A Linear Control Approach to ATM Congestion Control

by

Randall A. Berry

B.S., Electrical Engineering (1993)
University of Missouri-Rolla

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Signature of Author .....
Department of Electrical Engineering and Computer Science
May 7, 1996

Certified by .....
Charles E. Rohrs
Visiting Scientist
Department of Electrical Engineering and Computer Science
Fellow, Tellabs, Inc.
Thesis Supervisor

Accepted by .....
F. R. Morgenthaler
Chairman, Department Committee on Graduate Students

A Linear Control Approach to ATM Congestion Control

by

Randall A. Berry

Submitted to the Department of Electrical Engineering and Computer Science on May 10, 1996, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

Asynchronous transfer mode (ATM) is a networking protocol intended to support applications with a variety of different service requirements. Traffic management is needed to ensure that the network's resources are not overloaded and that the traffic receives its desired service. One proposed method of traffic management for non-real time, data traffic is 'rate based' congestion control. In this approach the network feeds back congestion information to the data sources, which then modify their transmission rates accordingly. This thesis addresses the rate based congestion control framework used for ATM networks, and specifically looks at rate based congestion control from the perspective of linear control theory. Simple congestion control algorithms are developed and the trade-offs inherent in such schemes are analyzed.

Thesis Supervisor: Charles E. Rohrs
Title: Visiting Scientist, MIT & Research Fellow, Tellabs, Inc.

Acknowledgments

I would like to thank Dr. Charles Rohrs for the time and advise he gave to me in working on this thesis. Many of the ideas within originated from him. I would also like to thank Jim Mills and Phil Lin of Tellabs for their time and many helpful discussions; and thanks to Steve O'Halek, who also did research related to this topic. Finally, I would like to thank everyone at LIDS for making MIT an environment in which it is enjoyable to work, and my parents who have always been there for me.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In 1988, Asynchronous Transfer Mode (ATM) was accepted by the Consultive Committee on International Telecommunication and Telegraph (CCITT) as the transfer mode it intended to use for the broadband integrated services network (B-ISDN) [Cci88]. B-ISDN is intended to be a high speed network capable of supporting a wide variety of applications including voice, video, and data on a single network. ATM requires that all information entering the network be broken down into fixed size cells. The cells from various applications are then multiplexed together based on their needed service and the available resources. Originally this integration of various types of traffic was the impetus behind ATM standards, but the current view is that ATM will first be adopted in high speed data network applications [Kav95], particularly in interconnecting high speed LAN's. Most of the current development of ATM protocols has been undertaken by the ATM Forum, a group consisting mainly of equipment vendors who are trying to accelerate the adoption of ATM products and services [Gor95].

ATM networks are connection-oriented. Before a message can be sent over the network, the source is required to inform the network about the type of traffic it wants to send. If the resources are available then the network establishes a path from the source to the destination. This path is called a virtual circuit (VC) in an ATM network.

The traffic expected on an ATM network can be classified into two classes: guaranteed and best effort [New94]. Guaranteed traffic generally consists of real time traffic which needs strict requirements on bandwidth and delay from the network. Voice is an example of this type of traffic. ATM supports guaranteed traffic with the constant bit rate (CBR)

and variable bit rate (VBR) service classes. Best-effort traffic is primarily data network type of traffic which has looser service requirements from the network. For this type of traffic it is desirable to dynamically allocate the available bandwidth among all the best-effort traffic. The ATM service class for this traffic is called available bit rate (ABR) service.

Traffic management is needed to ensure that the network is not overloaded with data resulting in congestion. The different classes of traffic in an ATM network require different approaches to traffic management. Admission control and traffic shaping are used to manage guaranteed traffic. When a connection is set up, information pertaining to the required bandwidth and delay are given to the network. If the network does not have the necessary resources available, the connection is refused. After accepting a connection the network then polices the user to make sure that the negotiated parameters are not violated. It is nearly impossible to *a priori* define strict service parameters for ABR traffic [New94]; thus, a different method of traffic control is required. A closed-loop feedback control scheme has been proposed for ABR traffic. In such a scheme the network relays congestion information to the data sources, which then modify their behavior accordingly.

The ATM Forum has decided upon a rate-based approach to congestion control, meaning that the data sources modify their transmission rates based on feedback from the network. This approach was selected over a credit based approach, in which the sources modify the number of cells they launch into the network in response to congestion information relayed as credits [KuC93], [KuM95].

The ATM Forum has proposed guide-lines on how the network should communicate congestion information to the traffic sources and how the sources should behave upon receiving this information. A fixed algorithm will not be proposed, this being left up to individual vendors. Sample algorithms have been given as examples of possible implementations. The algorithms presented to date consist largely of heuristic approaches. These contain significant non-linearities, making any detailed analysis of them difficult.

Rate-based congestion control is a feedback control system as shown in Figure 1.1. As such it seems natural to apply the body of linear control theory to this problem as sug-

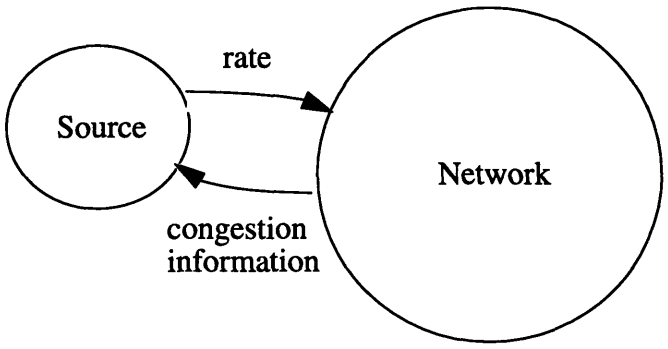


Figure 1.1: Rate-based feed-back system.

gested in [RoB96]. This approach restricts one to a linear algorithm, but in doing so the behavior becomes much more amenable to analysis.

In this thesis a linear control approach to rate-based congestion control is discussed. This approach allows simple congestion control algorithms to be developed and analyzed. The trade-offs and limitations of this approach are discussed, and these are compared to the nonlinear algorithms developed by the ATM Forum.

1.2 Goals of Rate Based Congestion Control

Any congestion control scheme for ABR traffic must consider several, sometimes opposing, criteria. These can be summarized by four guidelines:

(1) Use available bandwidth to the fullest possible extent.

(2) Keep queue sizes as small as possible and avoid cell loss.

(3) Treat all sources 'fairly'.

(4) Keep the algorithm as simple as possible.

Using as much of the available bandwidth as possible is a fundamental goal of ABR congestion control. Any unused bandwidth means that network resources are not being used to their fullest extent. It may be desirable to design a congestion control scheme that uses less than the available bandwidth for improved performance in some of the other areas.

If a cell arrives at a queue that is full then obviously either that cell or another cell must be dropped. If the dropped cell is part of a packet in a higher layer protocol, like TCP/IP, then the entire higher layer packet must be retransmitted. As shown in [RoF95] this can lead to serious network congestion. A goal of ABR service is to ensure that the chance of a cell being dropped is small. Maintaining short queue lengths is one way to reduce the probability of a cell loss. Keeping the queue sizes small also helps keep the round trip delay in the network short; this enables feedback information to reach the sources more quickly.

It is desirable that a congestion control scheme allocates the available bandwidth fairly among all users of the network. Fair is usually taken to mean fair in the max-min sense [BeG92]. This requires that all bottleneck links at a given switch get an equal share of the available bandwidth. A bottleneck link is a link whose rate is not limited by congestion at any other switches. Also related to the issue of fairness is the rate at which the algorithm responds. An algorithm that responds too slowly would be 'unfair' to newer sources, for it would take them too long to get their fair share of the bandwidth.

Keeping any congestion control algorithm simple is very important in an ATM network. The high speeds at which an ATM network is expected to operate require that the amount of processing done on each cell is kept to a minimum. Congestion control schemes that require statistics be kept for each source destination pair, or that require separate queues for each pair are not acceptable due to the amount of processing that would be required. This is one of the main reasons that the credit-based approach was not adopted.

1.3 Outline of Thesis

This thesis discusses the use of linear control theory in developing rate-based congestion control algorithms that meet the criteria in the proceeding section. The next chapter discusses the algorithms and the framework being developed by the ATM Forum. These are presented to demonstrate the non-linearities present in the current schemes and to serve as a basis for comparison. The algorithms developed here attempt to stay as close to

the ATM Forum's guidelines when possible, but, if it appears that modifying this framework may lead to better algorithms, such modifications are examined.

Chapter 3 presents a linear control algorithm that uses only binary feedback. Simulations of this algorithm are analyzed and its performance is analyzed. Chapter 4 discusses linear algorithm that allows a field to be fed back. Several algorithms are discussed and simulations are also presented. Chapter 5 compares these approaches and those of the ATM forum and discusses directions of further research.

Chapter 2

Overview of ATM Forum Congestion Control Schemes

2.1 Background

While not defining a specific algorithm for rate-based congestion control, the ATM Forum is developing standards within which any proposed algorithm must operate. These are intended to allow flexibility in choosing a particular switch algorithm, while insuring that the various algorithms interoperate. The ATM Forum is also considering several example algorithms to demonstrate the viability of the protocols it is establishing.

The standards that have been developed represent a combination of two approaches to rate-based control which we shall refer to as the binary approach and the explicit rate approach. In the binary approach the source receives a single bit of information representing whether or not there is congestion in the network. In the explicit rate approach the source receives from the network an explicit rate that represents the maximum rate at which it should transmit to avoid congestion. The use of an explicit rate requires a more complicated algorithm at the sources but results in better performance.

The standards specify how congestion information is communicated through the network and how the source and destination treat this information when they receive it. Very general guidelines on how a switch generates this information are also included. Congestion information can be communicated through the network in two ways. One is through the use of a resource management (RM) cell. These are special ATM cells which the source is required to generate periodically. They contain information on the current rate the source is transmitting and the minimum acceptable rate for a source. The switches may use this information to determine what congestion information the source should receive. The RM cells also contain a congestion indicator (CI) bit which is used to relay

binary congestion information, and an explicit rate (ER) field which is used for explicit rate type of algorithms. The RM cells follow the same path through the network as the data cells. When they reach the destination they are turned around and sent back to the source along the same path. As they traverse the network, switches can put congestion information into either the CI or ER fields. The second method for conveying congestion information is only applicable for binary schemes. This consists of using the explicit forward congestion indicator (EFCI) bit which resides in the header of all ATM data cells. A switch can set this bit if it is congested. The destination is required to monitor all incoming data cells of a virtual circuit to see if the EFCI bit is set and, if so, to notify the source via the CI bit in an RM cell.

In order to understand the proposed source and destination behaviors, the characteristics of binary and explicit rate algorithms are examined in the next two sections. The last section of this chapter then summarizes the source and destination behaviors being considered by the ATM Forum.

2.2 Binary Algorithms

The initial rate-based algorithms considered by the ATM Forum uses binary feedback. In such an algorithm the feedback information received by the source only conveys that congestion is present or that congestion is not present. A switch is usually considered congested when it has a queue whose length exceeds a threshold. When a switch is congested it conveys this information to the data source. One method for doing this is for the switch to set the EFCI bit in all data cells passing through it. When the destination receives a data cell with EFCI set then it is required to set the CI bit in the next RM cell before sending it back to the source. This method is called forward explicit congestion notification (FECN). In a second method, called backward explicit congestion notification (BECN), the congested switch itself sets the CI bit in RM cells as they pass through it returning to the source. This shortens the feedback delay, but requires a more complicated switch.

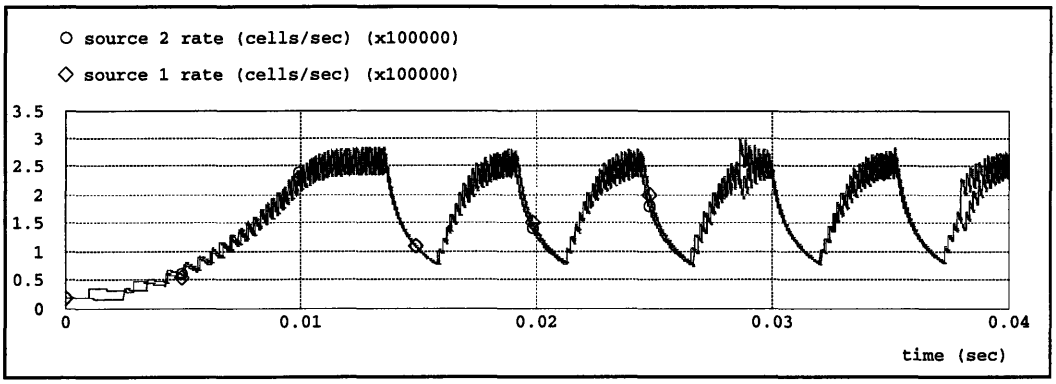
Early rate-controlled schemes [2] have the source increase its rate unless it receives a RM cell with the CI bit set. Only when receiving notification of congestion would it

reduce its rate. The problem with this approach is that if severe congestion occurs and RM cells are lost then the source would still increase its rate, adding to the congestion and possibly causing the network to collapse. To remedy this, the algorithm was changed to use positive feedback, meaning that the source would continually decrease its rate until it receives notification that congestion is not present, in which case it is allowed to increase.

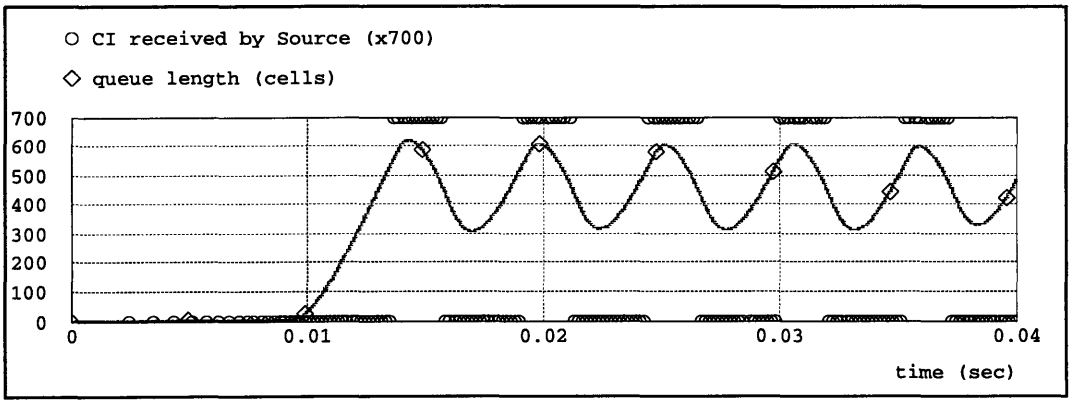
One framework currently being considered by the ATM Forum is an extension of a binary algorithm called the proportional rate control algorithm (PRCA) [BoF95]. This is an algorithm that uses positive feedback and can use either FECN or BECN. In PRCA, the source is required to send an RM cell for every *Nrm* data cells that it sends. When it sends an RM cell, the source multiplicatively decreases it rate. If it receives an RM cell with the CI bit not set then the source removes the effect of the last decrease and increases its rate by an additive factor. If the source receives an RM cell with the CI bit set then it takes no additional action.

The method of rate adjustment used in PRCA makes analysis of this algorithm difficult. When the source is increasing its rate, the additive increases are occurring at a rate proportional to the rate it is receiving RM cells back; when the source is decreasing its rate, the multiplicative decreases occur at a rate proportional to the rate it is sending RM cells. The rates that RM cells are being sent out and received need not be the same at any instant since returning RM cells were generated some time ago when the source's rate may have been lower or higher. Also, the time instants that adjustments in the rate are made depends on the rate itself. This makes a discrete time analysis difficult, for not only is the time between sample points constantly changing, but the increases and decreases are occurring on different time scales.

Simulation results of a PRCA algorithm using BECN is shown in Figure 2.1. The parameters used in the algorithm are the same as in the results presented to the ATM Forum in [Bar94]. The network used for this simulation is shown in Figure 2.2. The two sources are competing for the same output port of the switch which has a bandwidth of 3.54×10^5 cells/sec. The two sources are persistent, *i.e.* they always have data to send, and both start at the same rate. Ideally, each source should be allowed to transmit at half the available bandwidth or 1.77×10^5 cells/sec. Rather than converging to this value, both



Rates of the two sources.



The queue length and the CI value received by source 1

Figure 2.1: Simulation Results for PRCA method.

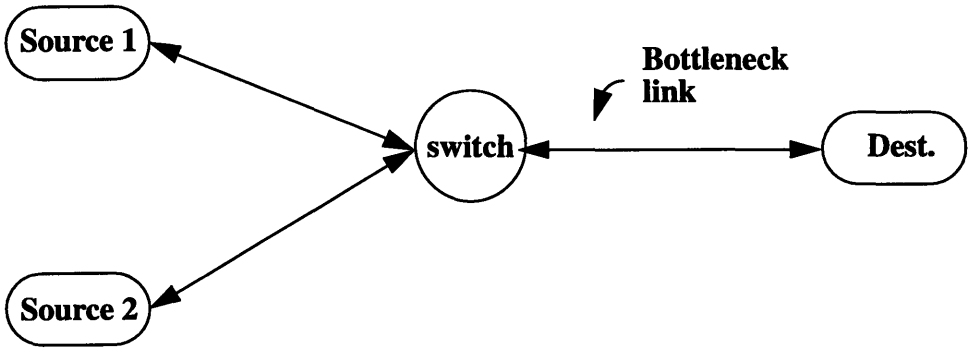


Figure 2.2: Network Topology for Simulation

sources oscillate in unison between $0.5x10^5$ and $3.0x10^5$ cells/sec. The threshold level for the queue is set at 500 cells, and the queue also oscillates between 300 and 700 cells. The amplitudes of the oscillations can be changed somewhat by tuning the parameters of the algorithm, but due to the nonlinearities of this algorithm and the variable time scale, a detailed analysis of the effect of the parameters is difficult.

The CI values in the RM cells received by one of the sources are also shown in Figure 2.1. Examining these reveals an interesting aspect this algorithm. At the top of the source's rate cycle, the source is receiving back RM cells with CI=0 (no congestion), but its rate levels off. At this point the rate is high and new outgoing RM cells are being generates much faster than RM cells are being returned. The result is that more slowdowns occur due to sending RM cells than are compensated for by receiving RM cells. This effect is in fact stabilizing, but makes accurate analysis difficult.

