J_l channels contending for the same output link at node k. Each channel has the traffic specification of $\{x_{min,j}, S_{max,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out} where $\sum_{j=0}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, if the packets generated on link l is limited by the sources' generation rate and the arrival time of the last generated packet is limited by the input link rate, the maximum delay on link l will occur by time*

When $\lceil\frac{t \bmod I_j}{x_{min,j}}\rceil^{+} \leq \lceil\frac{I_j}{x_{ave,j}}\rceil^{+}$

$$
\begin{aligned}
t \ & \geq (\textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& + \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \\
& - \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil))) \\
& /(\frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned} \tag{77}
$$

When $\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+ > \lceil \frac{I_j}{x_{ave,j}} \rceil^+$

$$
\begin{aligned}
t \geq \ & (\textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& + \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& - \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil))) \\
& /(\frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned} \tag{78}
$$

Proof

This case equates to choosing the first min value and the first max value.

$$
\begin{aligned}
b(t) \leq \ & \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{1}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} (\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) * r_{out} \qquad \leq 0
\end{aligned}
$$

To make the proof easier to read, the token *delay_b* is substituted for

$$
\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)
$$

$$\begin{aligned}
b(t) \leq \quad & \textstyle\sum_{\forall j} S_{max,j}(\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + delay_b) \\
& - \textstyle\sum_{\forall j \in l} S_{max,j}(\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + delay_b) * \frac{r_{out}}{r_l} \qquad\qquad\qquad\qquad \leq 0
\end{aligned}$$

The value $\lceil \frac{I_j}{x_{ave,j}} \rceil^+$ is strictly greater than or equal to $\min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+)$. This substitution is made below.

$$\begin{aligned}
b(t) \leq \quad & t \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} * delay_b \\
& - \textstyle\sum_{\forall j \in l} S_{max,j}((\frac{t}{I_j} - 1) \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+, \lceil \frac{I_j}{x_{ave,j}} \rceil^+) \\
& + delay_b) * \frac{r_{out}}{r_l} \qquad\qquad\qquad\qquad \leq 0
\end{aligned}$$

Neither of the choices in the min is strictly less than or equal to the other, so both cases must be considered independently. When $\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+ \leq \lceil \frac{I_j}{x_{ave,j}} \rceil^+$ the following t bound holds.

$$\begin{aligned}
b(t) \leq \quad & t \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} * delay_b \\
& - t * \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^+ \\
& - \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} * delay_b \qquad\qquad\qquad\qquad \leq 0
\end{aligned}$$

0 is a lower bound of $t \bmod I_j$.

$$\begin{aligned}
b(t) \leq \quad & t \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} * delay_b \\
& - t * \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \\
& - \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} * delay_b \qquad\qquad\qquad\qquad \leq 0 \\
b(t) \leq \quad & t(\textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil) + \sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^+ \\
& + \textstyle\sum_{\forall j} S_{max,j} * delay_b \\
& + \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \\
& - \frac{r_{out}}{r_l} \textstyle\sum_{\forall j \in l} S_{max,j} * delay_b \qquad\qquad\qquad\qquad \leq 0
\end{aligned}$$

$$
\begin{aligned}
t(\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil) \quad &\geq \textstyle\sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&+\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\\
&-\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j} * delay\_b
\end{aligned}
$$

$$
\begin{aligned}
t \quad \geq\ &(\textstyle\sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&+\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\\
&-\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j} * delay\_b)/(\frac{r_{out}}{r_l}\sum_{\forall j\in l}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil)
\end{aligned}
$$

If $\lceil\frac{t \bmod I_j}{x_{min,j}}\rceil^{+} > \lceil\frac{I_j}{x_{ave,j}}\rceil^{+}$ the following t bound holds.

