Rate Based Congestion Control for Multicast ABR Traffic

Dirceu CAVENDISH[1], Saverio MASCOLO[2], Mario GERLA[1]

1. **Department of Computer Science, University of California,Los Angeles CA 90024, USA.**

   Phone: (310) 825-4367. Fax: (310) 825-7578. E-mail: gerla@cs.ucla.edu

2. Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, 70125 Bari - Italy

Abstract

In this paper we present an end to end rate control algorithm for multicast (point to multipoint) ABR service in ATM networks. The algorithm is the multicast extension of the SP-EPRCA unicast (point-to-point) congestion control algorithm proposed in [1]. The goal is to control the multicast input rate in order to achieve high bandwidth utilization whithout overflowing any queue along the multicast tree. The straightforward approach would be to consider a multicast connection as a superposition of unicast connections, and to compute the multicast input rate as the minimum among all unicast input rates. The main problem is the congestion caused by feedback traffic from the multiple destinations. To solve this problem, we show how feedback information cells can be intelligently "merged" at each multicast tree "fork" without losing essential information needed by the control algorithm placed at the source. The algorithm proposed inherits all properties from the unicast SP-EPRCA.

[Technical Subject Categories]

- ATM Systems and Networks
- Routing and Control of Congestion, Admission, and Flow

[Keywords]

ATM Congestion Control, End to End Flow Control, Large Propagation Delay, Multicast Traffic

Mailing Address:

All the future correspondence should be addressed to **Mario Gerla** at the above address.

1 Introduction

Congestion control is known to be one of the main challenges of wide area high speed networks, such as ATM, which are characterized by high bandwidth-delay products. Congestion is either avoided by preallocating bandwidth (CBR and VBR services), or is controlled by regulating input traffic rate at the source (ABR and UBR). In this paper, we address only rate based congestion control.

As recommended by ATM Forum, the rate based congestion control framework for ABR/VBR traffic follows a closed-loop control scheme, where the input rates are adjusted based on current network congestion levels [5]. The congestion information is collected by RM (resource management) cells, and transported all the way back to the source, to perform rate adjustement. The challenge is, however, that, the large delay inside the feedback loop due to the large bandwidth-delay product, affects stability. Fitting into this closed-loop framework, many algorithms have been recently proposed. In particular, we have proposed the Smith Predictor Enhanced Proportional Rate Control Algorithm (SP-EPRCA) [1], which shows how to take into account the feedback loop delays when input rates are computed. The main advantage of SP-EPRCA over other rate based schemes is to strictly prevent cell loss when adequate buffers are allocated to each connection at each intermediate node.

Cell loss prevention is very desirable in reliable multicast applications which require detection and recovery of lost data (e.g., distributed simulation). Note that while unicast is generally protected by TCP, multicast runs under UDP (which does not provide retransmissions of lost packets). Thus, in ATM, the loss of a multicast cell must be recovered at the application layer, with higher latency and processing overhead than at the transport layer [2]. If the cell loss is drastically reduced or better yet, eliminated (by the SP-EPRCA control algorithm, for example), then the payoff performance vs cost is substantial.

For multicast ABR services, it is important to investigate the additional difficulty introduced by multicast rate controlling. Very few researchers have addressed the problem so far. A significant contribution is by *Kay et al.* [4], which shows how it is possible to extend a max-min fair unicast explicit rate indication algorithm to multicast traffic.

This paper extends the SP-EPRCA algorithm to multicast traffic. The paper is organized as follows: We first consider a multicast connection as a collection of independently rate controlled unicast connections and define an ideal multicast rate control algorithm. Then we propose a way of "merging" RM cells so that the feedback traffic be reduced back to the forward control traffic levels. The cell loss free and min-max fairness properties come primarily from the unicast SP-EPRCA algorithm. Simulations are used to substantiate our claims. In the Appendix we provide an abridged description of the unicast SP-EPRCA [1].

2 Ideal Multicast Rate Control Algorithm

In this section we first define the ideal multicast rate control algorithm(IMRCA).

