A Linear Control Approach to Explicit Rate Feedback in ATM Networks

Charles E. Rohrs
Tellabs Research Center, 3740 Edison Lakes Pkwy, Mishawaka, IN 46545
phone: (219)258-6417, fax: (219)258-6446, e-mail: charlie@trc.tellabs.com

Randall A. Berry
MIT, 77 Massachusetts Ave., Rm 35-303, Cambridge, Ma 02139
phone: (617)253-2147, email: randy1@mit.edu

Abstract

Rate-based feedback congestion control has been proposed as a form of traffic management for available bit rate traffic in ATM networks. This paper discusses applying linear control theory to these algorithms. A congestion control scheme for simple networks is designed and analyzed using the tools of classical control theory. This allows insight into the trade-offs in such schemes and suggests approaches to larger networks.

I. Introduction

Traffic management is necessary in an ATM network to enable the network to be utilized to the fullest extent while ensuring that each user receives their required service in a fair manner. Traffic management is accomplished in ATM in a variety of ways, such as admission control, traffic shaping and various queueing strategies. A method of traffic management used for the available bit rate (ABR) class of traffic is rate based feedback congestion control. This method allows the switches in a network to feedback congestion information to the sources, which then adjust their transmission rates accordingly. The goal of this system is to efficiently and fairly share the available bandwidth between all the users while avoiding cell-loss from queues overflowing. The ATM Forum has established a framework [2] within which these rate based algorithms must operate. Several example algorithms have also been developed to demonstrate possible implementations, but a specific algorithm is not being standardized.

The algorithms suggested to date fall into two categories-- binary algorithms, which use a single bit of feedback to communicate congestion information, and explicit rate algorithms, which allow the switches to feedback the maximum allowable rate for a source. The sample algorithms presented to date consist of mainly heuristic approaches which contain significant non-linearities. Such schemes are often difficult to analyze even in a single loop setting, and simulation must be relied upon to verify performance.

In this paper, we examine the rate-based congestion control problem using standard linear control theory. In particular, explicit rate feedback algorithms are developed using this theory. In [6] we examined a similar control theoretic approach to algorithms using binary feedback. Linear control theory provides an established set of tools which can be used to gain insight into rate based congestion control. Using this theory we are able to developed algorithms whose performance can be predicted analytically rather than relying on simulations. We are also able to determine the design trade-offs inherent in these algorithms and gain some understanding of the current algorithms.

II. Network model

In this paper, we consider the congestion control problem in a simple network. This enables a straightforward analysis, while still providing insight. In the final section, extensions to more complicated networks are examined. The network we consider has N sources trying to send cells to the same output port of a switch as shown in Fig. 1. Furthermore, we assume that the sources are 'greedy', *i.e.* they always have data to send and will use the maximum rate they are allowed. If we denote the bandwidth of the bottleneck link by B, then the congestion control strategy should allow each source to transmit at rate B/N in steady-state.

To enable a discrete-time analysis, we assume that the switch calculates a new feedback variable every Δ seconds. This variable is then fed back to the sources. In the ATM Forum's guidelines, this feedback is accomplished

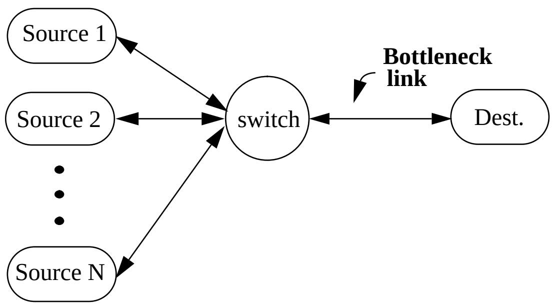


Figure 1: *N* source/1 switch network.