$$
\begin{aligned}
b(t) \leq\ & t\textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil + \sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&-\tfrac{1}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j}((\frac{t}{I_j}-1)\lceil\frac{I_j}{x_{ave,j}}\rceil + \lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+delay\_b) * r_{out} & \leq 0\\
b(t) \leq\ & t\textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil + \sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&-t * \tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil + \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&-\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j} * delay\_b & \leq 0
\end{aligned}
$$

$$
\begin{aligned}
b(t) \leq\ & t(\textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil)\\
&+\textstyle\sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&+\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&-\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j} * delay\_b
\end{aligned}
$$

$$
\begin{aligned}
t(\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \quad & \textstyle\sum_{\forall j}\frac{S_{max,j}}{I_j}\lceil\frac{I_j}{x_{ave,j}}\rceil) \geq \sum_{\forall j} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&+\textstyle\sum_{\forall j} S_{max,j} * delay\_b\\
&+\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil - \frac{r_{out}}{r_l}\sum_{\forall j\in l} S_{max,j}\lceil\frac{I_j}{x_{ave,j}}\rceil^{+}\\
&-\tfrac{r_{out}}{r_l}\textstyle\sum_{\forall j\in l} S_{max,j} * delay\_b
\end{aligned}
$$

$$
\begin{aligned}
t \geq \quad & (\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} + \sum_{\forall j} S_{max,j} * delay\_b \\
& + \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& - \frac{r_{out}}{r_l} \sum_{\forall j \in l} S_{max,j} * delay\_b) \\
& /(\frac{r_{out}}{r_l} \sum_{\forall j \in l} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil - \sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned}
$$

Theorem 15 *Let there be L links with J_l channels contending for the same output link at node k. Each channel has the traffic specification of $\{x_{min,j}, S_{max,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out} where $\sum_{j=0}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, if the packets generated on link l is limited by the input link rate and the arrival time of the last generated packet is* not *limited by the input link rate, the maximum delay on link l will occur by time*

$$
\begin{aligned}
t \geq \quad & (\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& + \sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil))) \\
& /(r_{out} - \sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned}
$$

Proof

This case equates to choosing the second min value and the second max value.

$$
\begin{aligned}
b(t) \leq \quad & \sum_{\forall j} S_{max,j} (\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^{+}, \lceil \frac{I_j}{x_{ave,j}} \rceil^{+}) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) - t * r_{out} \quad \leq 0
\end{aligned}
$$

Note that this equation is identical to that in the proof of theorem 13. The proof is therefore the same.

Theorem 16 *Let there be L links with J_l channels contending for the same output link at node k. Each channel has the traffic specification of $\{x_{min,j}, S_{max,j}, x_{ave,j}, I_j\}$. Given an output rate of r_{out} where $\sum_{j=0}^{J}(S_{max,j}/x_{ave,j}) \leq r_{out}$, if the packets generated on link l is limited by the input link rate and the arrival time of the last generated packet is limited by the input link rate, the maximum delay on link l will occur by time*

$$
\begin{aligned}
t \geq\ & (\textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{S_{max,l}}{r_l} * r_{out}) \\
& /(r_{out} - \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned} \tag{79}
$$

Proof

This case equates to choosing the second min value and the first max value.

$$
\begin{aligned}
b(t) \leq\ & \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{t}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil + \min(\lceil \frac{t \bmod I_j}{x_{min,j}} \rceil^{+}, \lceil \frac{I_j}{x_{ave,j}} \rceil^{+}) \\
& + \lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{1}{r_l}(t * r_l + S_{max,l}) * r_{out} && \leq 0 \\
b(t) \leq\ & t \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - t * r_{out} - \frac{S_{max,l}}{r_l} * r_{out} && \leq 0 \\
b(t) \leq\ & t(\textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil - r_{out}) + \textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{S_{max,l}}{r_l} * r_{out} && \leq 0
\end{aligned}
$$

$$
\begin{aligned}
t(r_{out} - \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil) \geq\ & \textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil^{+} \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1}(d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{S_{max,l}}{r_l} * r_{out}
\end{aligned}
$$

$$
\begin{aligned}
t \geq \quad & (\textstyle\sum_{\forall j} S_{max,j} \lceil \frac{I_j}{x_{ave,j}} \rceil + \\
& + \textstyle\sum_{\forall j} S_{max,j} (\lfloor \frac{\sum_{n=1}^{k-1} (d_{max,n} - d_{min,n})}{I_j} \rfloor \lceil \frac{I_j}{x_{ave,j}} \rceil \\
& + \min(\lceil \frac{\sum_{n=1}^{k-1} (d_{max,n} - d_{min,n}) \bmod I_j}{x_{min,j}} \rceil, \lceil \frac{I_j}{x_{ave,j}} \rceil)) \\
& - \frac{S_{max,l}}{r_l} * r_{out}) \\
& /(r_{out} - \textstyle\sum_{\forall j} \frac{S_{max,j}}{I_j} \lceil \frac{I_j}{x_{ave,j}} \rceil)
\end{aligned}
$$

The bounds calculated by Theorems 13 and 15 are identical. The bound calculated by Theorem 16 is strictly less than that calculated by Theorem 13. Therefore, it is sufficient to choose the maximum of the bounds calculated by Theorems 13 and 14.