If the rate changes are assumed to occur at fixed intervals, then some analysis is possible. In this case the rate, *R[n]*, is given by:

$$R_i[n] = (1-\alpha)^{C_1} R_i[n-1] + C_0\beta \qquad (2.1)$$

where

C_0 is the number of RM cells with CI = 0 received in a time interval, Δ,

C_1 is the number of RM cells with CI = 1 received in a time interval, Δ,

α is the multiplicative decrease factor and β is the additive increase factor.

An analysis of an algorithm similar to this is performed in [YiH94]. Even with the timing out problem removed, the system still exhibits steady-state oscillations similar to those shown in Figure 2.1. A typical cycle in the oscillation is described in [YiH94] where it is made apparent that the cycles are fundamentally caused by the threshold non-linearity used to generate the binary feedback.

This non-linearity creates a control system which exhibits large limit cycles. When the queue size crosses the threshold level, the input rates will be larger than the output bandwidth and growing. Once the switch enters the congested state, the congestion indicators will cause the source rates to begin to decrease, but the queue size will not decrease

until the input rates fall below the output bandwidth. The cycle will then be reversed. The two sources are synchronized in their oscillation since the nonlinear mapping of the single queue is driving them simultaneously in the same direction. The oscillations will become even larger when delay is added to the feedback loop.

In a more complex network, such as that shown in Figure 2.3, algorithms like PRCA can result in unfairness to those sources-destination pairs, like source/destination 1, which travel through more switches than other source-destination pairs. Data from these sources is more likely to experience congestion than data which travels through fewer links. Thus, the RM cell's from VC's going through more switches get marked more often than those going through fewer, and these sources do not get their fair share of the bandwidth. This is known as the beat-down effect. Simulation showing this effect can be found in [Bar94].

The beat-down problem is due to the fact that in PRCA, the switches cannot differentiate between the different VC's when marking cells to indicate congestion. A proposed solution to the beat-down problem is to have the switch employ 'intelligent marking'. This means that a congested switch should only mark cells from those VC's that are using more than their fair share of the bandwidth. The switch must use the rate information in the RM cells to decide which cells get marked. This also requires that the switch somehow determines what the fair-share for a VC is. However, if the switch can calculate the

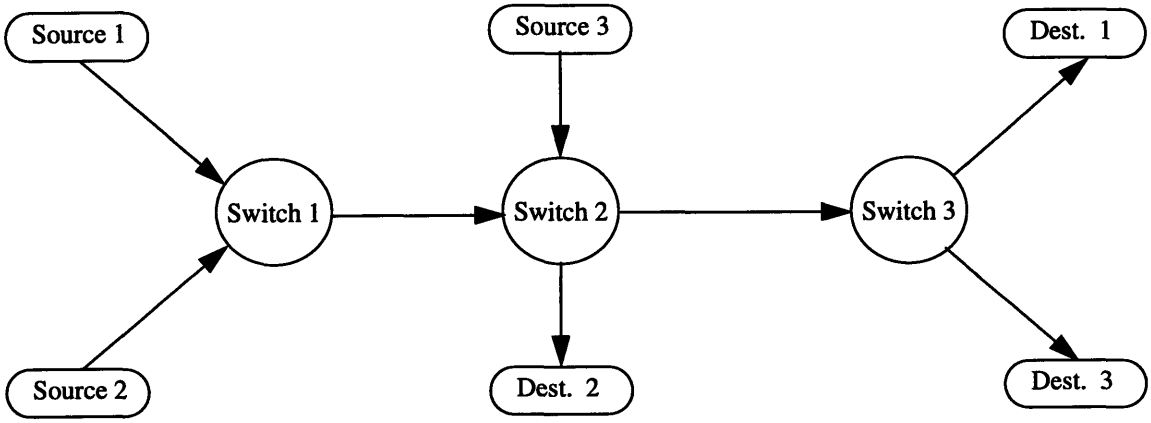


Figure 2.3: Network to Demonstrate 'Beat-down' Effect.

fair rate for a source, then it may make sense for the switch to simply communicate this rate to the source. This is the goal of the explicit rate algorithms.

2.3 Explicit Rate Algorithms

Binary algorithms are limited in that they can only tell a source to increase or decrease its rate. This limits the speed at which a binary algorithm can achieve a fair allocation of rates and is a main cause of the oscillations in PRCA. Explicit rate algorithms are intended to allow more information to reach the source, in particular they allow the switch to tell the source explicitly at what rate to transmit. This should allow the algorithm to react more quickly and have less oscillations than the binary algorithms. The issue here, as with intelligent marking, is how to calculate the fair rate to send to the source.

An early rate-base algorithm was developed by Charny [Cha94] and shown to converge to a max-min fair rate allocation. This scheme calculated the fair rate using the formula:

$$FairRate = \frac{B - B'}{f - f'} \tag{2.2}$$

where:

B is the link bandwidth, B' is the link bandwidth used by all VC's whose rates are less than the fair rate, f is the number of active VC's, and f' is the number of active VC's whose rates are less than the fair rate.

If a source's rate is greater than the fair rate, then the switch sends back the fair rate and the source adjusts its rate to this value. This algorithm requires that the switch store the rate of all active VC's. The computation of the fair rate also requires order n operations, where n is the number of active VC's per link. These considerations complicate the switch implementation, and a simpler algorithm is desired for ATM networks.

The enhanced proportional rate control algorithm (EPRCA) [Rob94] and a related algorithm proposed by Siu and Tzeng [Sit95] both try to emulate Charny's algorithm with fewer calculations and no per-VC accounting. The fair rate proposed in Charny's algorithm is the average rate of all the VC's which are not constrained elsewhere, i.e. all bottleneck VC's. These other algorithms attempt to estimate this rate by an exponential

average of the rates given in the RM cells entering the switch. This average is called the Mean Allowed Cell Rate (MACR). The Siu algorithm averages all the rates when the switch is not congested, but averages only those rates less than the current MACR when the switch is congested. EPRCA averages those rates greater than a fraction of the MACR (7/8 is suggested) when uncongested, and those rates less than the MACR when congested. In both algorithms, when a switch is congested it informs those VC's whose rates are greater than the MACR to reduce their rates to the MACR (EPRCA actually uses a value slightly less than the MACR to try to keep rates just under this point).

If a fair allocation of rates were achieved, then the average of all the rates entering the switch would be less than the average of all the bottleneck VC's. If the switch is congested, then this average must be too high, and so averaging only those rates less than the MACR pulls that average down. If the switch is not congested, the rates are allowed to increase. As the rates increase, MACR will also increase. In this way, the estimate of the fair rate oscillates around its true value. EPRCA only averages those VC's whose rates are greater than a fraction of the MACR when uncongested. This is done with the assumption that those VC's whose rates are much less than the MACR are probably constrained elsewhere. These algorithms also contain a condition where if the queue is "very congested" then the rates are set to a fraction of the fair rate; this drops the rates rapidly if major congestion occurs. Simulation results of these schemes in [SiT95] and [Bar94] show that they solve the beat-down problem of PRCA.

Several additional explicit rate schemes have been proposed to improve on the performance of EPRCA. An algorithm proposed by Jain [JaS95], claims to reach its target rate 10 to 20 times faster than EPRCA, but requires per-VC accounting. Another algorithm proposed by Barnhart [Bar95], does not require per-VC accounting and does not oscillate in steady state. Both of these algorithms set the target bandwidth slightly less than the available bandwidth. This helps keep the queue sizes near zero, which helps speed up the response. Both of these algorithms contain significant non-linearities, making analysis of them difficult.

The Jain algorithm measures the time, T, until the Nth cell arrives at the output link of a switch. The input rate is then calculated as N/T. The target cell rate for that link is set

slightly below the available bandwidth and used to calculate an *overload factor* as follows:

$$Overload\ Factor = \frac{Input\ Rates}{Target\ Rate} \tag{2.3}$$

Where *Input Rates* represents the sum of the input rates destined for a specific output port. The overload factor is used to calculate an *explicit rate based on load* for each VC. This is found by dividing the VC's current rate, read from an RM cell, by the overload factor. Thus, if the input rate is larger than the target rate, the overload factor will be greater than one, and dividing by the overload factor reduces the rates; the opposite effect occurs if the input rate is low. The algorithm also requires that the sources compute a *fair share* for the VC's; this is calculated as:

$$Fair\ Share = \frac{Target\ Rate}{N} \tag{2.4}$$

where N is the number of active VC's. (The switch must also keep track of this.) The explicit rate sent to each VC is the maximum of the *explicit rate based on load* and the *fair share*. This quantity must be stored for each VC and is sent back in all RM cells seen for a particular VC during the next calculation interval. The fair share is included in the algorithm to insure that VC's that start low are able to increase their rates quickly.

Barnhart's algorithm also specifies a target rate that is slightly less than the available bandwidth and an input rate that is calculated as in Jain's algorithm. These are used to calculate an explicit rate for the queue (*ERQ*) by the following formula:

$$ERQ = ERQ\left(1 + \left(1 - \frac{Input\ Rate}{Target\ Rate}\right)Gain\right) \tag{2.5}$$

The gain term is a constant less than one (suggested values of 0.1-0.5). If the input rate is larger than the target rate, then this will cause *ERQ* to decrease, and if the input rate is less than the target rate, then *ERQ* will increase. The switch is also required to keep an exponential average of the rates coming in (MACR) as in EPRCA. The switch uses *ERQ* as the explicit rate that it feeds back as long as this value is between .75*MACR and 1.25*MACR. If *ERQ* is outside of one of these bounds, then that bound is used instead. These bounds act in a similar role to the fair rate in Jain's algorithm.

Both of these algorithms represent a shift in the emphasis for rate based schemes. Previous schemes responded primarily based on the instantaneous queue length. When a certain threshold was crossed to indicate congestion, the algorithms responded to alleviate this problem. These later algorithms base their feedback on the rate that data is entering the queue, and try to keep this lower than available bandwidth. This helps to keep the queue length near zero and thus attempts to prevent congestion from occurring.

2.4 Congestion Control Framework

The framework being considered by the ATM forum would allow for switches to use either a binary or an explicit rate algorithm, or a combination of the two. The following is a summary of the main details of the proposed standard as given in [Atm96]. These standards are still being developed at this time, and this is meant to simply point out some of the main features being considered.

When a connection is set-up, the source must specify a minimum cell rate (MCR) and a peak cell rate (PCR) for the connection; an initial cell rate (ICR) at which the source begins transmitting is also specified. The source is required to transmit an RM cell before transmitting any data and transmit one RM cell for every *Nrm* data cells after that. Before sending an RM cell, if *Xrm* forward RM cells have been sent since the last backward RM cell was received, then the source must lower its rate to no lower than the MCR. When the source receives a returning RM cell with its CI bit set the source initially considers reducing its rate by a multiplicative decrease factor. If a returning RM cell is received without the CI bit set then the source initially considers additively increasing its rate. After computing the new possible rate, the source must check to make sure this rate is less than both the explicit rate (ER) in the last RM cell and the PCR. If not, then the rate is set to the lesser of these two values. The new rate must also be greater than the MCR, or it is set to this value. In this way the source algorithm can be thought of as essentially the same as the PRCA source behavior, except the ER provides an adjustable ceiling for the allowed rates. If the size of the additive increase is large, and the CI bit is never set, then this algorithm will usually set the source rate to the ER.

A switch can mark the EFCI bit in data cells, mark the CI bit in forward or reverse RM cells, lower the ER field in forward or reverse RM cells, or perform a combination of these. How a switch determines which cells to mark, or what to put in the ER field is not specified. It is also allowed for different switches in a network to be employing different feedback techniques. The destination is required to turn around any RM cells it receives. It must set the CI bit if it has received any data cells with EFCI set, and it may lower the ER field or set the CI bit if it is congested.

Chapter 3

Binary Algorithm

3.1 Introduction

The most serious restriction in the binary algorithms discussed in the previous chapter is that the single bit of feedback only indicates that congestion is present or not. In this chapter we agree to only use a single bit of feedback in each RM cell, but we use a sequence of RM cells to convey more than a single bit of information. It should be noted that this requires a source behavior that differs from the ATM Forum's guidelines. There are several ways this can be accomplished. One approach would be for the switches to compute a variable to feedback, and then to send this variable as an n-bit number, where each bit would be sent back in one of n RM cells. Upon receiving a block of n RM cells, the sources would act upon this information. Some problems with this approach are that it requires some means for the sources to stay synchronized to each block of n bits, and a way to deal with a lost RM cell. This would also require the switches to use some form of per-VC accounting to keep track of which bit needs to be sent on each VC.

A second approach avoids some of these difficulties while still conveying a multivalued variable to the sources. This is accomplished through the use of probabilistic feedback. Probabilistic feedback allows the switch to communicate a parameter, p, which is a number between 0 and 1 to the source. The switch can use a pseudo-random number generator to send back CI = 1 with probability p, and CI = 0 with probability 1-p in all RM cells that pass through it in an interval Δ. In the same interval, the source can recreate a noisy estimate of p by averaging the CI's received in that interval.

This chapter will examine the use of probabilistic feedback in implementing a binary congestion control algorithm. In the next section a linear model for a probabilistic feed-

back congestion control algorithm is developed. In Section 3.3 this algorithm is analyzed and some performance trade-offs are outlined. Simulation results of this algorithm are also presented. In the last section some extensions of this approach are outlined.

3.2 Model

The following describes an algorithm for rate based congestion control using probabilistic feedback. This algorithm is designed so that it can be analyzed as a discrete-time linear system. Since probabilistic feedback requires that the source estimate the feed back parameter p over an interval of size Δ seconds, this is a reasonable choice for the sampling interval of the discrete-time system. Hence in the following all variables will represent discrete-time samples spaced Δ apart.

This algorithm is developed for the two source/one switch network topology shown in Figure 2.2. This network is redrawn as a block diagram in Figure 3.1. The two sources generate cells at rates R_1 and R_2. Thus, the total rate that cells arrive at the switch is given by R, where

$$R = R_1 + R_2 \tag{3.1}$$

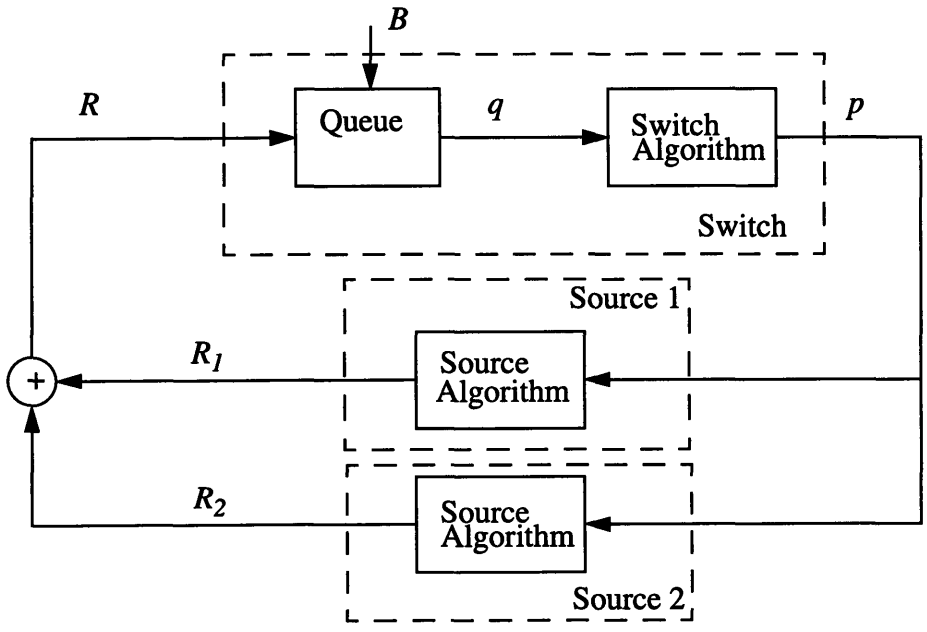


Figure 3.1: Block diagram of 2 source/1 switch network.

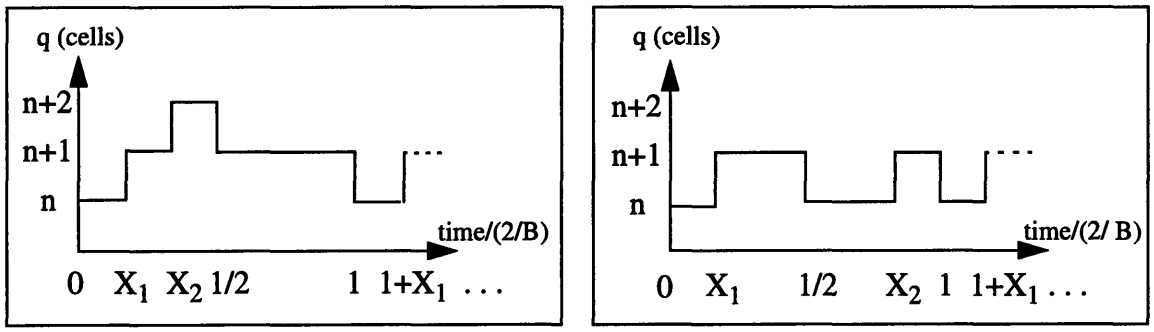
The output port of the switch that is the destination of these cells has an available bandwidth of B. If a cell arrives while another is in being transmitted, then it is queued. Based on the queue size, q, the feedback parameter, p, is calculated. This parameter is then conveyed to the sources using probabilistic feedback and the sources adjust their rates accordingly. In the following the source and switch algorithms are developed. These algorithms are represented as linear relations between the appropriate input and output variables shown in Figure 3.1. A model for the queue will also be developed. Once this is completed then the closed-loop system in Figure 3.1 can be analyzed using linear control theory.

We begin with the model for the queue. The switch queues the cells until they can be delivered. The length, q, of the queue at successive time instants separated by Δ seconds is modeled by

$$q[n] = max\{q[n-1] + (R[n-1] - B)\Delta + n_q, 0\} \tag{3.2}$$

As long as the queue length stays above zero, this represents a linear time invariant relationship between q and R. Equation 3.2 models the flow of cells through the queue as a continuously varying quantity. In reality, cells arrive and leave the actual queue at discrete times and the actual queue always contains an integer number of cells. The difference between the model of the queue in Eq. 3.2 and the real queue is accounted for by the noise source n_q.