Definition 1 *The Ideal multicast rate control algorithm (IMRCA) sets the rate of a multicast connection* $(S, D^1, D^2, \cdots, D^M)$ *to the minimum of the rates computed by* M *unicast SP-EPRCA controllers executed independently for each pair* (S, D^i)*, where each algorithm is fed by its corresponding RM cell stream. Before the multicast rate updating, the source must receive one RM cell from each multicast destination. In case any of RM cells is missing for a time interval longer than* $\triangle$*, the algorithm performs virtual feedback (see Appendix, or [1]) across individual unicast SP-EPRCAs in order to update the multicast input rate as the minimum among them.*

In other words, each unicast controller placed at source site computes its rate u_i according to equation 7. Since, by definition of multicast, the input rate is unique across the entire multicast tree, each unicast controller differs from the others because of the round trip delay (RTD). When a unicast rate (u_j) is computed, say at time t, the multicast rate is then updated by:

$$u_{min}(t) = \min_{1 \leq j \leq M} (u_j) \tag{1}$$

The straightforward implementation of the ideal multicast algorithm is as follows. Data and RM cells are generated at source site (at a proportion of NRM data per RM cell [1]), and travel downstream along the multicast tree, getting replicated at fork nodes (see Fig. 1). At each fork of the multicast tree, RM cells are stamped with their respective destinations IDs when replicated so that each RM stream can be identified at the source site. In the backward direction, RM cells are sent unaltered upstream to the respective source. I. e. M SP-EPRCA algorithms are running independently, each one fed by its own RM cell stream.

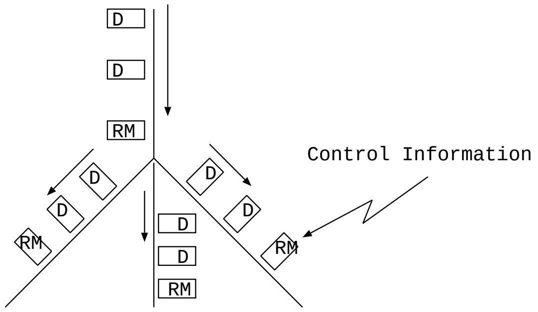


Figure 1: Splitting at multicast tree fork

Although this algorithm is clearly ideal (IMRCA), it is not feaseble, because the control traffic that is generated downstream the tree must return upstream, most likely causing congestion. As an additional drawback, the computation burden at the source grows with the number of destinations. In the next section we show how to solve this problem.

3 Merging RM cells

The previous section has exposed the infeasability of sending multiple control streams back to the source for rate control. Clearly the main issue here is how to "merge" RM cells coming back at fork nodes so that useful feedback information is *not* lost. In essence, merging is the inverse process of splitting. If a forward stream has been splitted in K streams at fork F, then the K control feedback streams are merged into one stream at F. Therefore, the merging process requires the fork F to wait for one RM cell from each of its "children" before deciding what to send back in a single RM cell. Figure 2 shows the RM merging at a tree fork.

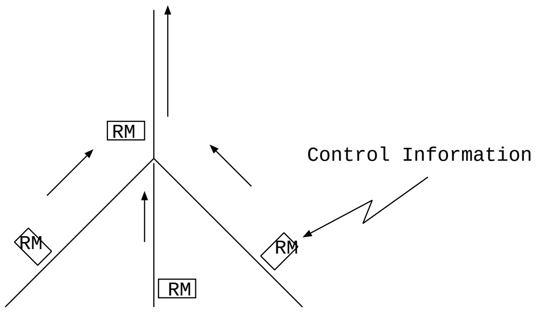


Figure 2: Merging control traffic at multicast tree fork

Suppose all K RM cells were able to reach the source, so that the source runs independent unicast SP-EPRCA controllers, one per source/destination path. Let U be the vector of rates $U = (u_1, u_2, \cdots, u_M)$ computed. It is clear that, using IMRCA, only one SP-EPRCA controller will dictate the next rate to be enforced. Therefore, the idea is to detect, at each fork point, which RM cell information will result in the lowest rate, and to pass only that value upstream.

