Phantom: A Simple and Effective Flow Control Scheme *

Yehuda Afek Yishay Mansour Zvi Ostfeld
Computer Science Department,
Tel-Aviv University, Israel 69978.
{afek,mansour,ostfeld}@math.tau.ac.il

November 14, 1999

Abstract

This paper presents *Phantom*, a simple constant space algorithm for rate based flow control. As shown by our simulations, it converges fast to a fair rate allocation while generating a moderate queue length. While our approach can be easily implemented in ATM switches for managing ABR traffic, it is also suitable for flow control in TCP router based networks. Both the introduced overhead and the required modifications in TCP flow control systems are minimal. The implementation of this approach in TCP guarantees fairness and provides a unifying interconnection between TCP routers and ATM networks. The new algorithm easily inter-operates with current TCP flow control mechanisms and thus can be gradually introduced into installed based TCP networks.

1 Introduction

A major goal of network protocols is to maximize the utilization of the network resources (e.g., bandwidth) while sharing the resources in a fair way among the users/applications. Flow control is the mechanism that controls the flow of traffic into the network and whose goal is to avoid and resolve data traffic congestion while ensuring high utilization and fairness among the different connections. As the speed and size of computer networks increase the flow control is becoming more critical and challenging. In such networks a small mistake by the flow control for a tiny period of time may quickly result in clogging and in the loss of many messages.

Communication protocols in general and in large high speed networks in particular should have the following properties: simplicity, minimal space requirements (both buffers and algorithm variables), short response time to bursty traffic requirements, and inter-operability across several networks. The flow control mechanism is no exception; the flow control algorithm presented here has all the above essential features.

One property, fairness, is unique to flow control. While intuitively we may understand the notion of fairness, namely, fairly sharing the network bandwidth among different users, defining it rigorously in a network environment is not trivial. A widely acceptable measure of fairness is the

*A preliminary version of this paper was presented in Sigcomm 96

max-min fairness [BG87, Jaf81, GB84, Gaf82, Hay81, Mos84, Cha94]. A bandwidth allocation is max-min fair if for every session, one cannot increase its bandwidth without decreasing the bandwidth of sessions of equal or lower bandwidth [BG87].

This paper presents an efficient, fair and simple scheme for flow control, called *Phantom*, which is suitable for both TCP router based networks and ATM networks. Thus, not only suggesting a flow control mechanism for the two network types, but also making the first step in integrating the flow control of both types of networks.

The key idea behind the Phantom algorithm is to bound the rate of sessions that share a link by the amount of unused bandwidth on that link. As though there is another imaginary session, a *phantom* session, on the link, and the link bandwidth is fairly shared among all the sessions, including the phantom. Using the Phantom mechanism max-min fairness is achieved for a network in which on every link there is an extra "phantom" session.

In addition the Phantom scheme has the following advantages: (1) it requires a constant number of variables in every output port of any router/switch, (2) it quickly and robustly adapts to variations in the traffic of higher priority classes, (3) in certain variations of Phantom the forward RM cells are not necessary, and (4) the scheme may be implemented with simple hardware, using only basic arithmetics, without multiplications or divisions.

The idea of adding another imaginary session to every link is not new. Jaffe [Jaf81] was first to use the idea in order to study the convergence of flow control algorithms, however, that algorithm, as well as other related works along this lines [GB84, Hay81, Mos84], may not be effectively used in a distributed environment in which the traffic frequently changes. In [CH96] Choudhury and Hahne employ the same principle of "using the spare resources to control the resource consumption" as suggested by Jaffe and as we do. However, Choudhury and Hahne use it to control the buffer allocation to different sessions while we employ it on the bandwidth allocation. This method may produce in an indirect way a flow control mechanism. However, its performances as a flow control algorithm were not explored.

Several flow control mechanisms have been suggested in recent years for TCP router based networks [Jac88, BP95, Zha89, FJ93]. In [Jac88, BP95] the flow control operates at the source ends of the sessions by adjusting the window size according to the detection of packet loss or the round trip delay. Recent extensions of these algorithms produced the Selective Active Discard (SACK, see [FH99] for example).

Other mechanisms that operate at the routers (gateways) are Random Early Detection [Zha89, FJ93] and Early Packet Discard [RF95], which in case of congestion drop new incoming packets in an intelligent approach. Recent mechanisms for restricting selected traffic and hence improve the total bandwidth and fairness of the network are presented in [FH99].