This noise source is characterized by considering the actual queue size in the two source/one switch network when the input rates are both set at 1/2 the output bandwidth, B. In this case the model in Eq. 3.2, without n_q, specifies that the queue size should stay constant. Figure 3.2 shows the queue size of the actual queue for this situation. In this figure X_i represents the time a cell arrives from source i. The x-axis is normalized by $2/B$ so that cells depart from the queue at 1/2, 1, . . . This figure shows that the actual queue size varies depending on when X_1 and X_2 occur and when the queue is sampled. The noise, n_q, is modeled as zero mean white noise whose variance is equal to the variance of the queue in this situation. To calculate this variance it is assumed that X_1 and X_2 are independent of each other and equally likely to occur anywhere between 0 and 1, *i.e.* independent uniform



case 1 case 2

Figure 3.2: The actual queue size, q, when the two input rates are both B/2 for two cases.

random variables. To calculate the variance of the queue, four equally likely cases are considered: (1) $X_1 < 1/2, X_2 < 1/2$, (2) $X_1 < 1/2, X_2 > 1/2$, (3) $X_1 > 1/2, X_2 < 1/2$, and (4) $X_1 > 1/2, X_2 > 1/2$. Cases 1 and 2 are shown in Figure 3.2. Cases 3 and 4 can be thought of as cases 2 and 1 respectively with the origin shifted to the left by 1/2. Due to this symmetry, cases 1 and 4 and cases 2 and 3 are considered together.

CASE 1 & 4: First we consider case 1. Conditional on being in case 1, then X_1 and X_2 are independent uniform random variables on (0,1/2). We want to find the pmf of the queue in this case. This pmf can be written as

$$p(Q=q|I) = p(Q=q|X_1<X_2,I)p(X_1<X_2|I) + p(Q=q|X_2<X_1,I)p(X_2<X_1|I) \tag{3.3}$$

where I represents the conditioning of being in case 1. Since the X_i are independent and indentically distributed, then we know that

$$p(X_1<X_2|I) = p(X_2<X_1|I) = 1/2 \tag{3.4}$$

and

$$p(Q=q|\, X_1<X_2,I) = p(Q=q|X_2<X_1,I) \tag{3.5}$$

thus Eq. 3.3 reduces to

$$p(Q=q|I) = p(Q=q|X_1<X_2,I) \tag{3.6}$$

From Figure 3.2, given X_1 and X_2 and that $X_1 < X_2$, the pmf for Q is given by:

$$p(Q=q|X_1=x_1, X_2=x_2, X_1<X_2,I) = \begin{cases} x_1, & q = n \\ x_2 - x_1 + 1/2, & q = n+1 \\ 1/2 - x_2, & q = n+2 \end{cases} \tag{3.7}$$

To get from Eq. 3.7 to Eq. 3.6, the pdf's $f(x_1|X_2=x_2, X_1<X_2,I)$ and $f(x_2|X_1<X_2,I)$ are needed. The first of these is given by

$$f(x_1|\, X_2 = x_2\,, X_1<X_2,I) = 1/x_2\,,\ 0<x_1< x_2 \tag{3.8}$$

Bayes rule is used to find $f(x_2|X_1<X_2,I)$ as follows

$$f(x_2|X_1 < X_2, I) = \frac{p\,(X_1 < X_2|X_2 = x_2, I) f(x_2|I)}{p\,(X_1 < X_2|I)}$$

$$= \frac{(2x_2)\ (2)}{1/2}\ \text{, if } 0 < x_2 < 1/2 \tag{3.9}$$

multiplying Eq. 3.7 by Eq. 3.8 and Eq. 3.9 and then integrating over X_1 and X_2 yields the desired pmf

$$p(Q=q|I) = p(Q=q|\, X_1<X_2,I) = (1/6)\delta(q-n) + (2/3)\delta(q-n-1) + (1/6)\delta(q-n-2) \tag{3.10}$$

The pmf for the queue in case 4 is found in an identical manner except the queue will now take on the values n-1, n, and n+1. The pmf for this case is:

$$\mathrm{p}(Q=q|IV) = (1/6)\delta(q-n+1) + (2/3)\delta(q-n) + (1/6)\delta(q-n-1) \tag{3.11}$$

CASE 2 & 3: The pmf of the queue in case 2 and case 3 is identically the same. We initially consider case 2. For this case X_1 is uniform on (0, 1/2) and X_2 is uniform on (1/2,1) and again both are independent. The ordering of X_1 and X_2 is already fixed in this case. The pmf of Q, given X_1 and X_2, can be written, by looking at Figure 3.2, as:

$$p(Q=q|X_1=x_1, X_2=x_2,II) = \begin{array}{ll} x_1 + x_2 - 1/2, & q = n \\ 3/2 - x_1 - x_2, & q = n+1 \end{array} \tag{3.12}$$

Multiplying this by the pdf's for X_1 and X_2 and integrating over these variables results in the pmf for Q given this case:

$$p(Q=q|II) = p(Q=q|III)=(1/2)\delta(q-n) + (1/2)\delta(q-n-1) \tag{3.13}$$

The unconditional pmf for the queue can then be found by summing the pmf's for each case (given in Eq. 3.10, 3.11 and 3.13) weighted by the probability of that case, which is 0.25 for each case. This results in:

$$p(Q=q) = (1/24)\delta(q-n+1) + (11/24)\delta(q-n) + (11/24)\delta(q-n-1) + (1/24)\delta(q-n-2) \tag{3.14}$$

From this pmf, the total variance of the queue is 0.417.

Next we consider the source algorithm. The source must do two things. First it must estimate p from the received RM cells, then it must adjust its rate based on this estimate. To estimate p, in the interval between time $(n\text{-}1)\Delta$ and $n\Delta$ each source accumulates as C_1 the number of RM cells received with CI = 1 and as C_0 the number of RM cells received with CI = 0. We denote source i's estimate of p as p_i. This estimate is calculated as

$$p_i[n-1] = \frac{C_1}{C_0 + C_1} \tag{3.15}$$

The difference between the variable p and the estimate p_i is accounted for by additive noise, n_i.

$$p_i[n] = p[n] + n_i[n] \tag{3.16}$$

In a representative time interval Δ, a source receives D congestion indicators from the switch. These congestion indicators can be modeled as D independent indentically distributed binomial random variables taking on the value 1 with probability p and 0 with probability 1-p. The source in Eq. (3.14) estimates p by computing the sample mean of these N random variable. The noise, n_i, represents the error made in this estimation. The errors in each interval are assumed independent so n_i can be represented as white noise with a variance given by:

$$var(n_i) = \frac{1}{D} p(1-p) \tag{3.17}$$

After the source estimates p, it must use this estimate to adjust its rate, R_i. The source will compute its new rate according to:

$$\begin{aligned} R_i[n] &= \gamma R_i[n-1] + (1 - p_i[n-1])\beta - p_i[n-1]\alpha \\ &= \gamma R_i[n-1] - (\alpha + \beta) p_i[n-1] + \beta \end{aligned} \tag{3.18}$$

where α, β and γ are positive constants. This equation has an effect similar to Eq. (2.1) except it has been made linear in the parameter p. The parameter γ is slightly less than one causing an exponential decrease in the rate that can be overcome by an additive increase

when there is little congestion, *i.e.* when p_i is near zero

Finally the switch algorithm is considered. The switch must use *q[n]* to compute the feedback parameter *p*. We denote as *H(z)* the transfer function of the LTI operator used by the switch to calculate *p*. The transfer function, *H(z)*, should be chosen so that the resulting closed loop system is stable and the performance is in some way satisfactory. This transfer function must also be chosen to attempt to keep the parameter *p* between 0 and 1, since this parameter is used as a probability. Using Eqs. (3.1), (3.2) and (3.14), the loop gain, *G(z)*, of the closed system is given by:

$$G(z) = \frac{2(\alpha+\beta)\Delta z^{-2}}{\left(1-\gamma z^{-1}\right)\left(1-z^{-1}\right)} H(z) \tag{3.19}$$

Assuming that *H(z)* is simply a constant, and setting $2(\alpha+\beta)\Delta H(z)$ to unity then the Bode plot for the control loop is shown in Figure 3.3. The parameter γ was set to .99 for this plot, which is shown in the next section to be a reasonable value.

In order to have an adequate phase margin at crossover, a small loop gain would be required. This would result in a low crossover frequency and a correspondingly slow sys-

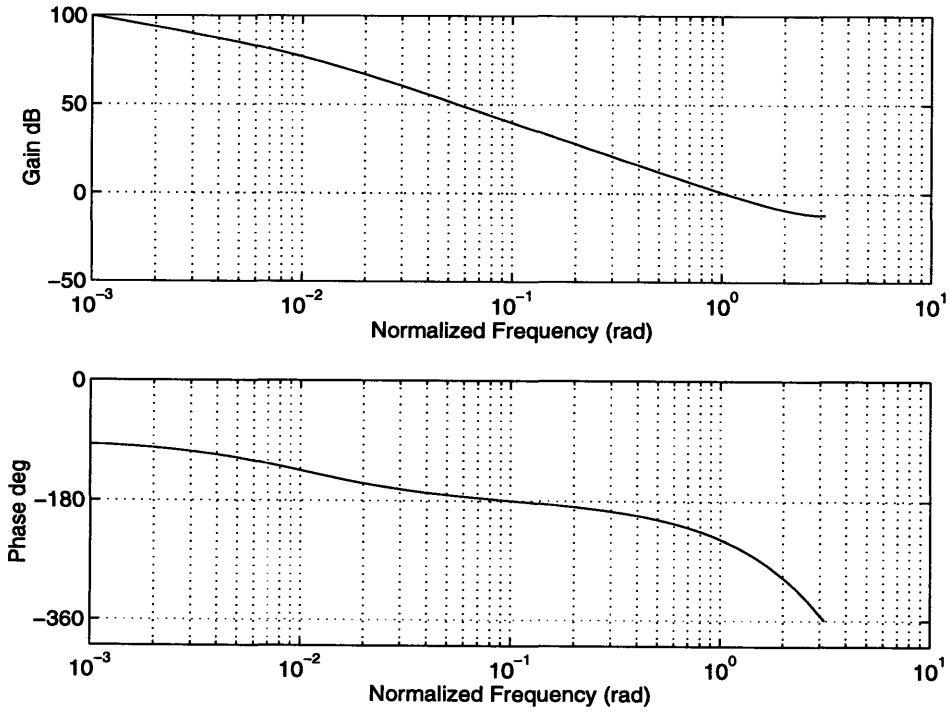


Figure 3.3: Bode plot of loop gain with constant H(z).

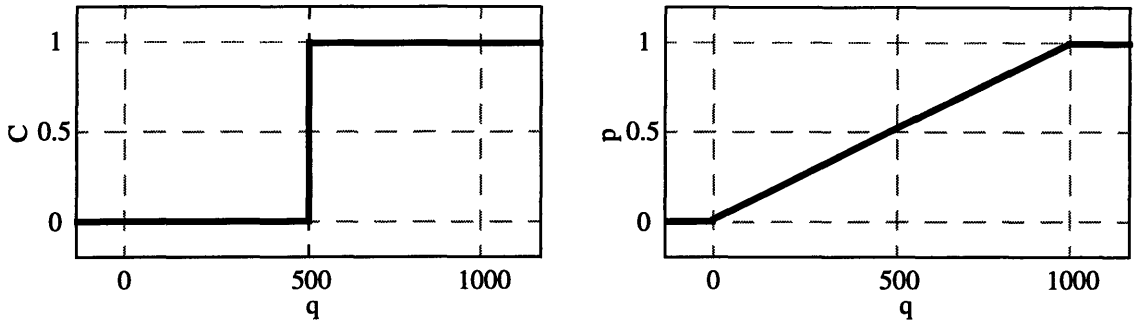


Figure 3.4: .(a) Threshold non-linearity. (b) Softer non-linearity for probabilistic feedback.

tem. To over come this, $H(z)$ is chosen to be a lead network; the additional phase lead will allow a larger crossover frequency to be chosen. The chosen lead network is

$$H(z) = (a+b)\left(1 - \frac{a}{a+b}z^{-1}\right) \tag{3.20}$$

This specifies the following relationship between p and q:

$$p[n] = a(q[n] - q[n-1]) + bq[n] = (a+b)q[n] - aq[n-1] \tag{3.21}$$

Thus, the feedback is proportional to the rate the queue length is changing and to the instantaneous queue length. The choice of a and b determines how much each of these terms is weighted.

Since p is interpreted as a probability it is allowed to saturate at 0 and 1. If the parameters a and b are chosen small enough, then p will not saturate and the algorithm will function as a linear control system. The feedback used in the ATM Forum's binary schemes discussed in Section 2.2 is based on a threshold non-linearity as shown in Figure 3.4a. The proposed algorithm has replaced this with a softer non-linearity as shown in Figure 3.4b. This provides a region of linear control. If the system saturates then it operates essentially the same as the ATM Forum's algorithms and produces stable oscillations as discussed in Section 2.2

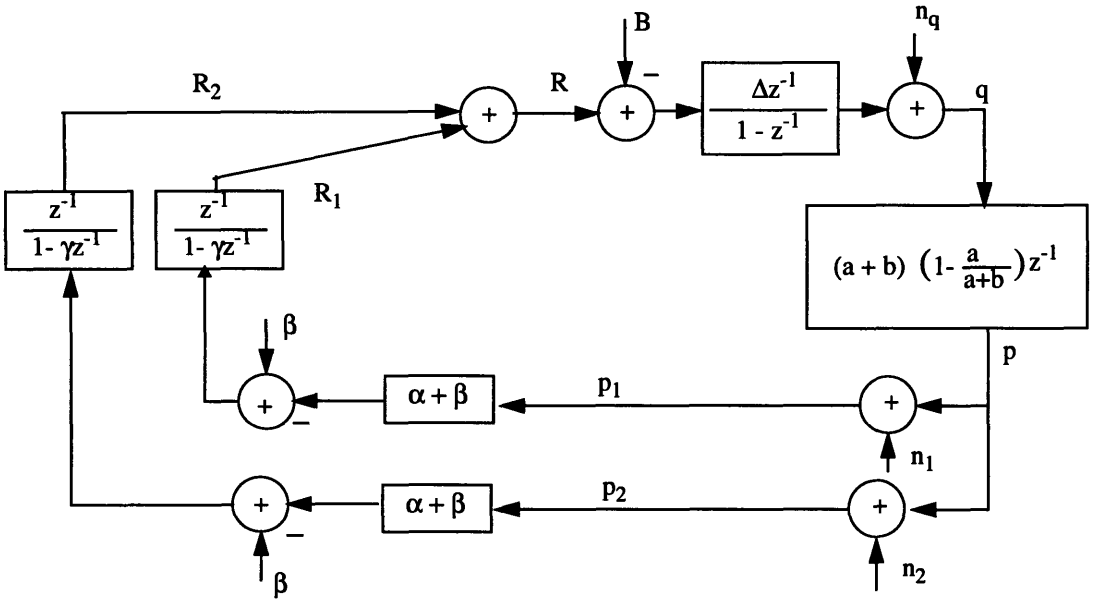


Figure 3.5: Block diagram of closed loop system.

3.3 Analysis and Simulations

Using the results of the previous section, a complete model for the closed-loop system is specified. A block diagram of this model is shown in Figure 3.5. Now that the model is given, linear control theory can be used to analyze the control system. This provides guidance in choosing the parameters so that design trade-offs are clear through analysis rather than the trial and error of simulation. In this section, such an analysis of the closed-loop system is presented. Simulation results are given as a verification of this analysis.

First we examine the steady-state values of the variables. We consider the case where the parameters are chosen so that the closed-loop system is stable and operates in the linear region, *i.e.* that p remains between zero and one. Then, except for the noise sources driving the system, the systems reaches a constant steady state. With $n_i = 0$ and $n_q = 0$, the steady-state values are:

From Eq's 3.1 and 3.2 $$\bar{R} = B \quad \text{and} \quad R_i = B/2 \tag{3.22}$$

From Eq. 3.18 $$\bar{p} = \frac{\beta - (1-\gamma)\,(B/2)}{\alpha + \beta} \tag{3.23}$$

From Eq. 21 $$\bar{q} = \frac{\bar{p}}{b} \tag{3.24}$$

Based on the expected steady-state rates of the sources, we choose the sampling time Δ. The sampling time Δ must be chosen large enough so that each source receives enough RM cells to make a good estimate of p. From Eq. 3.17, as Δ increases, the noise variance of n_1 and n_2 decrease. The switch calculates a new feedback value every Δ seconds, and the sources take another Δ seconds to estimate this new value. Thus, if an abrupt input transient occurs it can take 2Δ seconds plus the propagation delay before the source can react. Increasing Δ increases this delay. The sampling time must be chosen to achieve a desirable compromise between these considerations. In the network used in the following simulations, the output port on the switch has a bandwidth of 3.54×10^5 cells/sec. (155Mbps) and both sources send one RM cell for every 31 data cells. An initial choice for the sampling time is $\Delta = 9 \times 10^{-4}$ seconds so that at their steady-state rates, each source receives an average of 5 RM cells per sampling interval.

The steady-state value for p, given in Eq. 3.23, must be kept between 0 and 1 to stay in the linear region. This requires that:

$$0 < \frac{\beta - (1-\gamma)\ (B/2)}{\alpha + \beta} < 1 \tag{3.25}$$

$$(1-\gamma)\ (B/2) < \beta < \alpha + \beta + (1-\gamma)\ (B/2)$$

Since γ is less than 1, the right inequality will always be satisfied and so we have that β and γ must satisfy:

$$(1-\gamma)\ (B/2) < \beta \tag{3.26}$$

From Eq. 3.18, it can be seen that choosing a small value for γ increases the speed of the multiplicative decrease in the rates, and thus removes the effects of previous rates faster during a transient phase. This has the advantage of pulling the source rates together and decreasing the required queue sizes. On the other hand, a smaller γ is harder for the additive increase to overcome and slows down the speed at which rates can initially increase. An initial value of $\gamma = .99$ is used. We also initially choose $\alpha = 0$, making rate decreases solely due to γ.