3.1 How to choose the right RM cell information

Considering a generic fork F, let td_f and td_i be the transmission delays from F to the source and from F to the destination D^i, respectively. Assume that at time t the source gets one RM_i cell from each D^i. Then the unicast rate u_i will be computed as [1]:

$$u_i = K[X^o - x_i - \int_{t-RTD}^{t} u(\tau)d\tau] = K[X^o - N_i]^1 \tag{2}$$

where $N_i = x_i + \int_{t-RTD}^{t} u(\tau)d\tau$.

A physical interpretation for equation 2 is that the new rate is proportional to the available space at the bottleneck buffer (which has size X^o). Available space is X^o minus the queue level information received in the RM cell, minus all the in flight cells in the feedback loop, i. e. N_i, which might get stored in this buffer.

The minimum rate is, thus, given by the jth controller (corresponding to destination D^j) such that:

$$x_j + \int_{t-RTD_j}^{t} u(\tau)d\tau = \max_{1\le i\le M}[x_i + \int_{t-RTD_i}^{t} u(\tau)d\tau] \tag{3}$$

It remains to be shown how fork F can decide which of its children will provide the feedback information that satisfies Eq. 3 at the time $t' = t - td_f$

We can write N_i as the :

$$N_i = x_i + \int_{t-RTD_i}^{t-td_f} u(\tau)d\tau + \int_{t-td_f}^{t} u(\tau)d\tau = x_i + c_i + D \tag{4}$$

Since D is constant for all i, the j that maximizes N_i is given by the j that maximizes $(x_i + c_i)$.

Therefore, at time t', fork F selects the jth feedback value such that:

$$x_j + c_j = \max_i(x_i + c_i) \tag{5}$$

[1] We use x_i instead of $x_i(t - t_{fb})$ for sake of conciseness

This computation is performed sequentially from the destinations (multicast tree leaves) up to the source.

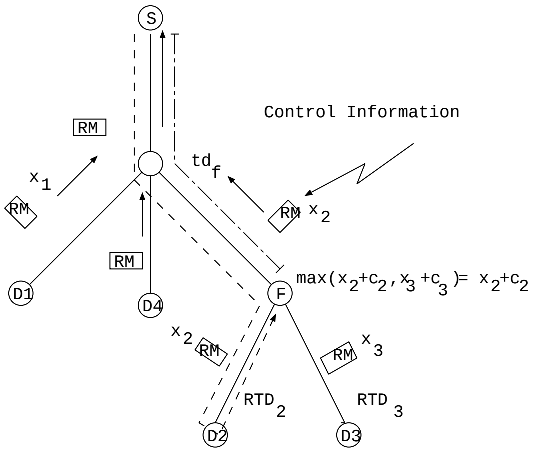


Figure 3: Feedback Information Selection

3.2 The SP Multicast Rate Control Algorithm (SP-MRCA)

The fork nodes in the multicast tree perform the merging scheme described previously, as long as they receive RM values from all their children in the timely fashion. Since at multicast setup time no information is present in the tree, the algorithm does not receive feedback information until RTD_{max} has elapsed, where $RTD_{max} = \max_{1 \leq i \leq M}(RTD_i)$ is the maximum round trip delay among all D^i belonging to the multicast connection. Therefore, we need to specify the algorithm behavior for two cases:

i) $t < RTD_{max}$: Since the algorithm does not receive feedback cells, it uses the virtual feedback of the unicast SP-EPRCA algorithm with RTD_{max}[2].

ii) $t \geq RTD_{max}$:

1- Upon receiving RM cells, the algorithm updates the rate using the unicast SP-EPRCA controller corresponding to the destination D_i from which the cell comes from.

2- Upon expiration of the feedback interrarival time $\triangle$, the source uses the last RM cell received and the corresponding unicast SP-EPRCA algorithm to perform a virtual feedback[1].

[2] We assume a RTD_{max} large enough such that virtual feedback is needed, since the other case is trivial (see Appendix or [1])

Theorem 1 *The SP-MRCA algorithm is equivalent to the IMRCA algorithm.*

We have to show that the two algorithms compute always the same rate. Since the proof is rather intuitive, for sake of brevity we give only a sketch of it.