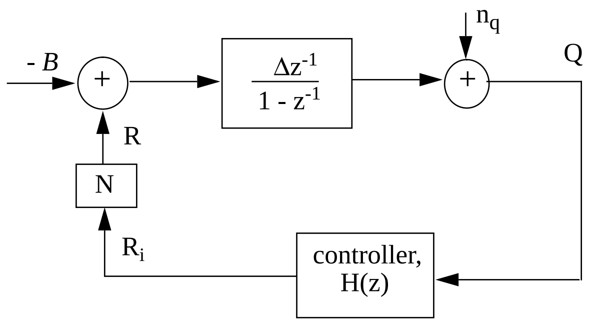


Figure 2: Model for the *N* source/1 switch network.

through the use of a Resource Management (RM) cell. These RM cells can be generated in a variety of different ways. This feedback is then received back at the sources after some delay. For the explicit rate methods considered here, this feedback is simply the maximum allowed rate at which the source may transmit. Under our assumption of greedy sources, the sources will always transmit at this maximum.

The switch queues cells at its output port until they can be delivered. If cells arrive at the switch with rate, *R*, then the size of the queue, *Q*, at successive time instants Δ seconds apart is modeled using the following fluid approximation:

$$Q[n] = max(Q[n-1] + (R[n-1] - B)\Delta + n_q, 0) \quad (1)$$

The noise term, n_q, accounts for the difference between the fluid approximation and the actual queue size. This term can be modeled as additive white noise as explained in [6].

The above model for the *N* source/ 1 switch system is shown in Fig. 2. The top part of this figure is the queue model from Eq. 1 operating in the region where the queue size is strictly positive. Based on the queue size, a controller, *H*(*z*), must be found which calculates the desired rate to be fed back to the sources. Since all the sources receive the same feedback, and thus transmit at the same rate, the total rate cells are arriving at the queue is *N* times the rate fedback.

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

$$R_i[n] = L - KQ[n] \quad (2)$$

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*-*B*)Δ. 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. 2 results in

$$\overline{R_i} = L - K\overline{Q} \quad (3)$$

Also, in steady-state we have

$$N\overline{R_i} = B \quad (4)$$

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

$$\overline{Q} = \frac{LN - B}{NK} \quad (5)$$

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

$$\overline{Q} = \frac{B(N-1)}{NK} \quad (6)$$

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

$$\frac{d\overline{Q}}{dN} = \frac{BK}{(NK)^2} \quad (7)$$

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, $\bar{Q}$ will approach B/K. For an arbitrary number of sources the steady-state queue size will be between 0 and B/K. Equation 5 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.

III. Analysis

With the proportional compensator given by Eq. 2, we now have a complete model for the closed loop system. Now we use the tools of classical control theory to analyze this system. The loop gain, *G(z)*, of the closed loop system with this compensator is given by

$$G(z) = \frac{KN\Delta z^{-1}}{1 - z^{-1}} \qquad (8)$$

A common way to determine the stability of a closed loop system is by examining the Nyquist plot of this loop gain [5]. The Nyquist plot for this system is shown in Fig. 3. For this system to remain stable the Nyquist plot must not encircle the -1 point on the real axis. This is equivalent to the condition that

$$KN\Delta < 2 \qquad (9)$$

Also, if $KN\Delta$ is too close to 2 then the Nyquist plot will come close to -1 and the closed-loop system will experience oscillations, which will be increasingly less damped as $KN\Delta$ approaches 2.

To this point, we have not included delay in this model. There is a propagation delay between when the switch tells the source to transmit at a rate and when the switch actually sees this rate from the source. Also, depending on the method used to generate and queue RM cells there will be an additional delay between when the switch calculates a new feedback value and when it transmits this value in an RM cell. Using control theory, we can find the maximum delay for which the linearized system will remain stable. This maximum delay includes both the delay terms mentioned above.

The maximum tolerable delay of the closed loop system can be found from the phase margin and the crossover frequency of the system. Figure 4 shows a Bode plot of the loop gain, $G(z)$, for the case where $KN\Delta = 1$. The phase margin and the crossover frequency are indicated on this figure. If we let ω_C be the normalized crossover frequency, then for a given sampling time Δ, the crossover frequency is $\frac{\omega_C}{\Delta}$, and the maximum delay that can be tolerated is given by

$$\text{max delay} = \left(\frac{PM(rads)}{\omega_C}\right)\Delta \qquad (10)$$

where PM(rads) is the phase margin in radians. For the loop gain in Fig. 4, the phase margin is about 60 degrees or 1 radian and ω_C is also about 1 radian, so 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 also reduces the required overhead, but lengthens the time it takes for the network to respond to a change in traffic conditions. Also, increasing Δ and keeping $KN\Delta$ constant, means that K is decreasing, which will increase the steady-state queue size as shown in Eq. 6. Thus a trade-off exists between choosing Δ large to improve the robustness to delays and to minimize the control overhead and choosing Δ small to improve the response time of the system and to reduce the steady-state queue size.