From Eq. 3.24, the parameter b relates the steady-state queue size to the steady-state probability, p. Since the steady-state probability must be between 0 and 1, choosing b = .01 makes the steady-state queue length less than 100 cells. Now that b is chosen, the choice of the parameter a will determine the zero location for the lead network. The zero location, z, is given by:

$$z = \frac{a}{a+b} \tag{3.27}$$

Placing the zero as close to one as possible will add the most phase lead at the crossover frequency. From Eq. 3.27, the zero location is an increasing function of a, so this makes it desirable to choose a as large as possible. But, from Eq. 3.21 the choice of a determines how much a change in the queue length is weighted in the calculation of p. Even in steady-state, the noise sources can cause the queue length to fluctuate by at least one cell between sampling times. It is desirable to make sure that a is chosen small enough so that this one cell fluctuation does not force p out of the linear range. Assuming we are in steady-state this condition can be written as:

$$a(1) + b\bar{q} < 1 \quad \text{and} \quad -a(1) + b\bar{q} > 0 \tag{3.28}$$

Using that $\bar{p} = b\bar{q}$ we can re-write this as:

$$a < min(1-\bar{p}, \bar{p}) \tag{3.29}$$

Choosing a smaller than the bound in Eq. 3.29 insures that the algorithm stays in the linear region for even larger fluctuations in the queue. As a compromise between the desire for a large a for maximum phase lead, and a small a to satisfy Eq. 3.29, a value of a = 0.0685 was chosen for the initial simulation.

For the closed loop system, the control loop gain, $G(z)$, is given by:

$$G(z) = \frac{2(\alpha+\beta)\Delta(a+b)z^{-2}\left(1-\frac{a}{a+b}z^{-1}\right)}{\left(1-\gamma z^{-1}\right)\left(1-z^{-1}\right)} \tag{3.30}$$

The Bode plot of a typical loop gain is shown in Figure 3.6. Changing β will shift the magnitude plot up or down and leave the phase plot unchanged. Thus, picking a large

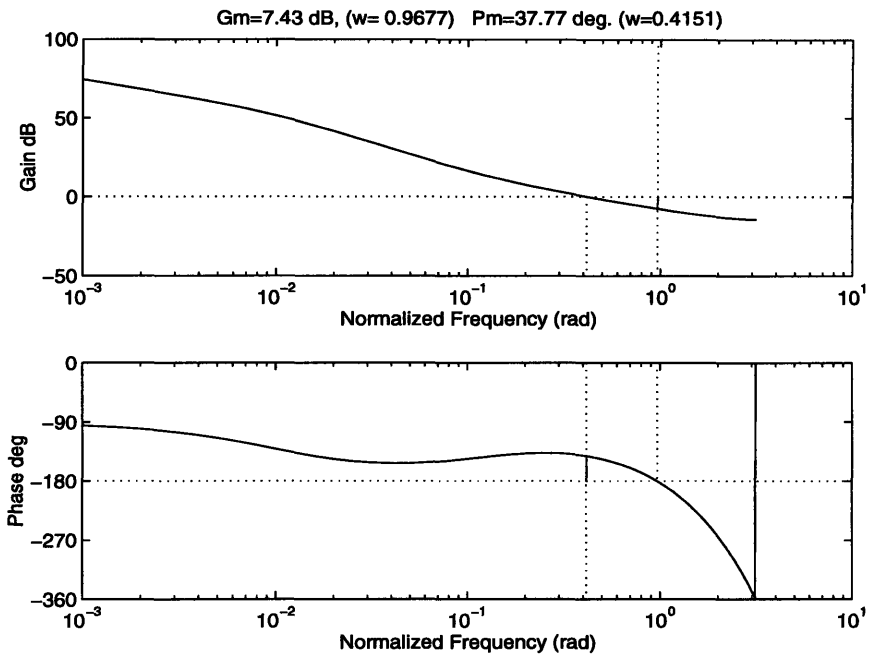


Figure 3.6: Bode plot of loop gain for γ=0.99, a=0.0685, b=0.01, α=0, and $\beta=2.95*10^3$

value for β results in a system with a larger bandwidth but a smaller phase margin. The final parameter, β, was chosen as a compromise between setting the bandwidth of the system and setting the steady-state probability, p, via Eq. 3.23. The resulting choice of $\beta = 2.95\text{x}10^3$ yields steady-state values of $\bar{p} = 0.4$ and $\bar{q} = 40$. The resulting closed-loop system has a bandwidth of about 500 rad/sec and a phase margin of about 38 degrees.

The amount of delay that can be handled in the linear region of this control system can be predicted using control theory. A pure delay adds negative phase lag to the Bode plot of the loop gain while leaving the magnitude plot unaffected. The phase lag associated with a delay of T seconds increases proportionally with frequency, *i.e.* the phase lag is ωT rad. The linearized system remains stable until the phase lag contributed by the delay at the crossover frequency exceeds the phase margin of the control system. In other words the maximum delay for which the linearized system remains stable is given by:

$$max\ delay = \frac{phase\ margin\ (in\ radians)}{crossover\ (in\ rad/s)} \tag{3.31}$$

For the loop gain shown in Figure 3.6, the linearized system will remain stable for delays up to about 1.5 msecs. If the delay were larger than this the linearized system would be unstable; the system would be controlled by the saturation non-linearity and it would

behave much as the system in Section 2.2. To handle larger delays, the loop gain could be lowered or Δ could be increased.

The effect that the noise sources have on the steady-state values can be predicted using linear system theory. Given the transfer function, $H(e^{j\omega})$, between a noise source and a variable of interest, then the variance in the variable is related to the variance of the white noise source by:

$$var(variable) = \left(\int_{-\pi}^{\pi} \left| H\left(e^{j\omega} \right) \right|^2 d\omega \right) var(noise) \tag{3.32}$$

We will call this scale factor the covariance response of the transfer function. There are three noise sources present in the system, one probabilistic noise, n_i, for each data source and the queue noise, n_q. These noise sources are independent and so the total variance seen in a variable is the sum of the variance in the variable due to each noise source. To demonstrate, the variance expected in the queue is calculated for the system we have designed.

The transfer function from each noise source, n_i, to the queue length, q, is given by:

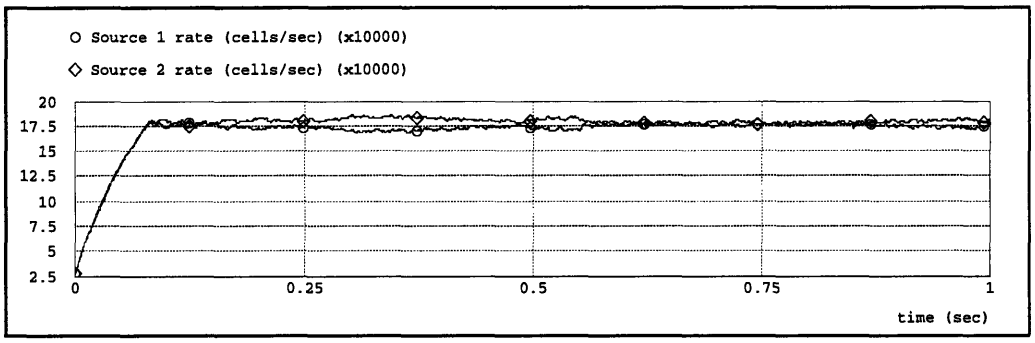
$$\frac{q(z)}{n_i(z)} = \frac{-(\alpha+\beta)\Delta z}{z^3 - (1+\gamma)z^2 + (\gamma + 2(\alpha+\beta)(a+b)\Delta)z - 2(\alpha+\beta)\Delta a} \tag{3.33}$$

For the model parameters above, the covariance response from each probabilistic noise, n_i, to the queue length, q, is 234. Using N=5 and the steady-state value of p = 0.4 in Eq. 3.17 estimates the variance for each n_i as 0.048. Multiplying this by the covariance response yields a variance in the queue of 11.2 cells^2 due to each n_i. Since the n_i's associated with each source are independent, the total variance in the queue due to the probabilistic noise is 22.4 cells^2.

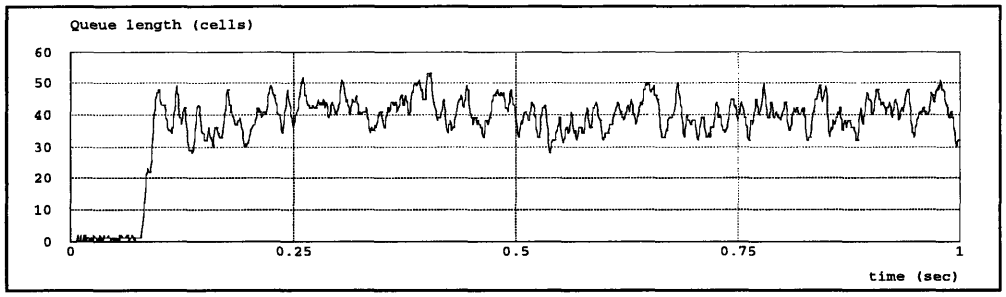
The transfer function from the queue noise, n_q, to the queue is:

$$\frac{q(z)}{n_q(z)} = \frac{z^3 - (1+\gamma)z^2 + \gamma z}{z^3 - (1+\gamma)z^2 + (\gamma + 2(\alpha+\beta)(a+b)\Delta)z - 2(\alpha+\beta)\Delta a} \tag{3.34}$$

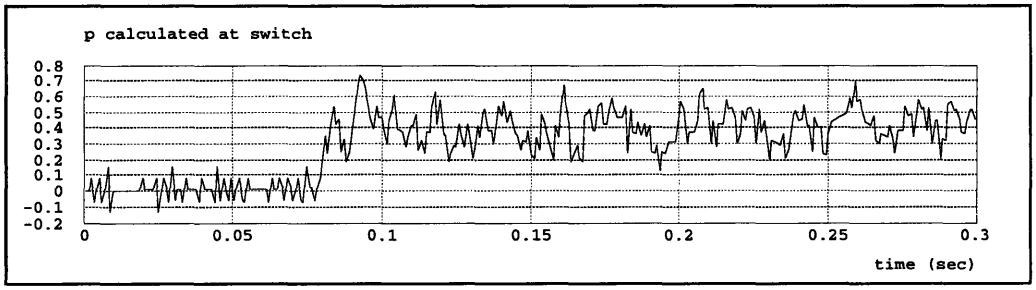
For the chosen parameters the covariance response from n_q to q is 1.6. The variance of n_q



a. Rates of the sources.



b. The Queue length



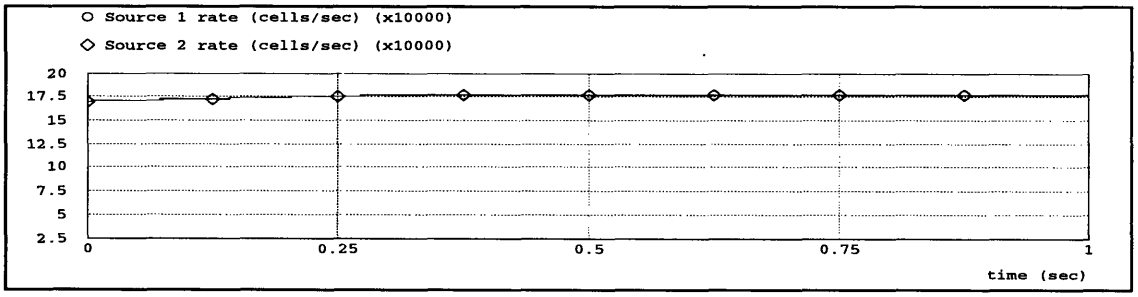
c. Value of p calculated by the switch

Figure 3.7: Simulation results for probabilistic system with γ=0.99, a=0.0685, b=0.01, α=0,and β=2.95x10^3.

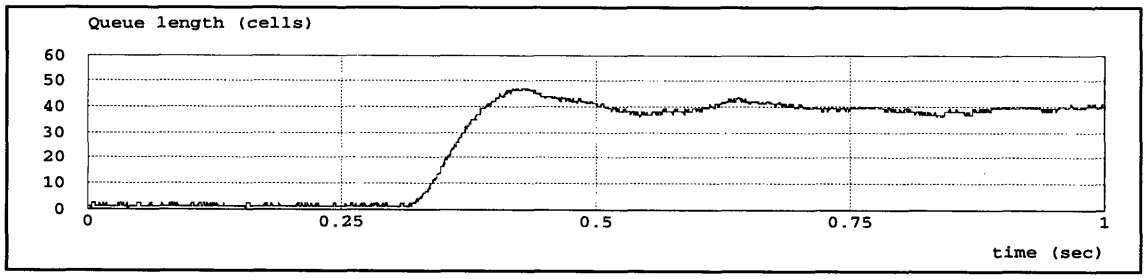
is 0.417, so the queue has a variance of 0.67 cells^2 due to the queue noise. Thus the total variance in the queue is 23 cells^2, corresponding to a standard deviation of 4.8 cells. The variance in the rates or any other variable can be found in the same manner.

As long as the variables remain in the linearized region, the entire response can be determined analytically. Simulations of the actual congestion control system were run to verify performance. The results of a simulation are in given in Figure 3.7. The rates of the two sources are displayed in Figure 3.7a, where it is seen that the rates converge to half the available bandwidth as predicted by Eq. 3.22. In the steady-state, the queue length

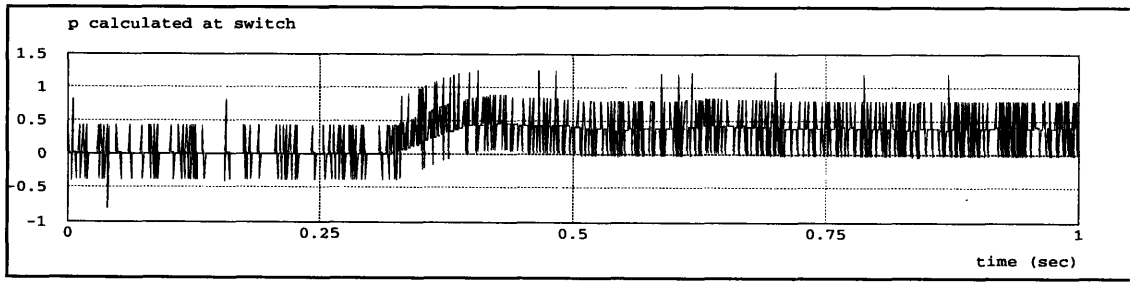
moves slightly around the predicted steady-state value of 40; the parameter p also moves around its predicted value of 0.4. The computed sample variance in the steady-state queue during the simulation run is 22.2 cells^2. This corresponds to a standard deviation of 4.7 cells, very close to the standard deviation of 4.8 cells predicted above. The presence of the noise which enters the system prevents a completely quiescent steady-state. Notice, however, as a general benchmark that the queue is kept to an order of magnitude smaller than the non-linear simulation of Figure 2.1 and that the small remaining deviations from a constant steady-state are due to noise and not to systematic oscillations.



a. Rates of the sources.



b. The queue length.



c. Value of p calculated by the switch

Figure 3.8: Simulation of the probabilistic system with γ=0.9998, a=0.4, b=0.01, α=0, and β=59.

To demonstrate the trade-offs that are achievable using linear control theory the system was redesigned with the goal of reducing the effect of the noise from the communication of p while maintaining the same steady-state values. To achieve this goal we could reduce the bandwidth or increase Δ, both of which also result in a slowing of the transient response. The important point is that because the system can be analyzed we know the precise nature of the trade-offs involved. We choose to reduce the bandwidth. To decrease the bandwidth, the loop gain of Eq. 3.22 is lowered by decreasing β. The parameter a must be increased to protect the phase margin. Finally, in order to keep the steady state p in Eq. 3.23 at $p = 0.4$, the value of γ must be increased making the rates of the two sources converge more sluggishly. We set γ=0.9998, a=0.4, b=0.01, α=0, and β=59.

The bandwidth of the resulting system is now 50 rad/sec with a 60 degree phase margin resulting in robustness to delays in the loop up to 19 msecs. The covariance response from n_q to q, Eq. 3.34, is now 1.04 and the covariance response from each n_i to q, Eq. 3.33, is now 31.9. The noise variances are still the same, so the resulting variance of the queue is now predicted to be 3.49 cells^2, corresponding to a standard deviation of 1.9 cells. The simulation was run starting the rates near their steady-state values to avoid excessive simulation through the longer transient. The simulation results appear in Figure 3.8. The most noticeable difference is the much reduced effect of the noise on the rates and the queue as predicted from the smaller closed-loop bandwidth.

3.4 Extensions

In the previous sections a model for rate based congestion control using probabilistic feedback was developed and analyzed. This model was developed for the simple 2 source/1 switch network of Figure 3.1. We now consider extending this method to larger networks.

We first consider applying the algorithm to a N source/1 switch network as shown in Figure 3.9. We still assume that all of these sources are greedy and are competing for the bandwidth of a single output length. For N sources the loop gain, G(z) becomes:

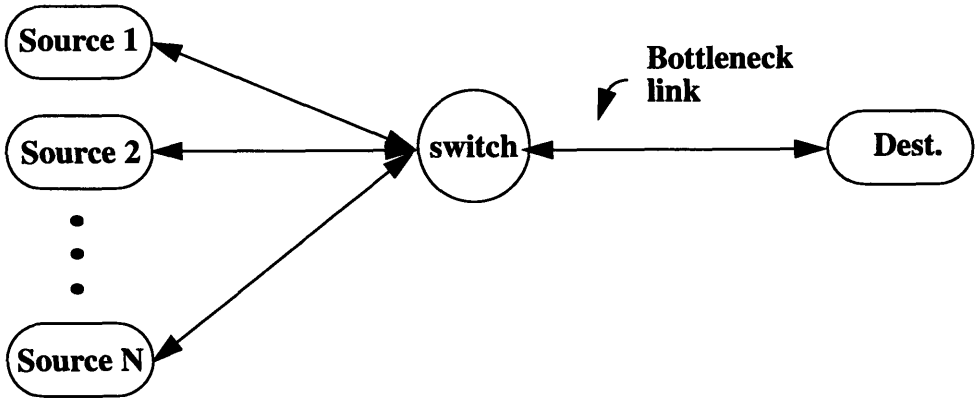


Figure 3.9: N source/1 switch network.

$$G(z) = \frac{N(\alpha+\beta)\Delta(a+b)z^{-2}\left(1-\frac{a}{a+b}z^{-1}\right)}{\left(1-\gamma z^{-1}\right)\left(1-z^{-1}\right)} \tag{3.35}$$

Thus as N increases the loop gain also increases. This results in less robustness to delays and more noise in the steady-state values. If the algorithm is designed to give adequate performance when all N sources are active, then when less than N sources are active, the algorithm will be slow to converge. For large N, this effect can be unacceptable.