Sketch of Proof 1 *We divide the proof in two cases:*

i) $t < RTD_{max}$*: During this interval, the connection with longest RTD does not receive feedback, and thus uses virtual feedback for its rate computation, say* u_v*. The rate* u_v *results to be the minimum among all independent SP-EPRCA controllers, and hence is equal to the rate computed by the IMRCA.*

ii) $t \geq RTD_{max}$*:*

1- Upon receiving RM cells, our merging scheme at the fork points sends upstreams the RM cell corresponding to the minimum rate. Thus the theorem follows in this case.

2- Upon expiration of the feedback interarrival time $\triangle$*, the IMRCA uses virtual feedback, that is, it increases each* $x_i + c_i$ *by the same amount (since the multicast rate is unique). Therefore, the minimum rate decreases, but it is computed by the same* j *controller as in the previous iteration, as in the MRCA.*

3.3 A pratical way for forks to filter out excessive RM cells

In merging feedback information, cells must be filtered out at fork points. In the SP-MRCA algorithm, a fork must have feedback information about all its children before relaying any information up the tree. In principle, feedback information from shorter paths must wait for that coming from the longest path. This suggests the following scheme for RM cell filtering and information reporting back to the source:

Define the longest path (F, D_j) among all D_i from a fork node F to be a kind of "conveyor belt" of feedback information. That is, RM cells circulating in this path are used to relay feedback information upstream after merging. Namely, all cells coming from a branch other than the conveyor belt are filtered out after recording their feedback information. The merged information then waits for the next RM cell coming from the conveyor belt to be carried back to the source. Each fork has its own conveyor belt, which is determined by its distance to its destinations (sub tree leaves rooted at F). The scheme is shown in Figure 4.

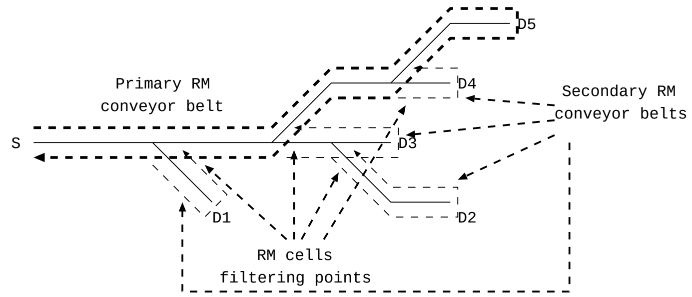


Figure 4: Recursive filtering of RM cells

4 Simulation Results

The network topology used in a simulation of the SP-MRCA is shown in Fig. 5. Links have uniform speeds, normalized to 1 cell per unit of time [cell/s]. Link labels represent the bandwidth-delay product. There are five unicast connections: $(S1, D1), (S2, D2), (S3, D3), (S4, D4), (S6, D6)$. There is also one multicast connection $(S5, D5, D7, D8)$, spanning across multiple bottlenecks.