Different ways of applying the Phantom scheme to flow control in TCP routers based networks are presented in Section 4. While the main advantage of these applications is in achieving fairness, it does require some modification of either the source end, or the routers, and the TCP packet format. However, it seems that fairness may not be achieved without some modifications either at the routers [Kes91] or at the TCP (or IP) header.

Applying the Phantom scheme to ATM networks is somewhat easier due to the handles provided by the ATM Forum. ABR (Available Bit Rate) is a class of service in ATM networks that allows bursty data applications to utilize the available network bandwidth in a *fair* way, while guaranteeing a low cell loss rate. The ATM Forum on Traffic Management has adopted rate based as the basic mechanism for flow control of ABR traffic. The basic principle of the rate based flow control is as follows[1]: A control message (Resource Management cell, RM cell) is looping around the virtual circuit of each session. On its forward way the RM cell carries the CCR (Current Cell Rate) of the session and "informs" the switches about the session's rate ACR (allowed Cell Rate), while on the backward direction, each switch may decrease the ER (Explicit Rate) field in the RM cell, according to the congestion that it observes. (Notice that switches may decrease the value of the ER field but may not increase it. Hence, when the RM cell arrives at the source it contains the minimum ER value for all switches on the path.) When the RM cell arrives back to the source, the value in the ER field is the maximum rate at which the source may transmit. However, the source may not increase its rate too fast, the amount of increase is limited by a parameter, AIR (Additive Increase Rate) per RM cell received [Sat96].

Since the ATM Forum has adopted the rate based scheme as a standard for flow control, [BF95], several proposals for its implementation in a switch have been proposed. One way of partitioning the different proposals is by their space requirements: (1) Flow control algorithms with constant space, i.e., the number of variables used by the algorithm is constant independent of the number of sessions passing through a port [Rob94, ST94, JKV94, Bar94, Bar95], and (2) unbounded space algorithms, i.e., algorithms in which the space depends (usually linearly) on the number of connections [CCJ95, KVR95, CR96, TW96, JKG$^+$95]. The Phantom algorithm presented here is a constant space algorithm. In Section 5 it is compared to three other constant space algorithms that have been presented in the ATM Forum. We believe it overcomes some of the weaknesses of these algorithms.

Constant space flow control algorithms are important for todays networks, e.g., if one would like to support both ATM and TCP traffic, then a dependence on the number of sessions is prohibitory.

2 Phantom - The Basic Scheme

The basic idea of the Phantom algorithm is to keep a certain portion of the link capacity unused and to limit the rates of sessions sharing the link by the amount of the *unused* bandwidth on that link (as if there is an extra phantom session on the link). Specifically, we define Δ to be the unused link capacity, i.e., $Link\ Capacity\ -\ \sum rates\ of\ sessions\ that\ use\ the\ link$. The rate of sessions that are above Δ is reduced towards Δ and the rate of sessions that are below may be increased. This mechanism reaches a steady state only when the unused capacity (Δ) is equal to the maximum rate of any session that crosses the link and all the sessions that are constrained by this link are at this rate. The value of Δ is easily computed in each output port by counting the number of cells (messages) arriving at the queue of that port over an interval of time and subtracting this amount from the number of cells that may be transmitted in that interval (this quantity is normalized by dividing it by the length of the time interval).

For example, if three sessions share a 100 Mbs link, then in steady state each session receives 25 Mbs and the link utilization is at 75% (in this case Δ is 25 Mbs too). However, if two of the three sessions are restricted elsewhere to 10 Mbs each, the third session gets 40 Mbs and Δ is also 40 Mbs.

[1]Following the DECbit [RJ90], the ATM flow control supports another mechanism using the CI (Congestion Indication) bit on control cells.

The basic Phantom scheme is very robust, as our simulations indicate. The robustness stems from the following observations:

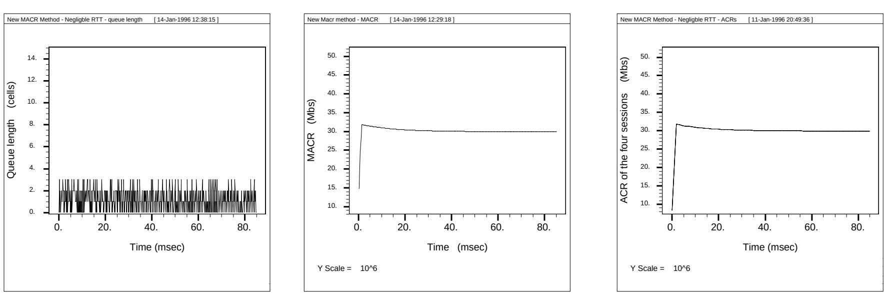
1. In a stable environment the scheme converges to a steady state in which fairness is achieved. This fact follows from the property that all the sessions that are constrained on some link get the same rate (which is the value of Δ for that link) in a stable state.