A solution to this problem is to estimate the number of active sources and adjust the loop gain appropriately. The Jain algorithm [JaS95] discussed in Section 2.2 also requires that the source estimate the number of active sources. In the Jain algorithm this is accomplished by the switch reserving a bit for every active VC. If a cell from that VC is seen in a particular interval, then the bit for that VC is set, and at the end of the interval the number of set bits is counted. This method would also work here, but an alternate method that would not require per VC accounting at the switch is also possible. In the linear control model, estimating N is equivalent to estimating the loop gain, for which adaptive algorithms exist [AsW95]. The precise method of setting the loop gain is left to further study. In the following we assume that this is accomplished and analyze the effect on the system performance. In particular we look at the effect on the steady-state values and on the noise present in the output.

In order to keep the loop gain constant we modify the switch algorithm of Eq. 3.20 to be:

$$H(z) = \frac{1}{N}(a+b)\left(1-\frac{a}{a+b}z^{-1}\right) \tag{3.36}$$

This results in a loop-gain of

$$G(z) = \frac{(\alpha+\beta)\,\Delta\,(a+b)\,z^{-2}\left(1-\frac{a}{a+b}z^{-1}\right)}{\left(1-\gamma z^{-1}\right)\left(1-z^{-1}\right)} \tag{3.37}$$

which does not depend on N. With this switch algorithm the steady-state values, given in Eq's 3.21 - 3.23 become:

$$\bar{R} = B \text{ and } R_i = B/N \tag{3.38}$$

$$\bar{p} = \frac{\beta-(1-\gamma)\,(B)}{\alpha+\beta} \tag{3.39}$$

$$\bar{q} = N\frac{\bar{p}}{b} \tag{3.40}$$

Thus, each source gets $1/N$ of the available bandwidth and the steady-state p does not change with the number of active sources. The steady-state queue increases linearly with the number of sources. This implies that the amount of buffer space in the switch must be based on the maximum number of active sources.

We now examine the effect of the noise sources in this algorithm. We first consider the probabilistic noise, n_i. With N active sources, the average number of RM cells, D, received in a Δ interval in the steady-state is given by:

$$D = \frac{1}{N}\left(\frac{B\Delta}{32}\right) \tag{3.41}$$

The sampling time Δ should be chosen large enough so that D is larger than one when the maximum number of sources are active. If D is less than one, then in some intervals the sources will not receive back any RM cells and thus time-out and slow down, resulting in behavior that is difficult to analyze. Substituting Eq. 3.41 into Eq. 3.20 yields the variance in each noise source, n_i:

$$var(n_i) = N\left(\frac{32}{B\Delta}\right)p(1-p) \tag{3.42}$$

With the $1/N$ normalization in the loop gain, the covariance response of each n_i to the queue remains unchanged as N varies. Thus, the variance in the queue due to all the n_i's is proportional to N^2. If we consider the algorithm with the parameters used in Figure 3.7, then increasing the number of sources from 2 to 10 increases the variance in the queue due to the n_i's from 22.4 cells^2 to 560 cells^2. For a large number of sources, this could be unacceptable. One possible solution to this problem is instead of keeping the loop gain constant as N changes, to lower the loop gain as N increases. Thus as N gets large, the covariance response would decrease. This would keep the variance from increasing as much, but would result in a slower system. The exact nature of implementing such a system is left to future work.

We next consider the queue noise, n_q. The calculation of the variance of n_q in Section 3.2 was only for the 2 source case. Extending this analysis to the N source case, requires considering more than N separate cases, and a direct analysis as in Section 3.2 quickly becomes intractable. Instead we find an expression for n_q as a function of N which can be evaluated numerically.

As in Section 3.2, we calculate the variance of n_q as equivalent to the variance in the queue size in the steady-state, *i.e.* when all N sources are transmitting at B/N. We normalize the time axis by N/B so that cells depart from the queue at times $1/N$, $2/N$, $3/N$. . ., and so there are N arrivals to the queue in an interval of length one. Each of these arrivals have a uniform probability of occurring at any location on a unit interval. We consider looking at the queue size at a random time, t, during this interval and want to know the variance that we see in the queue size. The time at which we observe the queue is also uniform over the unit interval.

To calculate the variance in the queue size, we consider two new random variables, A and D; where A is the number of cells that arrived to the queue in the current interval before our observation time, t, and D is the number of cells that have departed from the queue before our observation time. Thus, the observed queue size, Q, is given by A - D +

c, where c is a constant that accounts for the steady-state queue size. The variance in Q can then be found by:

$$\begin{aligned} var(Q) &= var(A) + var(D) - 2cov(A, D) \\ &= var(A) + var(D) - 2E(AD) + E(A)E(D) \end{aligned} \tag{3.43}$$

The probability that $A = a$ is the probability that a of the uniform arrivals occur before the observation time. Thus the pmf for A is:

$$P(A = a) = \frac{1}{N+1}, a = 0...N \tag{3.44}$$

From this we find the mean and variance of A to be:

$$E(A) = \frac{N}{2} \text{ and } var(A) = \frac{N^2 + 2N}{12} \tag{3.45}$$

The probability $D = d$ is the probability that the observation time occurs in the interval $(d/N, (d+1)/N)$. Thus the pmf for D is:

$$P(D = d) = \frac{1}{N}, d = 0...N-1 \tag{3.46}$$

and the mean and variance of D are given by:

$$E(D) = \frac{N-1}{2} \text{ and } var(D) = \frac{N^2 - 1}{12} \tag{3.47}$$

Finally we need to calculate E(AD). To calculate this, we first consider the pmf of D given $A=a$. If $A = a$, then the observation time is the $(a+1)^{st}$ of $N + 1$ iid uniform random variables (the N arrivals and the observation time), the pdf for the location on the unit interval of the $(a+1)^{st}$ uniform random variable of N is given by: [Gal96]

$$f(x) = \frac{(N+1)!x^a(1-x)^{N-a}}{(N-a)!a!} \tag{3.48}$$

Given that $A=a$, the probability that $D=i$ is the probability that the $(a+1)^{st}$ uniform random variable is in the interval $(i/N, (i+1)/N)$. This is given by:

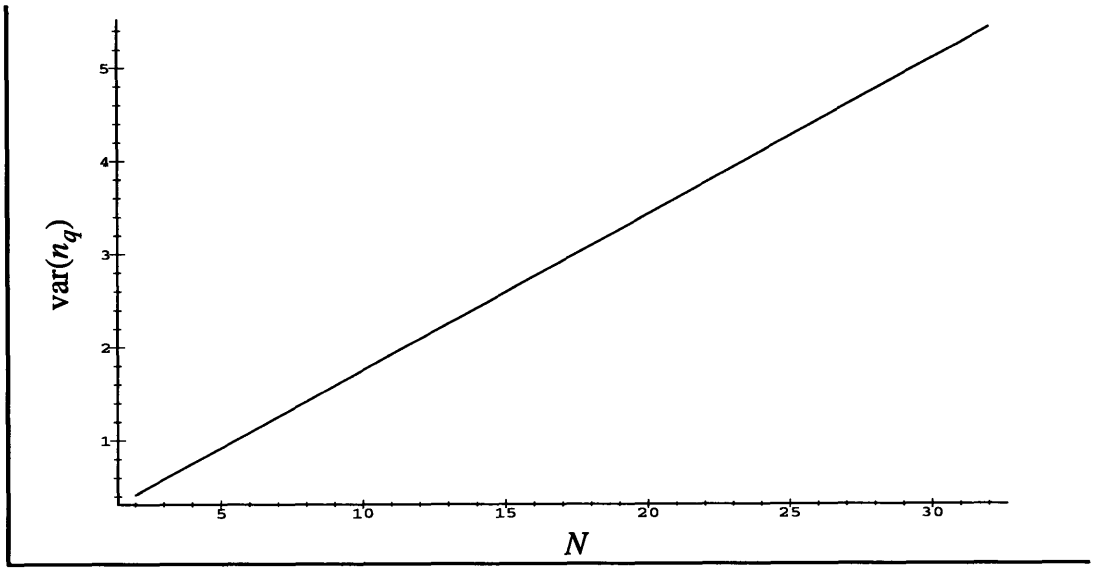


Figure 3.10: Variance of n_q for $N = 2,\ldots,32$.

$$P(D = i|A = a) = \int_{i/N}^{(i+1)/N} f(x)\,dx \tag{3.49}$$

where $f(x)$ is the pdf in Eq. 3.45. Multiplying Eq. 3.49 by p(A=a) (in Eq. 3.44) gives the joint pdf for A and D. This can then be summed over a and d to find E(AD), *i.e*:

$$E(AD) = \sum_{i=0}^{N-1}\sum_{a=0}^{N} (ai)\,P(D = i|A = a)\,\frac{1}{N+1} \tag{3.50}$$

Using Eq's 3.45, 3.47, and 3.50 then Eq. 3.43 can be evaluated numerically for a given N to find the variance in the Q. Figure 3.10 shows the variance of n_q for values of N from 2 to 32. In this figure the variance of n_q appears increase linearly as N increases. With the loop gain normalized by $1/N$, the covariance response from n_q to the queue length is independent of N. Thus, the variance in the queue due to n_q will also increase linearly with the number of sources. As shown in Section 3.3, the variation in the queue due to n_q for 2 sources is an order of magnitude less than the variation due to the probabilistic noise. Both of these variations increase linearly with N, so, for large values of N, the variation in the queue is still dominated by the probabilistic noise.

Next we consider extending the algorithm to a network with several hops like the one in Figure 2.3. The probabilistic feedback algorithm will also suffer from a 'beat-down' problem, as did the ATM Forum's binary algorithms discussed in Section 2.2. To demonstrate, we consider a VC which traverses 2 switches, which are marking cells with probabilities, 0.7 and 0.8 respectively. The cells in this VC will then get marked with probability 0.94. This VC will reduce its rate to a lower level than another VC which is only going through one of these switches.

One way to avoid this problem, as with the binary algorithms in Chapter 2, is to implement intelligent marking, where each switch only marks RM cells from those VC's that are above a fair rate. A second way to avoid this problem is to use explicit feedback, which is the subject of Chapter 4.

Chapter 4

Explicit Rate Algorithms

4.1 Motivation

The binary algorithm examined in the previous chapter used probabilistic feedback to convey a noisy estimate of a multi-valued feedback variable to the source. This noise results in fluctuations in the steady-state queue size, and these fluctuations get significantly worse as the number of active sources increase. Reducing these fluctuations requires lowering the loop gain, with the accompanying slowing down of the transient response. The use of probabilistic feedback also was shown to result in a 'beat-down' effect similar to that seen in the ATM Forum's binary algorithms [Bar94]. When the feedback parameter, p, is fed back directly, the noise, n_i, is not present and the steady-state queue fluctuations are reduced without lowering the loop gain.[1] Feeding back a parameter directly requires an algorithm similar to the ATM Forum's explicit rate algorithms, which use the ER field in the RM cells to carry the feedback information. With this type of feedback the switches can use the information in the ER field to perform "intelligent marking" and avoid the beat-down effect as was done in the ATM Forum's explicit rate algorithms.

If the parameter p is fed back directly, the restriction of this parameter to be a probability is unnecessary. This removes some of the design constraints that were needed in designing the probabilistic feedback system. Once the feedback parameter is allowed to take on a continuous range in values, the N source/1 switch problem can now be modeled as shown in Figure 4.1. The problem is to find a controller H(z) to produce the desired

1. A multivalued feedback variable will still be fed back as a binary string using a finite number of bits. This will result in some quantization noise entering the system, but the variance due to this quantization will be negligible compared to the variance of n_i in the binary case and will be ignored.

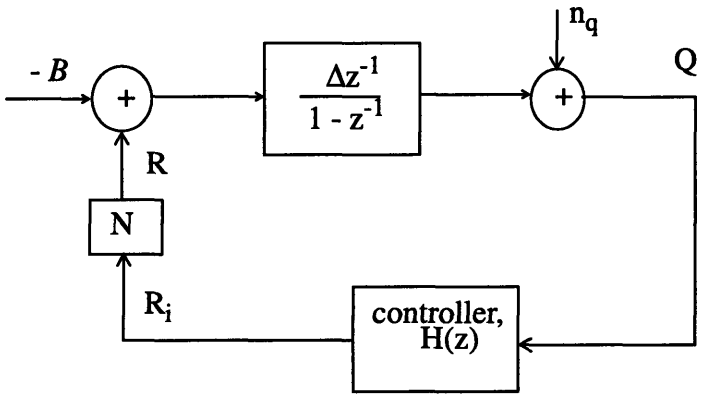


Figure 4.1: Model for continuous valued feedback system.

performance. This controller could be implemented in various ways. H(z) could be implemented at the various sources; then the switch would simply feedback the queue size, and based on this information each source could calculate the rate at which it should transmit. H(z) could be implemented at the switch; then the feedback information would simply be the rate at which the source is allowed to transmit. H(z) could also be implemented partly at the switch and partly at the source; this would be similar to the method used with probabilistic feedback. In this chapter we consider the second of these options. Implementing the controller at the switch has several advantages. The switch sends back the same rate to all sources, which induces a degree of fairness in the algorithm, *i.e.* no unconstrained source will be allowed a rate greater than the other unconstrained sources through a specific output port. This has the effect that when sources start at different times, within one round trip they will be at the same rate. Another advantage of implementing H(z) in the switch is that, since the switch now feeds back a rate, this class of algorithms is very similar to the ATM Forum's explicit rate algorithms as discussed in Chapter 2.

Since probabilistic feedback is not being used, the sources do not have to average the feedback information over any interval. The switch will still need some interval over which it calculates the feedback information and we will still consider this to be a fixed interval, Δ. The sources can simply respond instantaneously to the feedback information received in the backward RM cells. All RM cells marked during a Δ-interval at a given

switch will contain the same feedback information. Assuming all RM cells in each Δ-interval experienced the same delay getting to the source, the source will adjust its rate to the fed back value upon receiving the first RM cell in a interval, and keep its rate at that value for the rest of the interval. With probabilistic feedback the source must wait till the end of the interval to adjust its rate. So by explicitly feeding back the rate, the source adjusts Δ seconds quicker than if any type of processing had to be done on all the RM cells in a given interval. This is another benefit of having H(z) implemented solely at the switch.

In this chapter, classical control theory is used to design the controller H(z) to meet the requirements of a congestion control algorithm discussed in section 1.2. Several controller designs are examined and simulations of the resulting algorithms are presented.

4.2 Proportional compensator

Before considering the first controller model, some details of the model shown in Figure 4.1 need to be explained. The top part of this model represents the linear queue model of Eq. 3.2. The characteristics of the noise, n_q, are the same as for the probabilistic feedback case. As in the previous chapters, the initial model we will consider is for the 1 switch/2 source network topology shown in Figure 2.2. With the multi-valued feedback described above, after one round trip both sources will be at the same rate and thus the rate which cells arrive at the queue can be represented by the doubling of R_i, the rate sent back to each source. This is indicated in Figure 4.1 by setting N=2. Setting N to a different integer value generalizes this model to the 1 switch/N source network of Figure 3.9.

4.2.1 Model

A simple choice for H(z) is to simply use a proportional controller, in other words make the rate, $R_i[n]$, proportional to the instantaneous queue size, Q[n]. When the queue size increases, the rate should decrease, thus the controller should implement the equation:

$$R_i[n] = L - KQ[n] \tag{4.1}$$

where L and K are positive constant parameters. The constant K is the proportionality constant and the constant L is necessary to make sure that the resulting rate is a positive

value. If only one source is sending data through the output port of the switch and the queue is empty, then the rate sent back to that source will be L. If L is less then the output bandwidth, the queue will never grow and the rate will stay at L resulting in the link being underutilized. This suggests that L should be chosen to be greater than or equal to the output bandwidth, B. If there are N sources beginning transmission and the queue is initially empty, then the rate sent back to all the sources will be L, and thus in the next time interval the total input rate will be NL. If NL is larger than the output B then the queue size will begin to grow towards $(NL\text{-}B)\Delta$. This suggests that L should be chosen as small as possible to minimize the growth of the queue. Combining these two considerations we chose $L=B$.

In the steady-state, Eq. 4.1 results in

$$\overline{R_i} = L - K\overline{Q} \tag{4.2}$$

Also, in steady-state we have

$$N\overline{R_i} = B \tag{4.3}$$

Combining these yields the following equation for the steady-state queue size.

$$\overline{Q} = \frac{LN - B}{NK} \tag{4.4}$$

for our choice of $L=B$ this reduces to

$$\overline{Q} = \frac{B(N-1)}{NK} \tag{4.5}$$

Thus, if only one source is transmitting, the steady-state queue will be zero. Taking the derivative of Eq. 4.5 with respect to N results in

$$\frac{d\overline{Q}}{dN} = \frac{BK}{(NK)^2} \tag{4.6}$$

This is always positive, implying as N increases (*i.e.* more sources in the network), the steady-state queue size increases. In the limit as N gets infinitely large, $\overline{Q}$ will approach B/K. For an arbitrary number of sources the steady-state queue size will be between 0 and B/K.

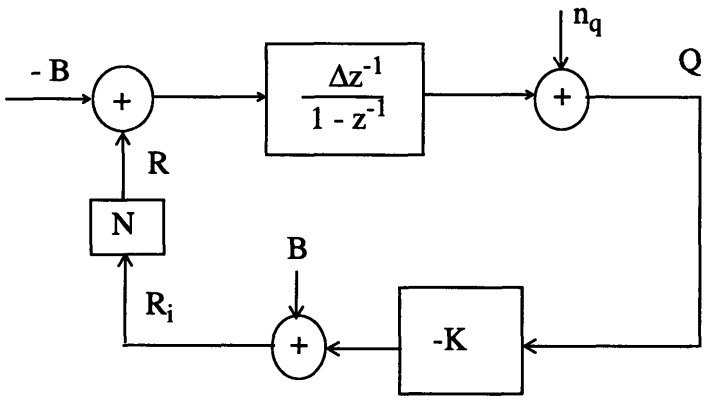


Figure 4.2: Closed loop model with proportional controller.