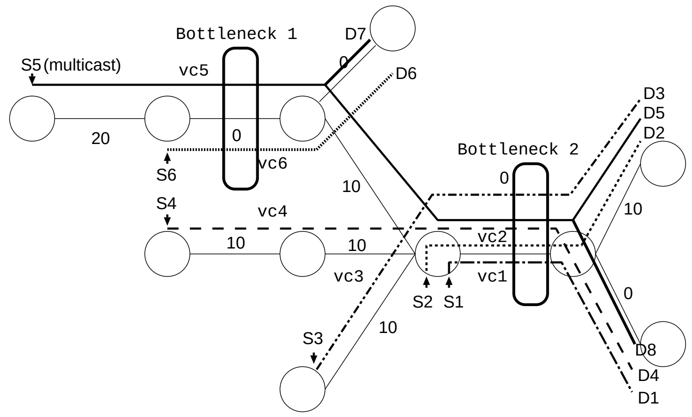


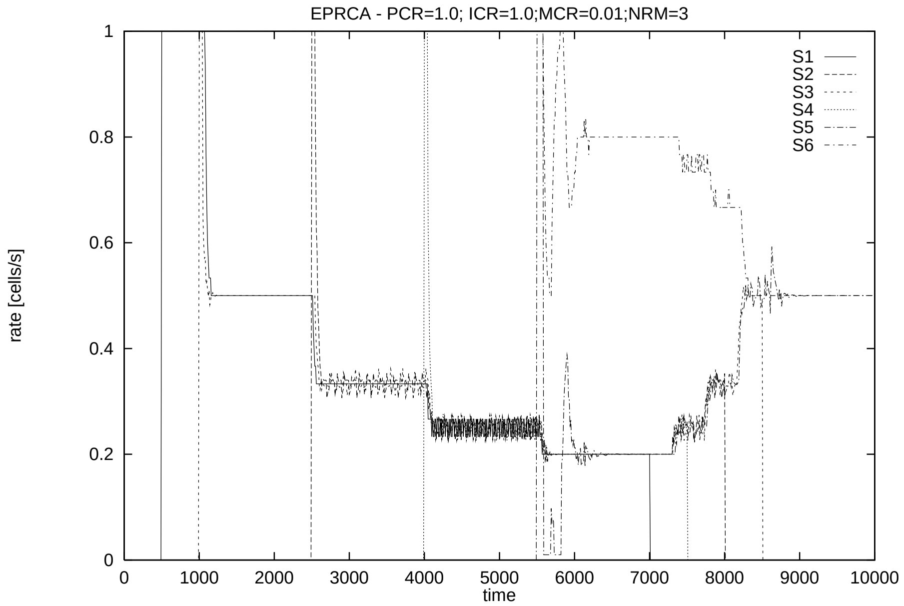
Figure 5: Network Topology

Connections are set and released according to Table 1. This time epochs were chosen so that bottlenecks shift in time from link to link across the network. We assume infinite backlog at each source. Buffer capacity is 60 cells.

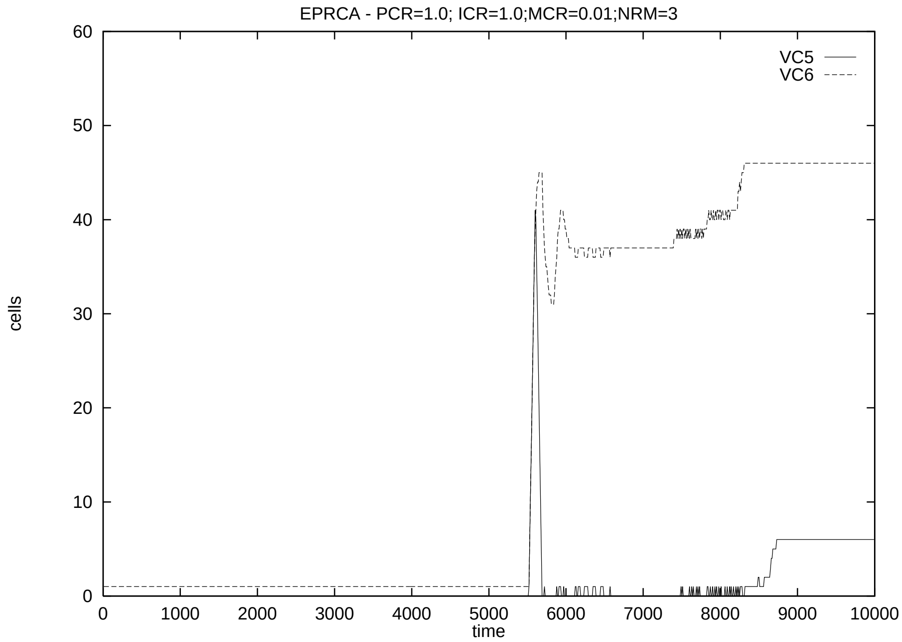
Table 1: VC Connections Activity

Connection #	1	2	3	4	5m	6
Start Time	500	2500	1000	4000	5500	0
End Time	7000	8000	8500	7500	10000	10000
RTD (cells)	0	20	40	40	80(max)	0