The robustness to delays can also be improved by lowering the loop gain, and thus increasing the phase mar-

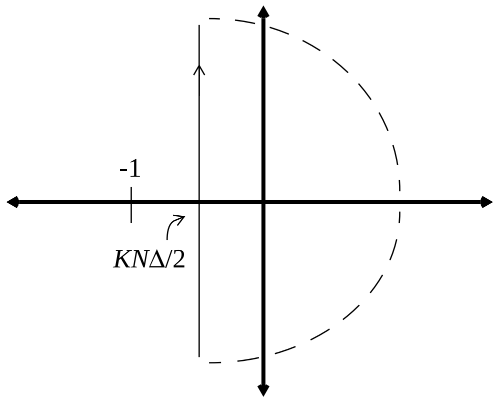


Figure 3: Nyquist plot of loop gain.

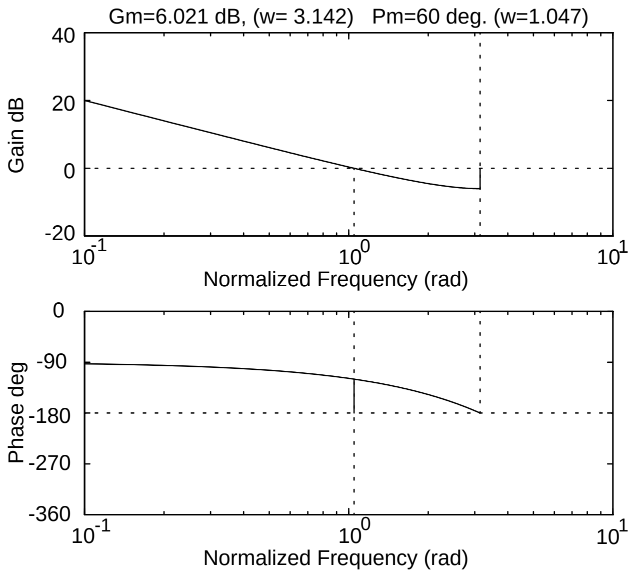


Figure 4: Bode plot of the loop gain with $KN\Delta$ =1.

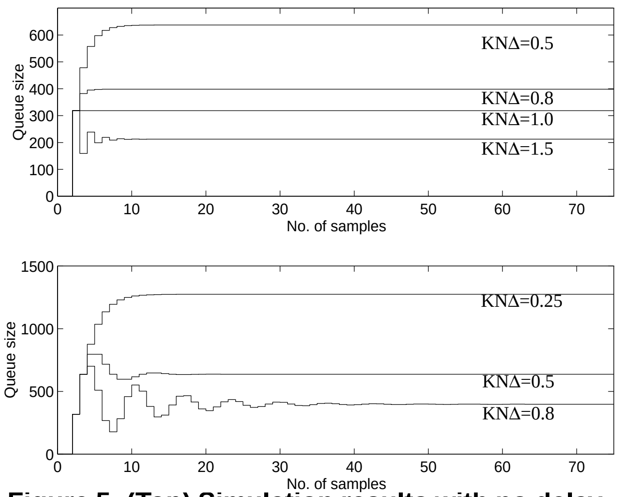


Figure 5: (Top) Simulation results with no delay. (Bottom) Simulation results with added delay of Δ seconds.

gin and decreasing the crossover frequency. This has some of 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 as shown in Eq. 6.

As an example of these trade-offs, if we choose $KN\Delta$=1 (as in Fig. 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 5 shows the simulation results of this algorithm for various values of the loop gain. The top simulations are for the network in Fig. 1 with no delay in the feedback loop. The abscissa points are samples placed Δ seconds apart. All the *N* sources start transmitting at an initial rate of *B* and eventually are brought down to their steady state values of *B*/*N*. 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.