Equation 4.4 also provides another justification for choosing L as small as possible. A larger choice of L would result in a larger steady-state queue size, which is undesirable.

The closed loop system with the above compensator is shown in Figure 4.2. The loop gain, $G(z)$, for this compensator is given by

$$G(z) = \frac{KN\Delta z^{-1}}{1 - z^{-1}} \tag{4.7}$$

The Nyquist plot for this loop gain is shown in Figure 4.3. Stability of the closed-loop system requires that

$$KN\Delta < 2 \tag{4.8}$$

Also, if $KN\Delta$ is too close to 2 then the nyquist plot will come close to -1 and the closed-

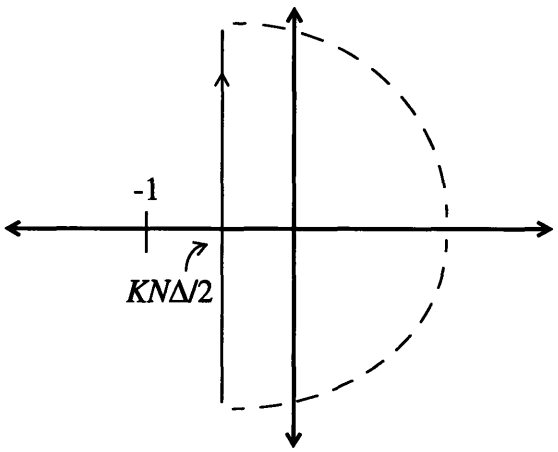


Figure 4.3: Nyquist plot of loop gain.

loop system will experience oscillations, which will be increasingly less damped as $KN\Delta$ approaches 2.

Figure 4.4 shows a Bode plot of the loop gain (Eq. 4.7), for the case where $KN\Delta = 1$. The phase margin and the crossover frequency are indicated on this figure. As in Section 3.3, these quantities can be used to determine the robustness of the closed loop system to delays. If we let ω_C be the normalized crossover frequency, then for a given sampling time Δ, the crossover frequency is $\frac{\omega_C}{\Delta}$, thus the maximum delay that can be tolerated is given by:

$$\text{max delay} = \left(\frac{PM\,(rads)}{\omega_C}\right)\Delta \tag{4.9}$$

where PM(rads) is the phase margin in radians. For the loop gain in Figure 4.4, the maximum tolerable delay is about Δ seconds. Thus increasing Δ, while keeping $KN\Delta$ constant, would make the system robust to larger delays. The switch sends back new feedback information every Δ seconds, so increasing Δ lengthens the time it takes for the network to respond to a change in traffic conditions. On the other hand, decreasing Δ will shorten the time for the switch to throttle a source when it becomes congested, but only up to the

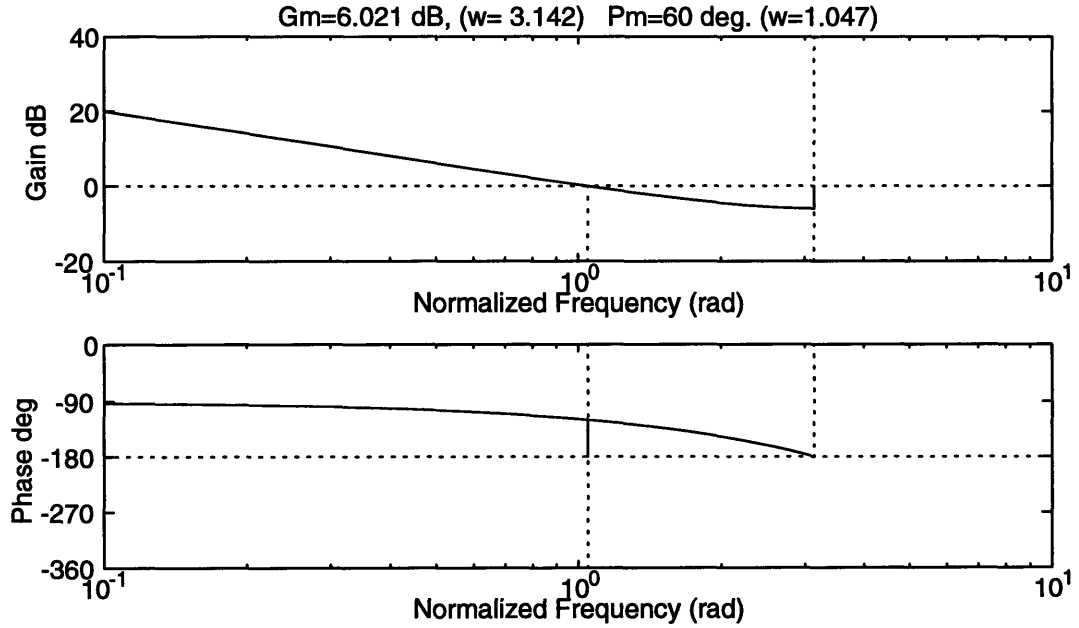


Figure 4.4: Bode plot of the loop gain with $KN\Delta = 1$.

point where Δ is still large enough that the switch sees at least one RM cell from the source in every Δ interval. Once Δ gets smaller than the interarrival time between RM cells, then this interarrival time becomes the limiting factor in the response time of the system. Thus a trade-off exists between choosing Δ to improve the robustness to delays and to improve the response time of the system. Also, increasing Δ and keeping $KN\Delta$ constant, means that K is decreasing, which will increase the steady-state queue size (From Eq. 4.4).

The robustness to delays can also be improved by lowering the loop gain, and thus increasing the phase margin and decreasing the crossover frequency. This has the same effects as increasing Δ. Decreasing the loop gain will also slow down the transient response and will increase the steady state queue size (from Eq. 4.4).

As an example of these trade-offs, if we choose $KN\Delta = 1$ (as in Figure 4.4) and $\Delta=9\text{x}10^{-4}$ seconds, then the system will be robust to delays up to about $9\text{x}10^{-4}$ seconds and will have a steady state queue size of 318.6 cells for 2 sources. Changing the loop gain to $KN\Delta = 0.8$, increases the phase margin to 66 degrees at a crossover frequency of 0.82 radians. This yields a robustness to delays up to 1.73 msec, but a steady state queue-size of 398.25 cells.

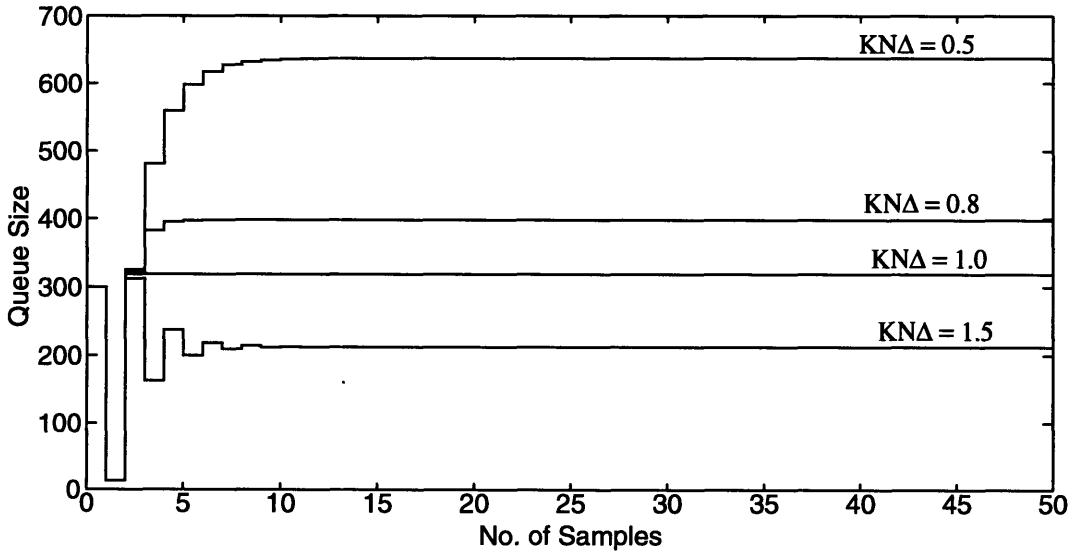


Figure 4.5: Simulation results of the queue size for various loop gains.

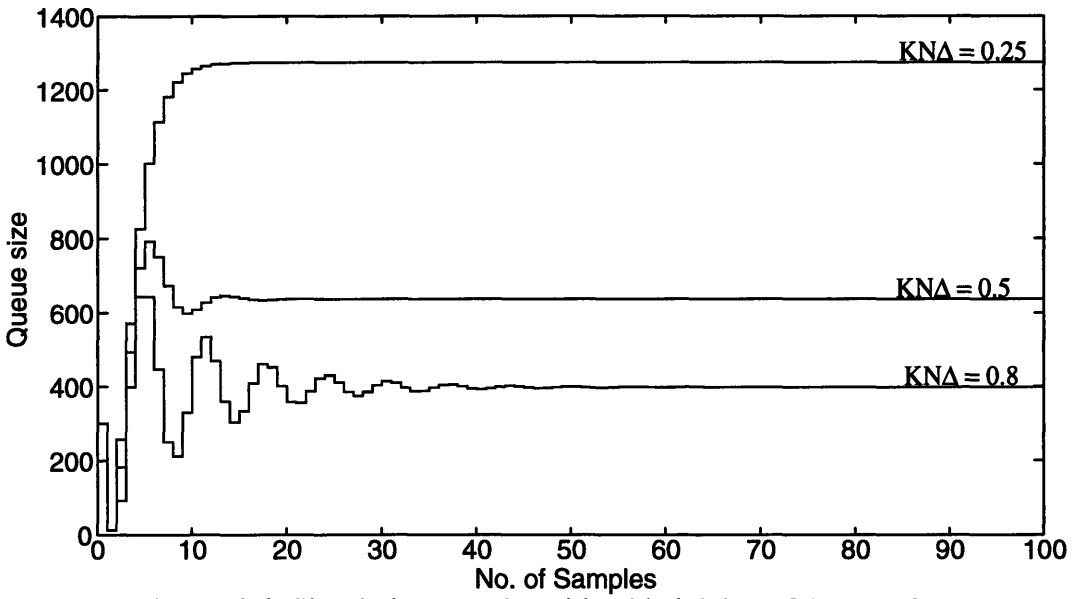


Figure 4.6: Simulation Results with added delay of Δ seconds.

Figure 4.5 shows simulation results of this algorithm for various values of the loop gain. The simulations are for the two source/one switch network with no line delays. The abscissa points are samples placed Δ seconds apart. As the loop gain is increased the steady-state queue size decreases as predicted. The transient response also speeds up as the loop gain is increased. Once the loop gain is increased past 1, the transient begins to exhibit oscillations. This is explained by the Nyquist plot approaching the -1 point as previously discussed.

Figure 4.6 shows simulations for the same system with an added delay of Δ seconds in the loop. With this added delay the linearized system is marginally stable for $KN\Delta = 1$ and is unstable for larger values of $KN\Delta$. As shown in this plot the transient response will now exhibit oscillations for values of $KN\Delta$ as low as 0.5. This is also predicted by Nyquist theory. The delay adds a pole at the origin to the loop gain (Eq. 4.7) which pulls the Nyquist plot closer to the -1 point and causes oscillations for lower values of $KN\Delta$.

The gain, $KN\Delta$, must be chosen small enough to assure stability for the worst possible delay in the system. If the network normally operates with a delay much less than the worst case, then $KN\Delta$ must still be chosen to be small enough to assure stability under the worst case.

As an example we consider the 2 source/ 1 switch network whose simulation results are shown in Figures 4.5 and 4.6. If the network normally operates with negligible delay, but the worst possible delay is 2.6Δ seconds, then we must choose $KN\Delta<0.5$ to assure stability. From Figure 4.5, under normal conditions the transient takes about 10Δ sec. From Eq. 4.5, the steady state queue size is approximately $2B\Delta$. Another possible design for this system would be to increase Δ. If we choose $\Delta'=2.6\Delta$, then the max delay is now Δ'. To assure stability we can now choose $KN\Delta' < 1$. The transient under normal conditions will now take approximately $2\Delta'$ sec. or 5.2Δ sec, but the steady state queue size is now $2.6(B\Delta)$. Thus, we have sped up the system's transient response, but also increased the steady-state queue size. The actual trade-offs are more complicated than this example, since increasing the steady-state queue size may increase the expected delay for the second case and increasing Δ can also add an additional delay, but this serves to illustrate some of the possible trade-offs.

Several difficulties of this proposed scheme are discussed next. The first of these is that, as with the probabilistic algorithm in Chapter 2, the gain, $KN\Delta$, depends on the number of sources using the particular output port of the switch. As this number increases, the gain also increases, and eventually this could result in the linearized system becoming unstable. Thus the gain must be chosen so that when the maximum number of sources are using the output port, the system is still stable. When the number of sources is much less than this maximum, then $KN\Delta$ will also be smaller, and the system will thus be slower. In the previous example when only one source is using the network, $KN\Delta' =0.5$, and the transient will take about 5 times as long as when both sources are active. With a greater variance in the number of sources this effect becomes even worse.

As with the probabilistic system, a possible solution to this problem is to periodically change K based on an estimate of N to keep $KN\Delta$ constant. Possible methods of estimating N are discussed in Section 3.4. If K is adjusted to keep KN constant, then from Eq. 4.5, the steady-state queue size will grow linearly with N. If K is adjusted so that KN decreases as N increases, then some of this queue growth can be traded-off for a slower transient.

There is a second problem with this algorithm. To achieve more robustness to delays a smaller loop gain and a larger Δ are required. Both of these result in a larger steady-state

queue size, which in turn can increase the delay and require more memory at the switch. As discussed above, this effect becomes worse as more sources are competing for the link. Two solutions to this problem are examined. The first solution is to consider the algorithm in Figure 4.2 as being implemented with all the B terms in the algorithm replaced with terms slightly smaller than B , *i.e.* gB, where $g<1$. The resulting closed loop system is shown in Figure 4.7. In this system, the switch calculates a variable, $Q_V[n]$, which represents the size of a 'virtual queue' with an output bandwidth gB that is less than the actual bandwidth.

In the steady state, the sum of the incoming rates will equal gB and the virtual queue will have a steady state queue size given by the same analysis as before. For the actual queue the steady state rate that cells are arriving is less than the bandwidth; thus the actual queue size must go to zero in the steady state for any choice of parameters for the control loop that results in a stable system. The switch's output bandwidth is not involved in the loop gain, $G(z)$, so a system that is designed using feedback based on the actual queue will have similar transient properties when the actual queue is replaced with a virtual queue. The actual queue's output bandwidth is greater than the virtual queue's, so the actual queue will always be smaller than the virtual queue. Thus the model with virtual queue can be analyzed and the results can be used to upper bound the required size for the actual

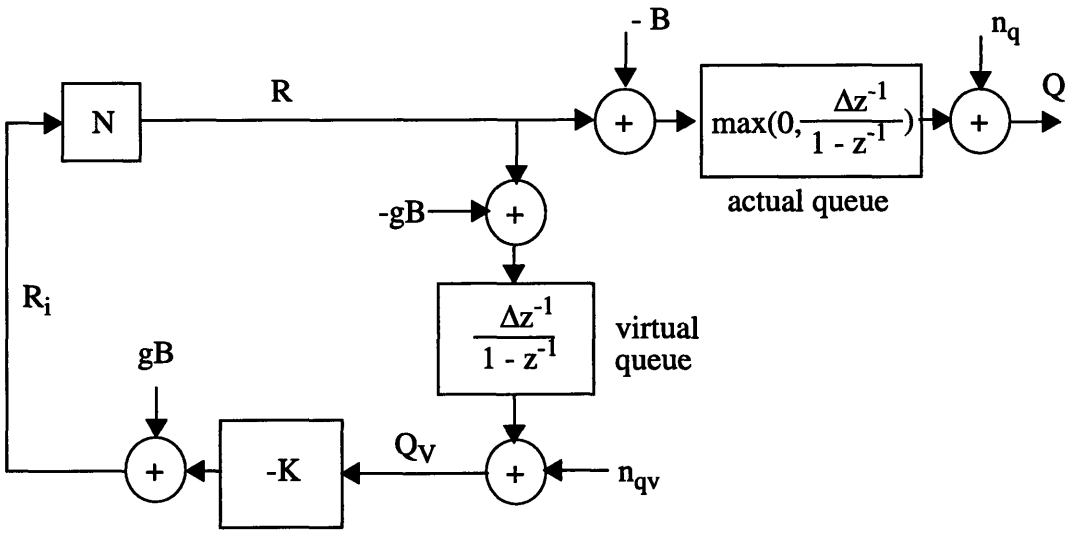


Figure 4.7: Closed loop system using a virtual queue.

queue. The actual queue's size can also be found explicitly by analyzing the model in Figure 4.7, but this requires taking into account the threshold non-linearity of the actual queue. Using the virtual queue has essentially traded-off some steady-state throughput for decreasing the steady-state queue size. The more throughput that is traded off, the quicker the steady-state queue will be driven to zero. All of the ATM Forum's explicit rate algorithms discussed in Chapter 2 set their target bandwidths slightly less than the actual bandwidth to keep the queue lengths small in this same manner.

A second method for keeping the steady-state queue lengths small is to design a different controller, H(z), which will accomplish this. One such controller is a lag compensator which is discussed in the next section.

4.3 Lag Compensator

The proportional compensator, considered in the last section, had the undesirable effect that increasing the compensator's robustness to delays also increased the steady-state queue lengths. If the compensator, H(z), contained an integrator, then the steady-state queue size, which is the input to this integrator would be forced to zero. This suggests that a PI[1] or lag compensator may be a good candidate for H(z).

In this section, the use of a lag compensator is examined. The compensator to be considered has the form:

$$H(z) = \left(\frac{-2K}{1+a}\right)\frac{(z-a)}{(z-1)} \tag{4.10}$$

H(z) can be thought of as a proportional compensator *-K* in series with a lag compensator $H_L(z)$ which, due to the normalizing factor of $2/(1+a)$, has a high frequency gain of one.