Appendix B

Algorithm for Calculating Maximum Delay

The algorithm on the following page gives the general abstracted algorithm for determining the worst case delay at a node. The details on how the calculations are made are described in Chapter 4.

Algorithm 2 General Algorithm for Calculating Maximum Delay at Node

---

Require: $Backlog > 0$

$tBound \leftarrow \min tBounds$ {get the $tBound$, this will be the minimum upper bound on t as calculated by the equations in Chapter 4.}

while $t \leq tBound$ and $Backlog > 0$ **do**

  $A(t) \leftarrow 0$

  for all $l \in InputLinks$ **do**

    if $SumOfInputRates \neq r_{out}$ **then**

      $A_{src,l} \leftarrow 0$

      for all $j \in l$ **do**

        $A_j(t) \leftarrow (gen_j(t)+delayed_{j,k})*S_{max,j}$ {$gen_j(t)$ and $delayed_{j,k}$ are calculated using the applicable equations from Chapter 4}

        $A_{src,l} \leftarrow A_{src,l} + A_j(t)$

      end for

      $A_l(t) \leftarrow \min(A_{src,l}, (t - idle(t)) * r_l + S_{max,l})$

      $A(t) \leftarrow A(t) + A_l(t)$

      $ArrivalTime \leftarrow \max(A_l(t)/r_l, t)$

      $Backlog \leftarrow A(t) - ArrivalTime * r_{out}$

    else

      $Backlog \leftarrow SumOfLinksInitialBursts - InitialBurst_l - delayed_l$ {This *if* statement tests for the case where equation does not have a solution (i.e. total input rate equals output rate) as discussed in Chapter 4. In this case the maximum backlog occurs during the initial burst and is then sustained for all times.}

    end if

    $Delay \leftarrow Backlog/r_{out}$

  end for

  $t \leftarrow EndOfNextBusyPeriod$

end while

Appendix C

Resource Overhead of IBR in a Mesh

Let $N = (V, E)$ represent a planar network with nearest neighbour connections, where V is the set of nodes and E is the set of links. Examples of planar networks are the regular mesh and network constructed from the map of the USA with all nearest neighbour major towns interconnected together.

Given n nodes in the network N, the maximum number of channels connecting any node in the network to any other node is approximately n^2 and the average number of hops in a channel is approximately $\sqrt{n}$. The number of node crossings by all basic channels is approximately equal to:

$$R_p \approx n^2\sqrt{n}$$

From 1 we can derive the number of node crossings by all basic channels at any one node:

$$X_n = \frac{R_p}{N} \approx n\sqrt{n}$$

Therefore the removal of a particular node will require X_n detour channels to be created.

Multiplying X_n by the average number of nodes in a detour channel gives the resource units needed to supply the detour channels:

$$R_b \approx n\sqrt{n}^2$$

The percentage overhead is:

$$O_{IBR} = \frac{R_b}{R_p} \approx n\frac{\sqrt{n}^2}{n^2\sqrt{n}} = \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}\%$$

For example, in a network of size $n = 100$ the overhead of detour channels in an IBR network is only $O_{IBR} = \frac{1}{\sqrt{100}} = 10\%$ which is only a small fraction of the overhead in a SFI network which in the best case is 100%.

Bibliography

[1] G. Agrawal, B. Chen, W. Zhao, and S. Davari. Guaranteeing synchronous message deadlines with the timed token medium access control protocol. *IEEE Trans. Computers*, 43(3):327–339, March 1994.

[2] C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne. Real-time communication in packet-switched networks. *Proceedings of the IEEE*, 82(1):122–139, January 1994.

[3] K. Arvind, K. Ramamritham, and J.A. Stankovic. A local area network architecture for communication in distributed real-time systems. *Journal of Real-Time Systems*, 3(2):115–147, May 1991.

[4] T. Au and H. Mehrpour. Real-time communications in broadband integrated networks. In *Proceedings of the Australian Telecommunication Networks and Applications Conference*, pages 869–874, December 1994.

[5] A. Banerjea. Queueing delays in rate controlled ATM networks. In *Proceedings of the IEEE INFO-COM'93*, pages 547–556, April 1993.