2. Since the Phantom scheme preserves a residual unused capacity, it smoothly accommodates the addition of a new session without a queue buildup.

3. The basic attitude of Phantom is to **avoid** congestion rather than to **react** upon it. For example, consider a 100 Mbs link shared by three sessions, two of which are constrained elsewhere to 10 Mbs, and one limited by this link to 40 Mbs. If the 10 Mbs restriction of the two sessions is removed at the same time then while the rate of the two sessions increases the rate of the third session decreases, even before a queue is built. This is because the value of Δ decreases as the rate of the two sessions increases above 10 Mbs. [2]

However, to employ the Phantom scheme in an actual network tuning and adjustments are necessary. Since the necessary mechanisms for its implementation in ATM are readily available we start by describing and analyzing it in an ATM environment. However, we believe the scheme is attractive and effective also in TCP networks and that the ATM analysis and results would basically carryover to the TCP settings. In Section 4 we discuss its implementation for flow control over TCP.

3 Implementation in ATM

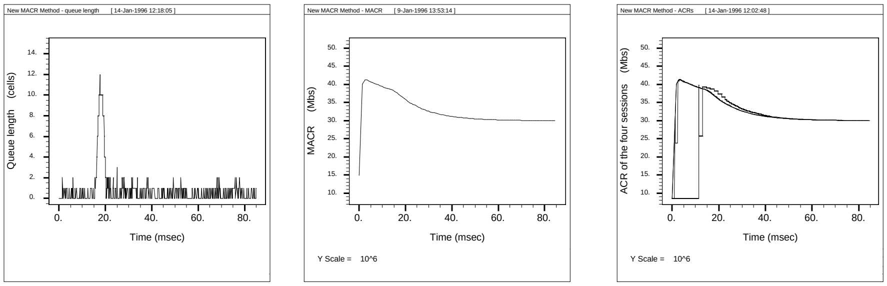


Queue length MACR Sessions' allowed rate

Figure 1: Phantom - Four greedy sessions with negligible RTT (0.01 ms) sharing a 150 Mbs link. Both $MACR$ and the sessions' rate converge to $\frac{150}{4+1} = 30Mbs$.

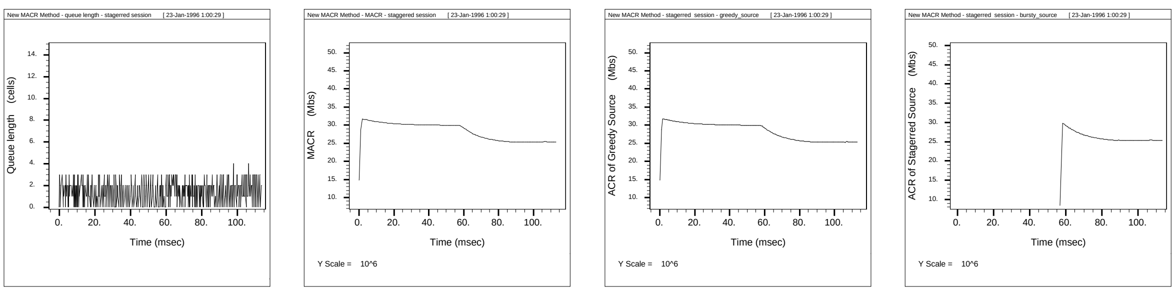
The following five issues, concerning the implementation of the algorithm in ATM, are addressed in the sequel:

[2]Assuming that the rate of the two sessions does not instantly jump to a very high value, that is, in ATM terms [Sat96], AIR (Additive Increase Rate) $\cdot Nrm$ is much smaller than 30 Mbs.



Queue length | MACR | Sessions' allowed rate

Figure 2: Phantom - Four greedy sessions as in Fig. 1 with different RTTs (11 ms, 1 ms, 0.1 ms and 0.01 ms) sharing a 150 Mbs link. Three sessions with a relatively small RTT get high rates at the beginning (they quickly receive the feedback from the switch) while the fourth session still uses its initial rate. However, it does not take long before the fourth session gets the feedback and converges to the desired value (which is $\frac{150}{4+1} = 30Mbs$). Note that a small queue (12 cells) is formed during convergence time and is drained shortly after.