From Figure 4.1, *H(z)* relates the rate $R_i(z)$ to the queue size, *Q(z)* according to the equation:

$$R_i(z) = H(z)\,Q(z) = \left(\frac{-2K}{1+a}\right)\frac{(z-a)}{(z-1)}Q(z) \tag{4.11}$$

Converting this equation into discrete time yields:

---

1. The term PI stands for a Proportional plus Integral control law. This is standard terminology in control theory. [RoM93]

$$R_i[n] = R_i[n-1] - \frac{2K}{1+a}(Q[n] - aQ[n-1]) \tag{4.12}$$

If we let G = $\left(\frac{2K}{1+a}\right)$, then this equation can be written as:

$$R_i[n] = R_i[n-1] - aG(Q[n] - Q[n-1]) - (1-a)GQ[n] \tag{4.13}$$

From this equation, it can be seen that the lag compensator results in adjusting the rate that is fed back by an amount proportional to the queue size and by an amount that is proportional to the change in the queue size over the last measurement interval. The zero location, *a*, determines the relative weighting of these two terms. The change in the queue size is proportional to the rate that cells are entering the queue. Both the Jain [JaS95] and Barnhart [Bar95] algorithms discussed in chapter 2 also attempt to measure the rate that cells are entering the queue and use this rate in determining their feedback.

A problem with implementing this proposed controller can be seen by examining Eq. 4.13. If the queue size, *q[n]*, is zero and the total rate cells are arriving at the queue is less than the output rate, *B*, then the change in the queue size will also be zero. Under these conditions, Eq. (4.13) reduces to:

$$R_i[n] = R_i[n-1] \tag{4.14}$$

In other words, the rate fed back to the sources will not increase. This means that if the queue is empty and there is available bandwidth, the proposed lag controller will not bring the total rate up to the available bandwidth. The cause of this short coming is that when the queue is driven to zero, one must consider the threshold in Eq. 3.2, which prevents a negative queue size. Several solutions to this problem will be examined later. In the following, we ignore this problem and look at the performance of the lag controller in reducing the incoming rates when they are greater then the available bandwidth. In this case the queue size is greater than zero, and the non-linearity can be ignored.

The loop gain of the closed loop system with the lag compensator is:

$$H(z) = \left(\frac{2KN\Delta}{1+a}\right)\frac{\left(1-az^{-1}\right)z^{-1}}{\left(1-z^{-1}\right)^2} \tag{4.15}$$

The normalization factor 2/(1+*a*) in Eq. 4.15 causes the closed-loop system with a lag

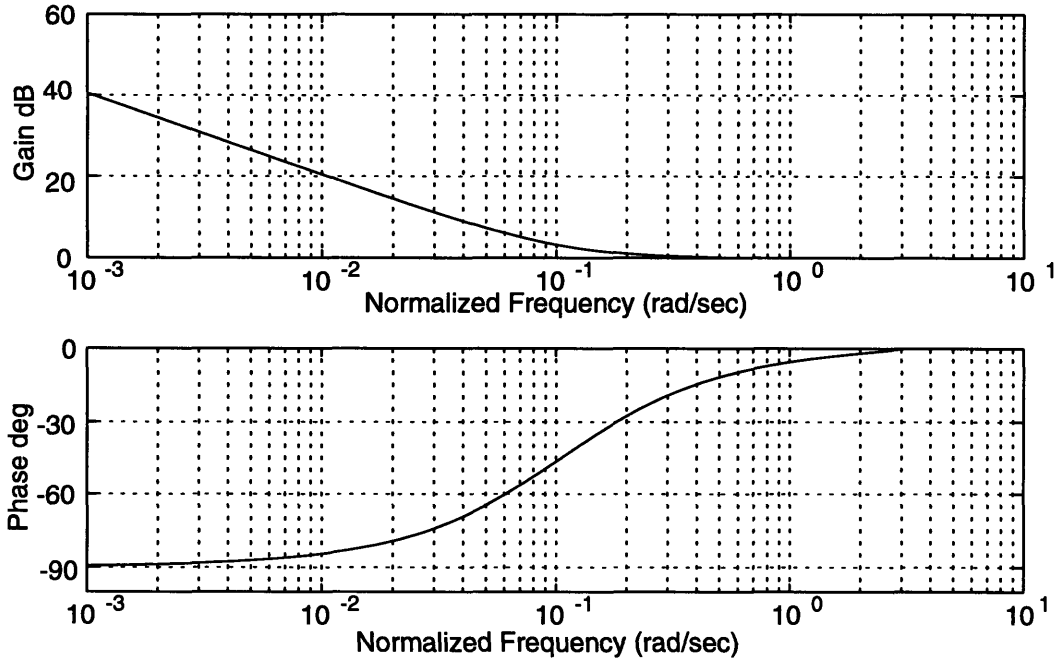


Figure 4.8: Bode plot of lag compensator, $H_L(z)$.

compensator to have the same gain margin as the closed-loop system with only a proportional compensator, K. Thus, the condition that $KN\Delta < 2$ given in Eq. 4.8 must still be satisfied for stability. Now, stability also depends on the choice of the parameter a which locates the zero of the compensator. A Bode plot of $H_L(z)$ is shown in Figure 4.8, for a=0.9. The parameter a determines the corner frequency of this plot. Moving a closer to the origin increases the corner frequency which will increase the phase lag added to the loop gain by the compensator, and thus decrease the over all robustness to delays. Moving a towards the origin also increases the low frequency gain, which leads to a faster transient.

The above trade-offs in choosing the parameter a are the standard choices involved in using a lag compensator in a typical control problem [RoM93]. This suggests a possible design methodology. First we design a proportional compensator, K, to get adequate robustness to delays, as in Section 4.2. Then consider adding a lag controller, $H_L(z)$, into this loop. If a is chosen so that the lag compensator's corner frequency is far enough below the closed-loop crossover frequency, then little phase lag will be added at crossover. This will result in the system with the lag compensator having about the same robustness

to delays as the system with just the proportional compensator, but with the added effect that the steady-state queue size will now go to zero.

The placing of the zero, a, of the lag compensator for a continuous-time system is simplified by using well-known asymptotic approximations for the Bode plot of the compensator [RoM93]. Such approximations are not available for a discrete-time system. One approach to designing discrete-time compensators is to transform the problem into a continuous-time problem and design the corresponding continuous-time compensator. The designed compensator is then transformed back to discrete-time.

One method of transforming between continuous-time and discrete-time systems is with the impulse invariance technique [OpW83]. This corresponds to transforming the discrete-time transfer function, $H(z)$, to a continuous-time transfer function, $H(s)$, by the mapping $z = e^{s\Delta}$. Using this mapping, the continuous-time frequency response is equal to the discrete-time frequency response with the frequency axis scaled by $1/\Delta$. Thus, the crossover frequency, Ω_c, of the continuous-time system is related to the crossover frequency, ω_c, of the discrete-time system by:

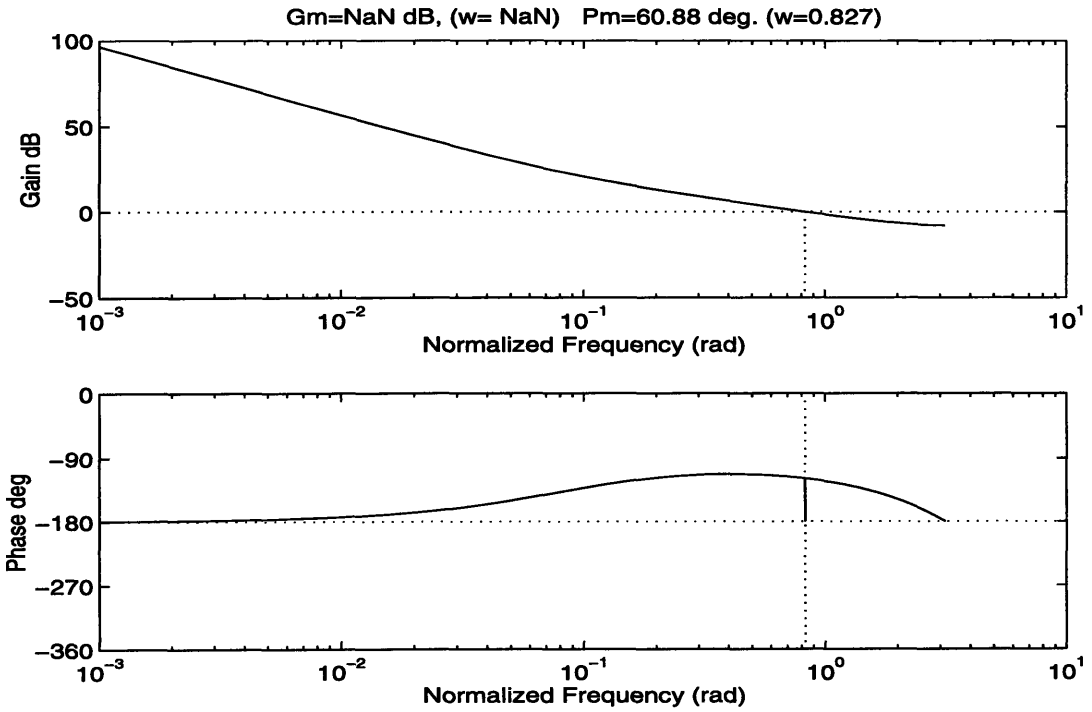


Figure 4.9: .Bode plot for loop gain with lag compensator.

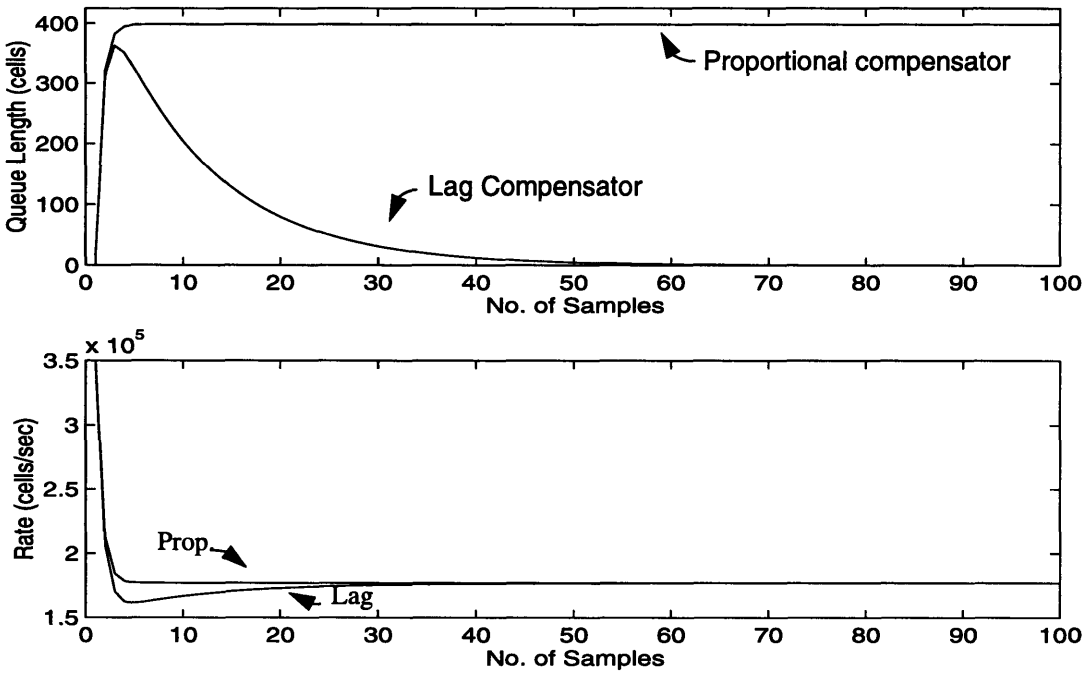


Figure 4.10: Transient response of system with proportional and lag compensators .

$$\Omega_c = \frac{\omega_c}{\Delta} \tag{4.16}$$

The continuous-time design approximations state that if the zero of the lag compensator is placed a decade below the crossover frequency then the amount of the phase lag added by the compensator is already down to about 5 degrees. Thus, if we choose a to be one decade below the crossover in continuous-time, the amount of added phase lag will be small; to further reduce the phase lag, a could be chosen even further below the crossover frequency. Choosing a one decade below the crossover in continuous-time, results in the following value for a in discrete-time:

$$a = e^{-\left(\left(\frac{1}{10}\right)\left(\frac{\omega_c}{\Delta}\right)\Delta\right)} = e^{-(\omega_c/10)} \tag{4.17}$$

As an example of the above procedure, consider the proportional compensator from Section 4.2, with $KN\Delta = 0.8$. This compensator has a crossover frequency of 0.82 radians and a phase margin of 66 degrees; thus it is robust to delays of up to 1.73 msec. Choosing a according to the Eq. 4.17 results in a=.92. The Bode plot of the loop gain with this compensator is shown in Figure 4.9. The crossover of this system is still at 0.82 radians and

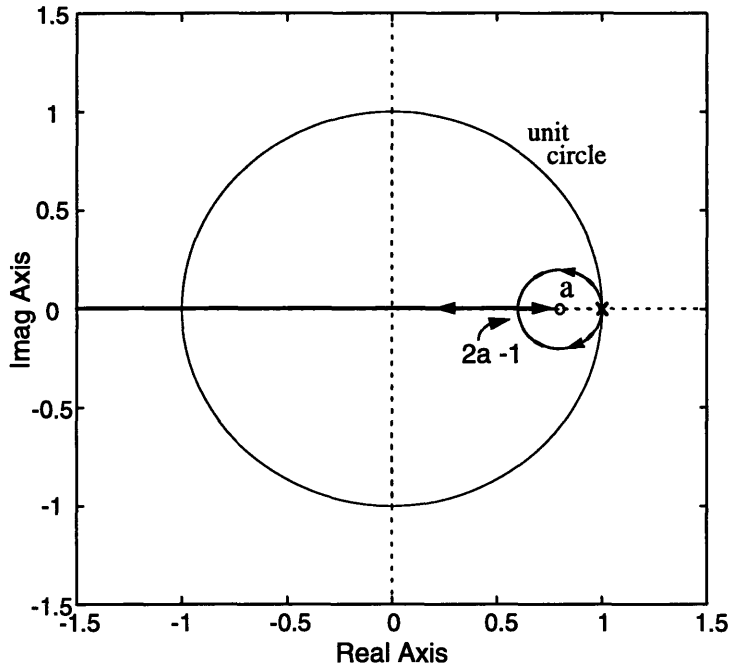


Figure 4.11: Root locus of closed loop system

the phase margin is now 61 degrees. This results in a robustness to delays up to 1.3 msec, nearly the same as for the system with the proportional compensator.

Figure 4.10 shows the transient performance for this system, given an initial rate of 3.54×10^5 cells/sec from each of 2 sources which are competing for the output port of a switch with a bandwidth of 3.54×10^5 cells/sec. For comparison, the response of the same network using the original proportional compensator ($KN\Delta = 0.8$) is also shown. This figure shows that the lag compensator does result in driving the queue toward zero as expected.

Using the above design method, a stable closed-loop system results, with the desired robustness to delay. Can we come up with another design by adjusting K and a that tolerates the same delay, but performs better in some other regard? In the following these possible performance trade-offs are examined in more detail. Specifically, the range of behavior for the closed-loop system, given a maximum delay requirement, is considered.

The response of the closed-loop system will contain two poles. The root locus for these poles is shown in Figure 4.11. For the closed-loop system not to exhibit ocsillations, the two poles must lie on the real axis between zero and one. For this case, we denote the

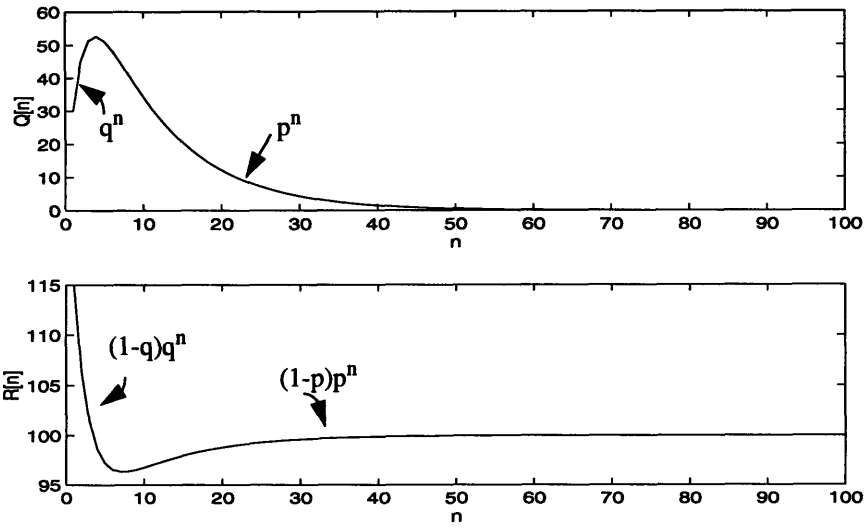


Figure 4.12: Plot of p^n and q^n.

pole closest to a by p, and the second pole by q. The values of p and q are determined by the choice of K and a as shown in the following equations:

$$p = \frac{-KN\Delta}{1+a} + 1 + \sqrt{\left(\frac{(KN\Delta)^2}{(1+a)^2} + \frac{2(KN\Delta)(a-1)}{(1+a)}\right)} \tag{4.18}$$

$$q = a - \frac{(a-1)^2}{a-p} \tag{4.19}$$

Given an initial rate of R_o, the response of the queue in terms of p and q is:

$$Q[n] = \frac{(NR_o - B)\Delta}{p-q}\left(p^n - q^n\right) \tag{4.20}$$

Since p and q are both less than one, then p^n and q^n are both decaying exponentials. The response of the queue consists of the difference between these terms. From our assumptions above, $p > q$, and thus the p term will decay more slowly. Figure 4.12 shows a typical plot of $Q[n]$. Given the condition that $NR_o > B$, the queue will initially grow and the rate of this growth is determined by q. Eventually, the queue will decay to zero, the rate of this decay is determined by p. Given the same initial rate, R_O, the transient for the source rate in terms of p and q is

$$R[n] = \frac{B}{N}u[n] + \frac{(R_O N - B/N)}{(p-q)}\left[(1-q)q^n - (1-p)p^n\right] \tag{4.21}$$

A typical response for the rate is also shown in Figure 4.12. In this case, q determines how quickly the rate initially comes down, and p determines how quickly the rate rises back to its steady state value. If, for example, we desire to minimize the initial growth of the queue, then we would want to choose K and a so that q is as small as possible. This would also result in the rates quickly dropping much lower than their steady-state value.