[6] A. Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channel. In *Proceedings of SIGCOMM*, pages 194–205, October 1996.

[7] A. Banerjea. Fault recovery for guaranteed performance communications connections. *IEEE/ACM Transactions on Networking*, 7(5):653–668, October 1999.

[8] Anindo Banerjea. *Fault Management for Realtime Networks.* PhD thesis, University of California at Berkeley, 1994.

[9] J. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of INFOCOM'96*, pages 120–127, March 1996.

[10] J. Beran, R. Sherman, M.S. Taqqu, and W. Willinger. Long-range dependence in variable bit rate video traffic. *IEEE Transactions on Communications*, 43:1566–1579, Feb/Mar/April 1995.

[11] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) – version 1 functional specification. *Internet Request for Comments: 2205, Standards Track*, September 1997.

[12] H. Zhang C. Parris and D. Ferrari. A dynamic management scheme for real-time connections. In *Proceedings of IEEE INFOCOM'94*, pages 698–707, April 1994.

[13] C. C. Chou and K. G. Shin. Multiplexing statistical real-time channels on a multiaccess network. In *Proceedings of the 16th Int'l Conference on Distributed Computing Systems*, pages 133–140, May 1996.

[14] C. C. Chou and K. G. Shin. Statistical real-time channels on multiaccess bus networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):769–780, August 1997.

[15] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Trans. on Networking*, 6(4):362–373, August 1998.

[16] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanisms. In *Proceedings of SIGCOMM*, pages 14–26, August 1992.

[17] J. Cobb and M. Gouda. Flow theory. *IEEE/ACM Transactions on Networking*, 5(5):661–674, October 1997.

[18] R. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[19] R. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[20] Martin de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN*. Ellis Horwood, 1991.

[21] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of SIGCOMM*, pages 1–12, September 1989.

[22] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependant packet traffic. *IEEE/ACM Transactions on Networking*, 4(2):209–223, April 1996.

[23] D. Ferrari and D.C. Verma. A scheme for real-time channel establishment in wide area networks. *IEEE Journal on Selected Areas in Communication*, 8(3):368–379, April 1990.

[24] Domenico Ferrari. Real-time communication in an internetwork. Technical Report TR-92-001, International Computer Science Institute, Berkeley, California, 1992.

[25] N. Figueira and J. Pasquale. Leave-in-time: A new service discipline for real-time communications in a packet-switching network. In *Proceedings of SIGCOMM*, pages 207–218, August 1995.

[26] N.R. Figueira and J. Pasquale. An upper bound on virtualclock service discipline. *IEEE/ACM Transactions on Networking*, 3(4):399–408, August 1995.

[27] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[28] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[29] L Georgiadis, R. Guerin, and A. Parekh. Optimal multiplexing on a single link: Delay and buffer requirements. In *Proceedings of INFOCOM'94*, pages 524–532, April 1994.

[30] L Georgiadis, R. Guerin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[31] L Georgiadis, R. Guerin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. In *Proceedings of INFOCOM'96*, pages 102–110, March 1996.

[32] S. Golestani. Congestion free communication in high speed packet networks. *IEEE Transactions on Communications*, 39:1802–1812, December 1991.

[33] S. Golestani. A framing strategy for congestion management. *IEEE Journal on Select Areas of Communications*, 9(7):1064–1077, September 1991.

[34] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of INFOCOM'94*, pages 636–646, April 1994.

[35] P. Goyal, S. Lam, and H. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proceedings of the Fifth International Workshop on Network and OS Support for Digital Audio and Video*, pages 287–298, April 1995.

[36] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: A framework. Technical Report TR-95-30, University of Texas at Austin, Department of Computer Science, 1995.

[37] P. Goyal and H. Vin. Generalized guaranteed rate scheduling algorithms: A framework. *IEEE/ACM Transactions on Networking*, 5(4):561–571, August 1997.

[38] R. Guerin and A. Orda. QoS routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350–364, June 1999.

[39] A. Gupta and D. Ferrari. Resource partitioning for real-time communication. *IEEE/ACM Transactions on Networking*, 3(5):501–508, October 1995.

[40] M. Hamdaoui and P. Ramanathan. Selecting timed token protocol parameters to guarantee real-time messages. In *Proceedings of the Parallel and Distributed Real-Time Systems Workshop*, April 1993.