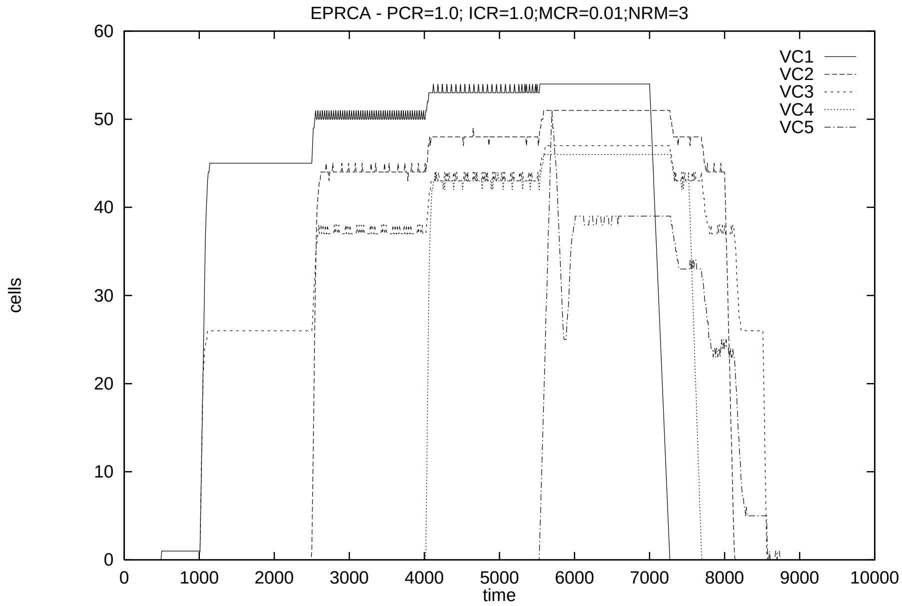
Figure 6(a) shows the behavior of the six input rates, corresponding to connections $S1 - S6$, at source nodes. We assume an initial cell rate of 1 [cells/s], although the results are independent of this value. After each traffic pattern change (start/end of a connection), each rate rapidely settles to a new fair stationary value. Sources $S1$, $S2$, $S3$, $S4$, and $S5$ (multicast) are constrained by bottleneck #2, and thus get the same bandwidth share each, while source $S6$ is constrained by bottleneck #1, and therefore takes all the bandwidth left by the multicast source $S5$. We can see, also, that due to traffic variations during simulation time, the multicast bottleneck moves along the tree (i.e., from bottleneck 2 to bottleneck 1, see Figure 5). Figure 6(b) shows the dynamic behavior of the two queues at bottleneck link #1, corresponding to $VC5, VC6$, while Fig. 6(c) shows the dynamic behavior of the five queues at bottleneck link #2, corresponding to $VC1 - VC5$ bottleneck queues (since connection #6 does not cross this bottleneck). No queue overflow occurs, by virtue of the SP-EPRCA overflow prevention property. Figure 6(b), for time ¿ 8500, shows all queues emptying due to traffic termination, which again means that the bottleneck queue for multicast source $S5$ has moved from bottleneck #2 to bottleneck #1 (Fig. 6(b)). The overall performance is consistent with the ideal multicast rate control algorithm (IMRCA), as expected.



(a) Rates



(b) VC Queues at Bottleneck #1



(c) VC Queues at Bottleneck #2

Figure 6: Multicast Rate Control Algorithm

5 Conclusion

We have proposed an efficient multicast rate control algorithm (SP-MRCA), which extends the unicast SP-EPRCA algorithm to multicast traffic. Key issues, such as the reduction of excessive feedback information, were addressed. Simulation results were used to exemplify the algorithm performance with respect to dynamic tracking of bottlenecks, cell loss free property and fairness. The trade-offs between buffer sizes and throughput are the same as in the SP-EPRCA, and thus were not addressed here. Although we have assumed individual queues per VCs at each switch output port, work is in progress to provide single buffer per output port, as reported in [3] for unicast connections. As for the implementation of the Smith Predictor controller at the source, this can be placed either in the Host or in the edge switch. The latter implementation is preferred since it complies with the ATM Forum UNI specifications. In this case, the entire SP-MRCA is transparent to the end user; the edge switch simply dictates the allowed rate to the Host.