We would like to understand the possible choices for p and q given a delay constraint. We express the maximum allowable delay as $k_d\Delta$, where Δ is the sampling time. Using this in Eq. 3.9, we have:

$$k_d = \left(\frac{PM\,(rads)}{\omega_C}\right) \tag{4.22}$$

The phase margin as a function of ω_C and the zero location a is:

$$PM = \pi - arctan\left(\frac{sin\,(\omega_C)}{cos\,(\omega_C) - a}\right) - \omega_C \tag{4.23}$$

Equation 4.22 can be substituted into Eq. 4.21 and the resulting expression can be solved to yield a as a function of ω_C and k_d.

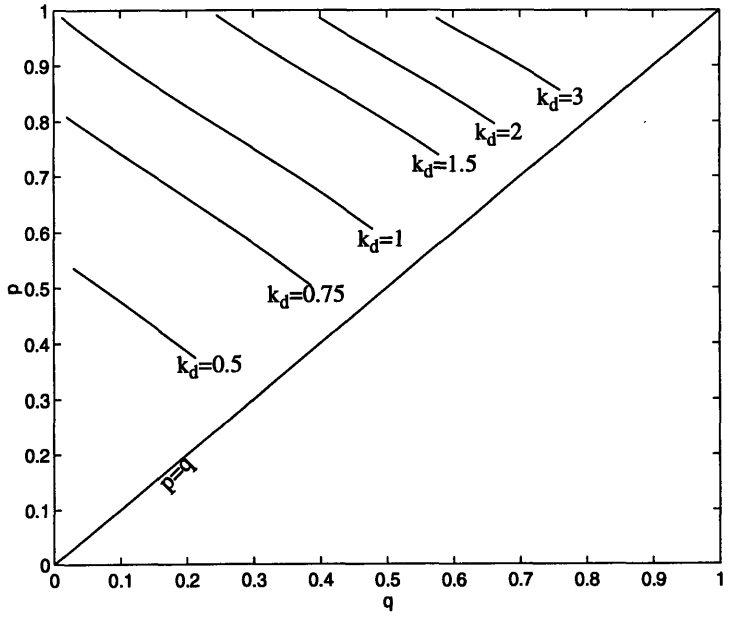


Figure 4.13: Values of p and q for various values of k_d.

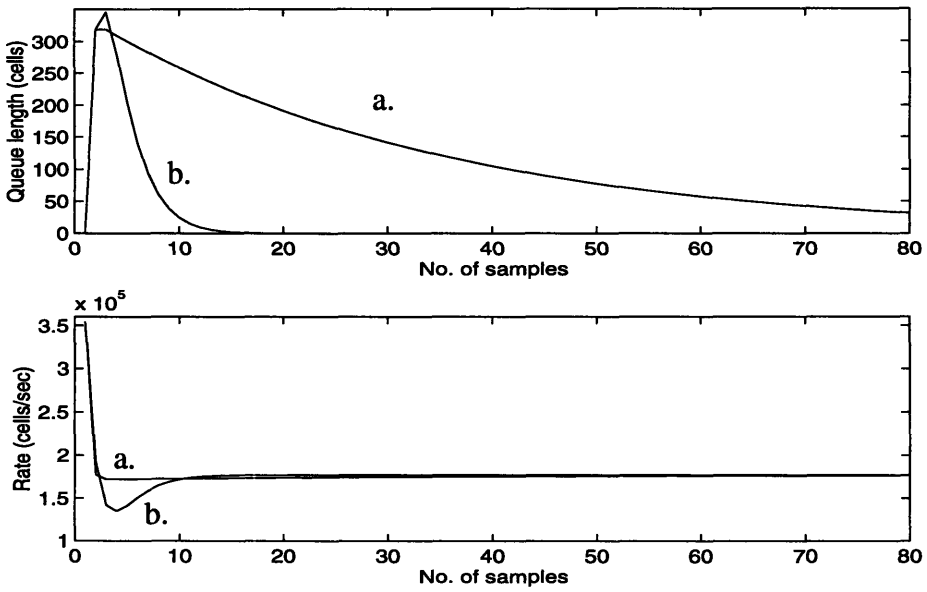


Figure 4.14: Transient performance for maximum of delay of k_d=1 for (a) large p case and (b) small p case.

$$a = \frac{sin\,(\omega_C k_d)}{sin\,(\omega_C (k_d + 1))} \tag{4.24}$$

The gain, $KN\Delta$ can be solved for as a function of a and ω_C.

$$KN\Delta = (1 + a)\frac{(1 - cos\,(\omega_C))}{\sqrt{\left(1 - 2acos\,(\omega_C) + a^2\right)}} \tag{4.25}$$

The normalized crossover frequency, ω_C, can take on only values between 0 and π. For a given k_d (*i.e.* delay constraint), the possible values of a which meet that constraint can be determined by from Eq. 4.24 by sweeping through these values of ω_C. The possible values for K can then be found from Eq. 4.25. Equations 4.18 and 4.19 can be used to translate the allowable K and a into values of p and q. Figure 4.13 shows the resulting values for p and q for several delay constraints.

Several things can be noted from Figure 4.13. First for a given delay constraint, choosing a smaller value of q requires a larger value of p. A small q results in less overshoot in the queue size, but the larger p results in the queue taking longer to reach zero. Figure 4.14 illustrates this trade-off. Also note that the larger q results in more under shoot in the rates. Figure 4.13 also demonstrates that as k_d gets larger, the choices in p and

q become more restricted. To handle more delay in a network, either k_d or Δ must be increased. As with the previous compensators, increasing Δ slows the system down. Increasing k_d requires larger values of p and q, which also result in a slower system.

Choosing the parameters to be used in a given system involves deciding on a choice of Δ and k_d to handle the necessary delay. After k_d is selected the desired p and q can be chosen. These are chosen to reach as a compromise between getting the queue to zero quickly, minimizing the over-shoot, and getting the rates to their steady-state values quickly. Once p and q are chosen, this fixes the values of the parameters K and a.

Next, several difficulties with the lag compensator are examined. First, the problem of increasing the rates when the queue is empty is addressed. As shown in Eq. 4.14, when the incoming rates are less than the available bandwidth and the queue is empty, the lag compensator will not increase the rates. This is due to the fact that the queue size can not be negative. One solution to this problem would be to compute a variable at the switch which behaves just like the queue, but is allowed to go negative and use this variable to calculate the rates fed back. In other words define a variable q_N, where

$$q_N[n] = q_N[n-1] + (R[n-1] - B)\Delta + n_q \tag{4.26}$$

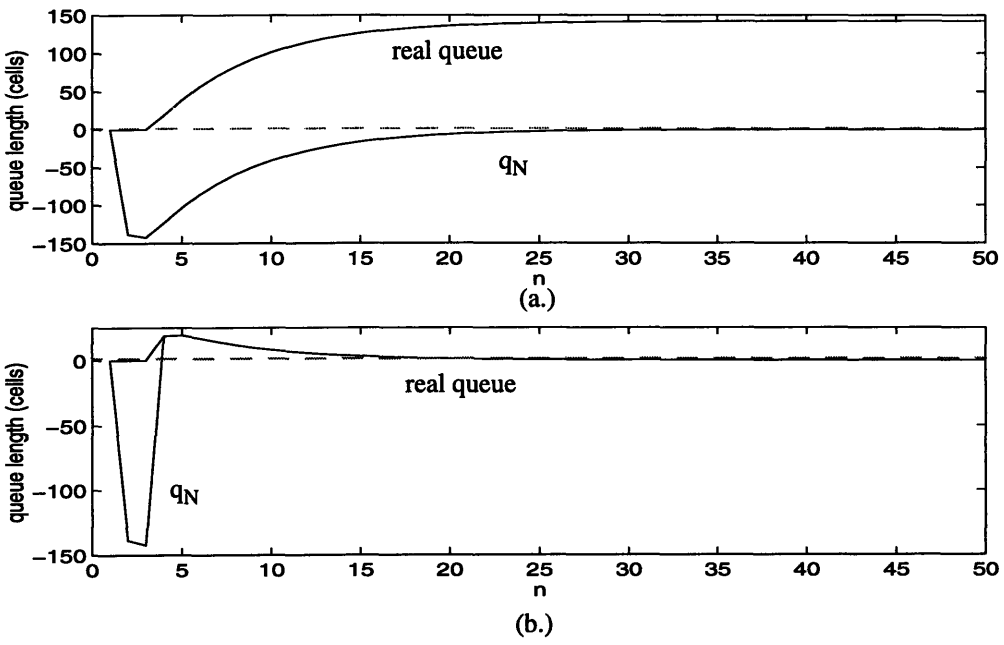


Figure 4.15: Transient response of real queue and q_N.

Now, the non-linearity is removed from the control loop and the rates will converge to the desired value. A problem with this approach is that in the steady-state, q_N is driven to zero, but the actual queue may not be. This effect is shown in Figure 4.15(a). One solution to this problem that does not result in a non-zero steady-state queue size is to calculate the rate that is fed back based on q_N when the queue is empty, but to calculate the rate based on the actual queue size when it is not empty. Furthermore, once the queue becomes positive, we reset q_N to the size of the real queue. Figure 4.15(b) shows the behavior of such an algorithm. When the queue is empty and the rates are low, the system can be analyzed as a linear system implementing Eq. 4.26. and when the queue is not empty the system can be analyzed as the original linear system. The switching back and forth between these systems may result in oscillations in a larger network. The behavior in these circumstances still needs to be addressed. A final point about this algorithm is that there is that the rate we calculate when the queue is empty need not be based on Eq. 4.26, but could be based on any equation that would tend to drive the rates to their steady-state value.

Finally we consider the effect of changing the number of sources in the network. As with the other systems analyzed, the loop gain depends on the number of sources, N. Thus it is necessary to adaptively adjust K to compensate for this. Since the lag compensator forces the queue to zero, it does not have the problem of the steady state queue size growing with the number of sources, as did the previous approaches.

The steady-state queue size going to zero does not necessarily mean that larger buffers are not needed as N increases. For large N, there is a greater chance that a number of sources begin transmitting at the same time. This causes the peak queue size during the transient stage to be larger and requires larger buffers than for a network with a small N. One way to mitigate this effect is for large networks to have a smaller initial rate for the sources and to make sure that the sources are not allowed to increase their rates to quickly. Thus a trade-off exist between how fast sources increase to their full rate and the required buffers in the network.

Chapter 5

Conclusion

5.1 Summary

In this thesis three new approaches to rate-based congestion control were examined. These three approaches were:

1.) Binary approach using probabilistic feedback.

2.) Explicit rate approach using a proportional compensator.

3.) Explicit rate approach using a lag compensator.

All of these approaches were developed using linear control theory. This enabled the various design trade-offs to be analyzed and understood. The various parameters can then be chosen based on this analysis rather than trial and error simulation.

The binary approach using probabilistic feedback succeeded in avoiding the steady-state oscillations exhibited by the previous binary algorithms. A shortcoming with this approach is that the noise involved with the probabilistic feedback requires a slower system (*i.e.* smaller bandwidth) to avoid a large variance in the steady-state parameters. Also the steady-state queue size and the variance in the steady-state queue increased with the number of sources. This approach also suffers from the beat-down effect that is inherent in any binary algorithm.

The use of explicit rate feedback avoids the beat-down effect and the noise involved with probabilistic feedback. This requires the use of more overhead for the feedback information and some additional processing at the switch. Using simply a proportional compensator with explicit rate feedback results in a system with adequate performance. This approach still leads to a non-zero steady-state queue size. To handle longer delays,

this steady-state queue size increases and it also increases with more sources. For small networks the proportional compensator would probably offer sufficient congestion control.

The use of a lag compensator with explicit rate feedback was the final approach examined. The lag compensator helps drive the steady-state queue to zero thus avoiding the problem of a large steady-state queue that the other algorithms have. Since the lag compensator drives the queue to zero, the non-linearity in the queue size at zero must be considered. This requires an addition to the algorithm to increase the incoming rates when the bandwidth is under-utilized. The design of the lag compensator involves additional trade-offs concerning how fast the queue decreases, how much it overshoots and how fast the rates get to their steady-state values.

In the analysis of all three of these approaches some fundamental trade-offs were evident. Increasing the allowable delay of a system always results in a slower system. This implies that users can expect their transmission rates to ramp up more quickly on a small local area network than on a wide area network. A second trade-off is that as the number of users competing for a link increases, either the system must slow down or the needed buffer space must increase. This also implies that wide area networks should expect to be slower and have larger buffers.

In developing these algorithms, several parallels with previous algorithms were pointed out. One of the methods proposed to keep the steady-state queue size near zero was the use a virtual queue. This trades some through-put for a smaller queue size; similar trade-offs are evident in many other algorithms. In all these algorithms and in many others it is necessary to estimate the number of sources competing for the link and use this information in calculating the fedback. This is needed to ensure that the algorithm's performance stays relatively constant regardless of the number of users.

5.2 Future work

There are many aspects of rate-based congestion control that still need to be addressed. In this thesis, only very simple networks were analyzed. The analysis of these algorithms in large networks still needs to be addressed. There are two approaches to using rate based

congestion control in large networks. One approach is for the algorithm to simply work end-to-end for a given connection. With this approach a given VC could receive feedback information from any of the switches that it passes through. The second approach is for the switches to act as virtual sources and destinations. In this case the feedback loop is closed between each pair of switches. This lessens the delay in the feedback loop but requires more complicated switches. The interaction of these algorithms in either of these cases needs to be understood, as well as the advantages/disadvantages of each method.

The estimation of the number of sources using a link is another area of further research. As mentioned in Section 3.4, this can be accomplished by adaptively keeping the loop gain constant. The precise method for implementing this and its performance needs to be examined.

In all the situations considered here the data sources were 'greedy' and always on. In a practical network the data sources are very bursty and the number of sources that are using a link is changing. Analysis of these algorithms under these conditions is needed.

For the explicit rate case, there is a problem when a number of sources begin using a link that was previously not being used. When the sources receive back their initial feedback, they will all get permission to increase their rates to the maximum value. If they all begin transmitting at this rate, the queue will quickly build up and cells may be lost. A method to control the rate that the sources increase their rates may be desired. Such a method would have to consider the trade-offs between minimizing the queue growth and slowing down the rate increase.

Finally, the interaction of switches using different switch algorithms needs to be considered. The interaction of rate based congestion control in a network with other service types, using different congestion control strategies is also of interest.

References

[AsW95] Astrom, K. and B. Wittenmark. *Adaptive Control.* Addison Wesley, 1995.

[Atm96] ATM Forum. "ATM Forum Traffic Management Specification Version 4.0-Draft Version," AF-TM 95-0013R10, Feb. 1996.

[Bar94] Barnhart, A. "Use of Extended PRCA with Various Switch Mechanisms," AF-TM 94-0898, Sept. 1994.

[Bar95] Barnhart, A. "Example Switch Algorithm for Section 5.4 of TM Spec.," AF-TM 95-0195, Feb. 1995.

[BeG92] Bertsekas, D. and R. Gallager. *Data Networks.* Prentice Hall, 2nd Edition, 1992.

[BoF95] Bonomi, F. and F.W. Fendick. "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," I*EEE Network* Vol. 9, No. 2, March/ April 1995, pp 25-39.

[Cha94] Charny, A. *An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback.* Master's Thesis, MIT Lab for Computer Science, 1994.

[Cci88] CCITT. *Blue Book.* Volume III, Fascicle III.7, Recommendation I.121, "Broadband Aspects of ISDN," 1988.

[Gal96] Gallager, R. *Discrete Stochastic Processes.* Kluwer Academic, 1996.

[Gor95] Goralski, W.J. *Introduction to ATM Networking.* McGraw-Hill, 1995.

[JaS95] Jain, R., S. Kalyanaraman, and R. Viswanathan. "A Sample Switch Algorithm," AF-TM 95-0178, Feb. 1995.

[Kav95] Kavak, N. "Data Communication in ATM Networks," *IEEE Network* Vol. 9, No. 3, May/June 1995, pp 28-37.

[KuC93] Kung, H. and A. Chapman. "The FCVC (Flow-Controlled Virtual Channels) Proposal for ATM Networks: A Summary," Proc. 1993 International Conf. on Network Protocols, 1993, pp 116-127.

[KuM95] Kung, H.T. and R. Morris. "Credit-Based Flow Control for ATM Networks," *IEEE Network* Vol. 9, No. 2, March/April 1995, pp 40-48.

[New94] Newman, P. "Traffic Management for ATM Local Area Networks," *IEEE Communication Magazine*, Aug. 1994, pp 44-50.

[OpW83] Oppenheim, A. and A. Willsky. *Signals and Systems.* Prentice Hall, 1983.

[RoM93] Rohrs, C., J. Melsa, and D. Schultz. *Linear Control Systems.* McGraw-Hill, 1993.

[RoB96] Rohrs, C., R. Berry and S. O'Halek. "A Control Engineer's Look at ATM Congestion Avoidance" To be published in *Computer Communication Review.*

[Rob94] Roberts, L. "Enhanced PRCA (Proportional Rate-Control Algorithm)," AF-TM 94-0735R1, Aug. 1994.

[RoF95] Romanow, A. and S. Floyd. "Dynamics of TCP Traffic over ATM Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 4, May 1995, pp 933-641.

[SiT95] Siu, K. and Tzeng, H. "Intelligent Congestion Control for ABR Service in ATM Networks," *Computer Communication Review* Vol. 24, No. 5, October, 1995, pp. 81-106.

[YiH94] Yin, N. and M. Hluchyj. "On Closed-Loop Rate Control for ATM Cell Relay Networks," Proc. INFOCOM 94, 1994, pp 99-108.

2304.4