[41] Rainer Händel and Manfred Huber. *Integrated Broadband Networks: An Introduction to ATM-based Networks.* Addison Wesley, 1991.

[42] C. Hedrick. Routing information protocol. *Internet RFC 1058*, June 1988.

[43] G. Huston. Quality of service - fact or fiction? *The Internet Protocol Journal, Cisco Systems*, 3(1):27–34, 2000.

[44] Internet 2 project home page. http//www.internet2.edu.

[45] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *Proceedings of SIGCOMM*, pages 2–12, August 1995.

[46] G. Karlsson. Asynchronous transfer of video. *IEEE Communications Magazine*, 34(8):118–126, August 1996.

[47] S. Keshav. Real 5.0 overview. http://www.cs.cornell.edu/skeshav/real/overview.html.

[48] S. Keshav. *An Engineering Approach to Computer Networking*. Addison Wesley, 1997.

[49] B. Krishnamurthy and J. Rexford. The web over IP networks: Protocols and networks. tutorial, SIGCOMM 1999.

[50] S. Lam and G. Xie. Burst scheduling: Architecture and algorithm for switching packet video. In *Proceedings of INFOCOM'95*, pages 940–950, April 1995.

[51] R. Landry and I. Stavrakakis. Study of delay jitter with and without peak rate enforcement. *IEEE/ACM Transactions on Networking*, 5(4):543–553, August 1997.

[52] J. Liebeherr, D. Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885–901, December 1996.

[53] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real time environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[54] D. Lorenz and A. Orda. QoS routing in networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, 6(6):768–778, December 1998.

[55] Q. Ma and P. Steenkiste. Quality of service routing for traffic with performance guarantees. In *Proceedings of the 5th International Workshop on Quality of Service (IWQOS '97)*, pages 115–126, 1997.

[56] N. Malcolm and W. Zhao. Guaranteeing synchronous messages with arbitrary deadline constraints in an FDDI network. In *Proceedings of the Conference on Local Computer Networks*, pages 186–195, September 1993.

[57] J. Moy. OSPF version 2. *Internet RFC 1583*, March 1994.

[58] A. Odlyzko. The internet and other networks: Utilization rates and their implications. *Information Economics & Policy*, 12:341–365, 2000.

[59] A. Orda. Routing with end-to-end QoS guarantees in broadband networks. *IEEE/ACM Transactions on Networking*, 7(3):365–374, June 1999.

[60] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[61] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):344–357, April 1994.

[62] A.K. Parekh. *A generalized processor sharing approach to flow control in integrated services networks*. PhD thesis, Massachusetts Institute of Technology, 1992.

[63] C. Paris and D. Ferrari. A dynamic connection management scheme for guaranteed performance services in packet-switching integrated services networks. Technical Report TR-93-005, International Computer Science Institute, Berkeley, California, 1993.

[64] Colin J. Parris and Anindo Banerjea. An investigation into fault recovery in guaranteed performance service connections. In *Proceedings of SUPERCOMM/ICC*, pages 175–181, March 1994.

[65] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. In *Proceedings of SIGCOMM*, pages 257–268, September 1994.

[66] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 1996.

[67] Cheryl Pope and Jay Yantchev. Performance evaluation of fault-tolerant routing protocols for real-time channels. In *Proceedings of the Australasian Workshop on Parallel and Real-Time Systems*, pages 305–313, 1994.

[68] Cheryl Pope and Jay Yantchev. Fault-tolerant real-time communication with reduced resource overhead. *Australian Computer Science Communications*, 17(1):441–448, February 1995.

[69] R. Procter, M. Hartswood, A. McKinlay, and S. Gallacher. An investigation of the influence of network quality of service on the effectiveness of multimedia communication. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 160–168, 1999.

[70] Prepat Pusopa. Real-time communication in ATM networks (guaranteed deadlines in ATM networks). Master's thesis, The University of Adelaide, 1995.

[71] Prepat Pusopa, Cheryl Pope, and Jay Yantchev. Real-time communication in ATM networks. Technical Report TR96-06, Adelaide University, 1996.

[72] N. Rao and S. Batsell. Algorithm for minimum end-to-end delay paths. *IEEE Communications Letters*, 1(5), September 1997.

[73] R. Guerin S. Shenker, C. Partridge. Specification of guaranteed quality of service. *Internet RFC 2212*, September 1997.