It is worth saying that the algorithm can be easily extended to support dynamic join/leave of receivers in the multicast group. Namely, if the multicast VC tree can be dynamically expanded/contracted, then the SP-MRCA algorithm will automatically adjust to the new configuration. Likewise, the SP-MRCA can also be extended to the many-to-many multicast case.

Strict buffer overflow protection via SP-MRCA may require large buffers for WAN type propagation delays in order to have full bandwidth utilization. To reduce implementation cost, without giving up bandwidth utilization, smaller buffers than the strictly required for large delays may be used, thus incurring some cell loss. In this case, Host retransmissions using schemes such as the one reported in [2] must be used. Work is in progress to determine the trade-offs between buffer costs and Host retransmission overhead and latency.

References

[1] S. Mascolo, D. Cavendish, M. Gerla,"ATM Rate Based Congestion Control Using a Smith Predictor: an EPRCA Implementation", *To appear at INFOCOM'96.*

[2] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *Proceedings of SIGCOMM'95*,pp.342-356.

[3] D. Cavendish, M. Gerla, and S. Mascolo "ATM Rate Based Congestion Control Using a Smith Predictor: Implementation Issues", *Proceedings of First Workshop on ATM Trafic Management, IFIP*,pp. 289-296.

[4] K. Y. Siu and H. Y. Tzeng, "Congestion Control for Multicast Service in ATM Networks", *Proc. IEEE GLOBECOM'95*,pp. 310-314.

[5] A. W. Barnhart, "Baseline Performance Using PRCA Rate-Control," *ATM Forum/94-0597*, July 1994.

[6] O.J.Smith, "A controller to overcome dead time," *ISA J., vol.6, no.2*, Feb. 1959,pp.28-33.

Appendix: Unicast SP-EPRCA Algorithm

SP-EPRCA is a closed loop rate based congestion control algorithm in which source rates are adjusted according to VC queue levels. Its novelty is that it uses a Smith Predictor [6] to overcome the large bandwidth x delay product characteristic of high speed wide area networks. Moreover, queue levels are used rather than rates as feedback information, relieving the switches of any real time computation, since queue length measurements are readily available.

In what follows, we summarize the work done in [1]. The interested reader is referred to that paper for more formal and detailed analysis of the algorithm.

The continuous time model of the system to be controlled can be modelled as shown in Fig. 7. There we represent the bottleneck queue of a VC by an integrator (fluid flow model), the proportional controller gain by K, and the delays T_{fw} and T_{fb} inside the control loop. The variables of interest are:

$u(t)$	-	Source input rate
$x(t)$	-	Bottleneck buffer level
$d(t)$	-	Bottleneck service rate
X^o	-	Target filling level of switch queues
K	-	Gain of the proportional controller
T_{fw}	-	Feedforward delay
T_{fb}	-	Feedback delay

The main problem to control the system is the delays inside the feedback loop. Thus we pull the delays out of the feedback loop, which changes the system behavior from infinity order to first order, using the well established technique in control theory of time delayed system, called Smith Principle [6].

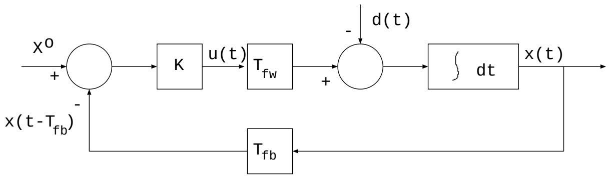


Figure 7: Queue dynamic model of a proportional controlled source

Namely, by using the Smith Principle, we design a controller K^* (the Smith Predictor) so that the system has dynamics similar to a first order system with a delay in cascade, which is shown in Fig. 8.

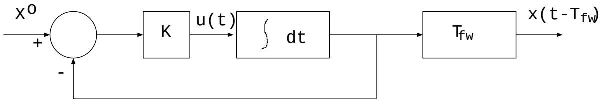


Figure 8: Equivalent model of the system using a Smith Predictor

The controller sought is shown in Fig. 9. The gain K is a proportional gain, which drives the control system responsiveness. The resultant system is shown in Fig. 10. It is easy to prove that the systems of Figures 8 and 10 have equivalent transfer functions.