The bottom of Fig. 5 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.

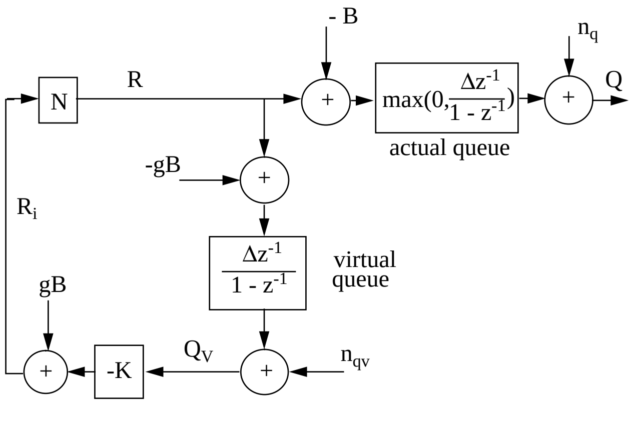
As an example we consider the 2 source/ 1 switch network whose simulation results are shown in Fig. 5. If the network normally operates with negligible delay, but the worst possible delay is 2.6Δ seconds, then we must choose $KN\Delta$<0.5 to assure stability. From Fig. 5, under normal conditions the transient takes about 10Δ sec. From Eq. 4.5, the steady state queue size is approximately 2*B*Δ. Another possible design for this system would be to increase Δ. If we choose Δ'=2.6Δ, 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 2Δ' sec. or 5.2Δ sec, but the steady state queue size is now 2.6(*B*Δ). 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 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.

A possible solution to this problem is to periodically change *K* based on an estimate of *N* to keep $KN\Delta$ constant. A rate control algorithm proposed by Jain [4] also requires that the switch estimate the number of active sources. In this 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 [1]. 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.

To achieve more robustness to delays and to accommodate more users both result in a larger steady-state queue size. This in turn can increase a users end-to-end delay and require more memory at the switch. Two methods of avoiding this growth in the steady-state queue size are considered. The first method is to consider the above algorithm 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 Fig. 6. 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 queue. The actual queue's size can also be found explicitly by analyzing the model in Fig. 6, but this requires taking into account the threshold non-linearity of the actual queue. Using the virtual queue



- B
n_q
R
N
+
$\max(0, \frac{\Delta z^{-1}}{1 - z^{-1}})$
+
Q
actual queue
-gB
+
R_i
$\frac{\Delta z^{-1}}{1 - z^{-1}}$
virtual queue
gB
Q_V
n_{qv}
+
-K
+

Figure 6: Closed loop system using a 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. Many of the algorithms presented to the ATM Forum also set their target bandwidths slightly less than the actual bandwidth to keep the queue lengths small in this same manner [3], [4].

A second method for keeping the steady-state queue lengths small is to design a different controller, $H(z)$, which will accomplish this. 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. A standard compensator that accomplishes this is a proportional-integrator (PI) or lag compensator. Such a compensator has the form:

$$H(z) = \left(\frac{-2K}{1+a}\right)\frac{(z-a)}{(z-1)} \tag{11}$$

This can be thought of a a proportional term, K, in series with a lag term, which has a unity high frequency gain. A standard design procedure for such a compensator [5] is to first design the proportional compensator, K, to achieve adequate robustness to delays as above. Then, if the zero, a, of the lag compensator is properly chosen, the resulting system will have about the same robustness to delays as the proportional compensator, but the steady-state queue size will now go to zero.

As an example of this consider the proportional compensator with $KN\Delta = 0.8$. This compensator has a crossover frequency of 0.82 radians and a phase margin of 66 degrees. For $\Delta = 9\text{x}10^{-4}$ seconds, it is robust to delays of up to 1.73 msec. Adding a lag compensator with a=.92 results a system with a robustness to delays up to 1.3 msec, nearly the same as for the system with the proportional compensator.