[74] A. Burns S. Zhang. A study of timing properties with the timed token protocol. Technical Report YCS-94-243, Real Time Systems Research Group, Department of Computer Science, University of York, 1994.

[75] S. Shenker and L. Breslau. Two issues in reservation establishment. In *Proceedings of SIGCOMM*, pages 14–26, August 1995.

[76] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in computer networks: Reshaping the research agenda. *Computer Communication Review*, 26(2):19–43, July 1996.

[77] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison Wesley, 1994.

[78] J.K. Strosnider and T.E. Marchok. Responsive, deterministic IEEE 802.5 token ring scheduling. *Journal of Real-Time Systems*, 1(2):133–158, 1989.

[79] K. Thompson, G. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, November/December 1997.

[80] J. Turner. New directions in communications, or which way to the information age? *IEEE Communications Magazine*, 24(4):8–15, 1986.

[81] C. Venkatramani and T. Chiueh. Design, implementation, and evaluation of a software-based real-time ethernet protocol. In *Proceedings of SIGCOMM*, pages 27–37, August 1995.

[82] D. Verma, H. Zhang, and D. Ferrari. Delay jitter control for real-time communication in a packet switching network. In *Proceedings of TriComm*, 1991.

[83] World Wide Web Consortium (W3C). About the world wide web. http://www.w3.org/WWW.

[84] G. Xie and S. Lam. Delay guarantee of virtual clock server. *IEEE/ACM Transactions on Networking*, 3(6):683–689, December 1995.

[85] David Yates, James Kurose, Don Towsley, and Michael Hluchyj. On per-session end-to-end delay and the admission problem for real-time applications with QoS requirements. In *Proceedings of SIGCOMM*, pages 2–12, September 1993.

[86] David Yates, James Kurose, Don Towsley, and Michael Hluchyj. On per-session end-to-end delay and the admission problem for real-time applications with QoS requirements. Technical Report 93-20, Department of Computer Science, University of Massachusetts at Amherst, May 1994.

[87] H. Zhang. *Service disciplines for Integrated Services Packet-Switching Networks*. PhD thesis, University of California at Berkeley, 1993.

[88] H. Zhang. Providing end-to-end performance guarantees using non-workconserving disciplines. *Computer Communications Journal*, 18(10), October 1995.

[89] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10), October 1995.

[90] H. Zhang and D. Ferrari. Rate controlled static priority queueing. In *Proceedings of IEEE INFOCOM'93*, pages 227–236, 1993.

[91] H. Zhang and D. Ferrari. Improving utilization for deterministic service in multimedia communication. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 295–304, May 1994.

[92] H. Zhang and D. Ferrari. Rate-controlled service disciplines. *Journal of High Speed Networks*, 3(4):389–412, 1994.

[93] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of SIGCOMM*, pages 113–121, August 1991.

[94] H. Zhang and E. Knightly. RCSP and stop-and-go: A comparison of two non-work-conserving disciplines for supporting multimedia communication. *Multimedia Systems Journal*, 4(6):346–356, December 1996.

[95] L. Zhang. Virtualclock: A new traffic control algorithm for packet switching networks. In *Proceedings of SIGCOMM*, pages 19–29, August 1990.

[96] L. Zhang. Virtualclock: A new traffic control algorithm for packet switching networks. *ACM Transactions on Computing Systems*, 9(2):101–124, May 1991.

[97] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Networks Magazine*, 31:8–18, Sept 1993.

[98] Q. Zhang, C. Wolf, S. Daijavad, and M. Tourma. Talking to customers on the web: A comparison of three alternatives. In *Proceedings of the ACM Conference on Computer supported cooperative work*, pages 109–117, 1998.

[99] S. Zhang and A. Burns. An optimal synchronous bandwidth allocation scheme for guaranteeing synchronous message deadlines with the timed-token MAC protocol. *IEEE/ACM Trans. on Networking*, 3(6):729–741, December 1995.

[100] Q. Zheng and K. G. Shin. Establishment of isolated failure immune real-time channels in harts. *IEEE Transactions on Parallel and Distributed Systems*, 6(2):113–119, February 1995.

[101] Qin Zheng. *Real-Time Fault-Tolerant Communication in Computer Networks.* PhD thesis, University of Michigan, 1993.

[102] Qin Zheng and Kang G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(2/3/4):1096–1105, March 1994.

[103] Qin Zheng and Kang G. Shin. Fault-tolerant real-time communication in distributed computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 9(5):470–480, 1998.