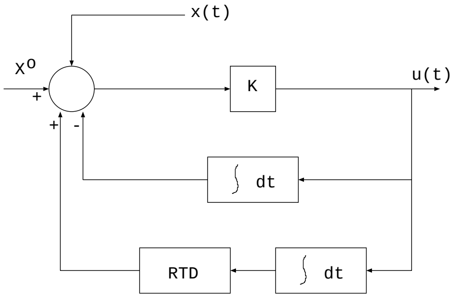


Figure 9: Smith Predictor

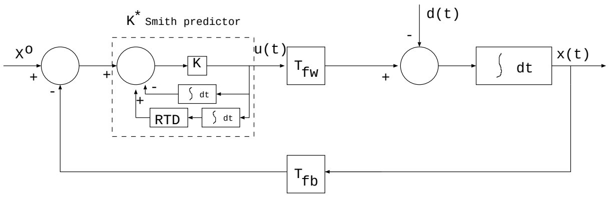


Figure 10: Queue dynamic model using a proportional controller plus a Simth Predictor

The resulting control equation is:

$$u(t) \;=\; K[X^o - x(t - T_{fb}) - \int_{t-RTD}^{t} u(t^{'})dt^{'}] \tag{6}$$

This equation can be interpreted as a proportional control of the queue level $N(t) = x(t) - \int_{t-RTD}^{t} u(t^{'})dt^{'}$, i.e. the in pipe cells can be seen as being stored at the bottleneck queue.

Mascolo at al[1] presents a discrete time version of this continuous time control equation, in order to provide the effective equation used by the controller. Requirements about sampling frequency arises, which lead to the creation of safety procedures to ensure loss free property in case of missing feedback information.

Discretizing the control equation 6, it turns out:

i) $RTD \geq \triangle$:

$$u(t_k) = u(k\triangle) \;=\; K[X^o - x(k\triangle - T_{fb}) - u(k\triangle - (m+1)\triangle)\epsilon\triangle - \sum_{1}^{m} u((k-i)\triangle)\triangle] \tag{7}$$

ii) $RTD < \triangle$:

$$u(t_k) = u(k\triangle) \;=\; K[X^o - x(k\triangle - T_{fb}) - u((k-1)\triangle)RTD] \tag{8}$$

where u is the controlled rate, K is the proportional gain, X^o is the queue threshold,$\triangle$ is the sampling period, and $RTD = T_{fw} + T_{fb}$ is the round trip delay. Figure 11 shows the relation among the parameters and notation adopted.

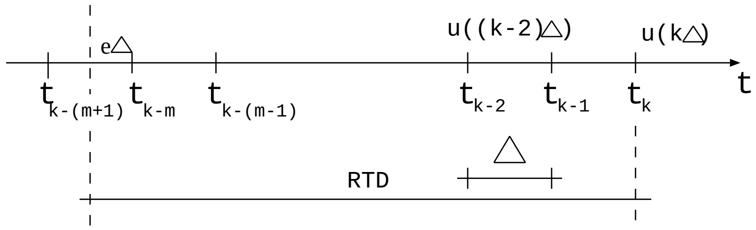


Figure 11: Discrete Time Notation

In case of missing (either lost or late) feedback information, a "worst case" estimate of feedback queue level is performed. The queue estimate is, thus, given by:

$$x(t_k + \triangle_k - T_{fb}) \ = x(t_k - T_{fb}) + \int_{t_k+\triangle_k-RTD-\triangle_k}^{t_k+\triangle_k-RTD} u(\tau)d\tau \tag{9}$$

and its discrete time version:

$$x(t_k + \triangle_k - T_{fb}) = x(t_k - T_{fb}) + u(t_{k-m-2})(RTD - \sum_{i=1}^{m+1} \triangle_{k-i}) + u(t_{k-m-1})(\triangle - (RTD - \sum_{i=1}^{m+1} \triangle_{k-i})) \tag{10}$$

where $x(t_k - T_{fb})$ is the last feedback information received, $\triangle_k$ is the k sampling period, and the remaining subscripts are explained in Fig. 11. This means that the source consider zero bottleneck service rate since the last feedback information received. This worst case estimation is called "virtual feedback", and again the interest reader is referred to [1] for more detailed explanation.