Figure 7 shows the transient performance for this system, given an initial rate of $3.54\text{x}10^5$ cells/sec from each of 2 sources which are competing for the output port of a switch with a bandwidth of $3.54\text{x}10^5$ 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. This is accomplished by initially driving the rates of the sources below their steady state value, so that the queue empties out. Thus, as with the virtual queue, some throughput is being sacrificed for a low steady-state queue, but with the lag compensator this throughput is only sacrificed when the queue needs to be emptied out.

One difficulty with using a lag compensator must be addressed. Since we are driving the steady-state queue to zero, the non-linear threshold in the queue (Eq. 1) will be active. This has the effect that if the queue is empty, and

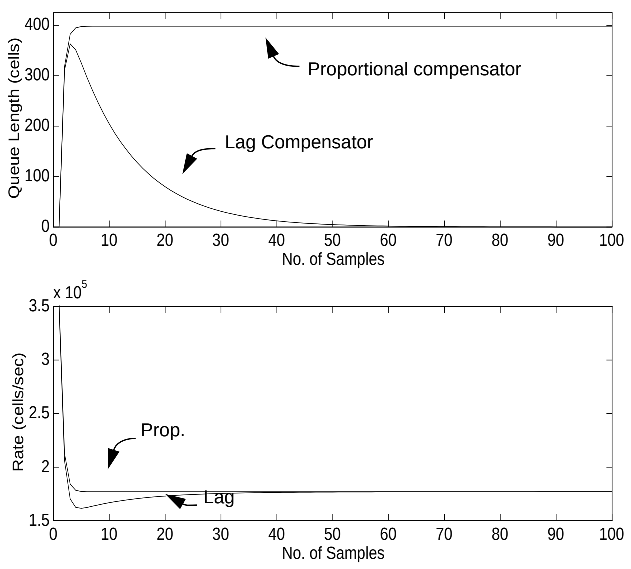


Figure 7: Transient response of system with proportional and lag compensators .

the incoming rates are less than the available bandwidth, then the lag compensator will not increase the rates. To increase the rates in this case, we must use another algorithm. Any algorithm which increases the rates to their steady-state values will suffice, *e.g.* telling all sources to additively increase their rates whenever the queue size is small.

IV. Extensions

Next we consider applying similar linear control approaches to more complicated networks than that in Fig. 1. There are two approaches to feedback control in larger networks. One is to consider the feedback loop being end-to-end, and the second is to consider the loop being closed over each hop, with the switches acting as 'virtual sources' and 'virtual destinations'.

In the end-to-end approach each switch would calculate a rate to feedback to the sources. RM cells would circulated through the network containing the allowed rate for a source. A switch would lower the allowed rate for a source if the allowed rate was greater than the desired rate of the switch. In this way each source essentially receives the feedback from the most constrained switch in its path. A difficulty with this approach, is that if a switch, S_1, has a queue built up, then it will output cells at its full bandwidth. Thus if a source slows down in response to a switch, S_2, which is after S_1, then S_2 will not see the true effect of the source slowing down until the queue in S_1 is empty. This added delay can make the system difficult to analyze. This also provides another motivation for using either the virtual queue or the lag compensator to drive the queues to zero.

Using a hop-by-hop approach avoids the above problem. In the above example, S_1 would slow the rate at which it was sending cells to S_2. This would cause the queue in S_1 to increase, and thus S_1 would tell the sources sending data to it to slow down. In the model in Fig. 2 this effect can be captured by considering the bandwidth, B, of a switch to be the input rate to the next switch. Thus B changes in response to feedback from the second switch. Such a system seems more promising to analyze than a end-to-end system, but implementing this system requires much more complicated switches.

V. Conclusion

In this paper linear control theory has been used to design and analyze explicit rate congestion control algorithms for a single hop network with greedy sources. Fundamental trade-offs such as between the response speed of the system and the robustness to delays were examined. Methods of driving the steady-state queue to zero were also discussed.

Only a simple network was analyzed in detail. Several approaches to larger networks were also presented along with some of the difficulties inherent in them.

References:

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

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

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

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

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

[6] Rohrs, C., R. Berry and S. O'Halek. "A Control Engineer's Look at ATM Congestion Avoidance", *Computer Communications* 19 (1996) 226-234.