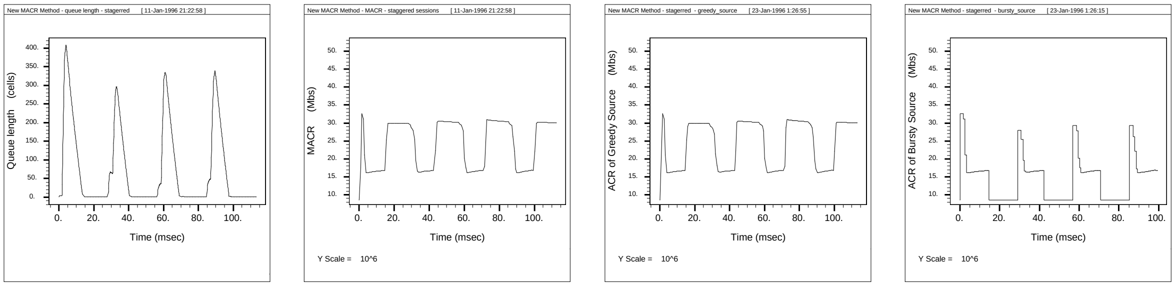
Queue length | MACR | ACR of a Greedy Source | ACR of a Stagerred Source

Figure 3: Phantom - Four greedy sessions sharing a 150 Mbs link; The allocation for each session is $\frac{150}{4+1} = 30Mbs$. A stagerred session is added after 58 ms. The addition does not cause a queue buildup and all the sessions' rate converge to a new value of $\frac{150}{5+1} = 25Mbs$.

1. **Measuring Δ:** A naive measurement of Δ may be very noisy and unstable due to the inherent delay in the system and to the nature of the traffic. Hence, a technique to filter out the variations and noise in the measurement is necessary. The filtered value of Δ is called $MACR$ (details are given in the sequel).

2. **Sensitivity to queue length:** Inaccuracies, rounding errors, and noise, may still introduce an error which might accumulate over time and exhibit itself in a gradual queue buildup. Hence, the operation of the algorithm has to be coupled with the absolute value of the queue length.

3. **Utilization:** If only a few "heavy" sessions use the network then the scheme might underutilize the available bandwidth. Below we introduce a slight modification to ensure high utilization also in these cases.



Queue length | MACR | ACR of a Greedy Source | ACR of a Bursty Source

Figure 4: Phantom - Four greedy sessions and four on/off sessions sharing a 150 Mbs link. The four on/off sessions are on for a period of 14 ms and then become idle for a period of 14 ms. $MACR$ is either $\frac{150}{4+1} = 30Mbs$ when there are 4 active sessions or $\frac{150}{8+1} = 16.7Mbs$ when there are 8 active sessions. Note that a queue is built when the on/off sessions become active and is drained out afterwards. When the on/off sessions become idle adjustment is achieved without queue buildup. Note also that the ACR of the on/off sessions is set to be their Initial Cell Rate (ICR), 8.5 Mbs when they are idle. However, during that period no data is transmitted by those sessions.

4. **Variance Consideration:** In some circumstances a minor change in the value of $MACR$ may cause a large surge in the incoming traffic. This in turn may result in large oscillations in $MACR$ even if the sources are greedy (steady). We introduce a method that reduces these oscillations, while still maintaining the fast response to changes in the sources' traffic.

5. **Reducing the maximum queue length:** Under certain scenarios, e.g., many synchronized sessions starting to transmit at the same time, a large queue may momentarily develop. Here we present a method that allows us to tune the tradeoff between the size of such a spike in the queue length and the response time of the algorithm.

Before we proceed let us describe the ATM simulation configuration used to test our scheme: All simulations are done in BONeS [ALT94]. BONeS is "an event driven" simulator. Each network device (such as a source or a switch) is an independent entity with its own independent state machine. An entity receives events which are either ATM cells or clock pulses. An entity may send ATM cells or request a clock pulse to occur after a specific period of time. The ATM end systems are as specified in [Sat96], Appendix I. The major parameters of the end systems are as follows (unless specified otherwise): $Nrm = 32$, $AIR \cdot Nrm = 42.5Mbs$, $RDF = 256$, $PCR = 150Mbs$, $TOF = 2$, $TCR = 10cells/sec(4.24Kbs)$ and $ICR = 8.5Mbs$. Though we use the optional use-it-or-lose-it behavior, the use of this option by the sources does not effect the correctness of the algorithm (and in particular, it does not effect most of our simulations, where the sources are greedy and the "lose-it" behavior is unlikely to show-up). The links parameters (bandwidth and delay) are given separately for each simulation. The parameters used in Phantom are specified in the sequel.

Measuring Δ: The amount of residual bandwidth, Δ, is measured in fixed time intervals of length τ and is accumulated in a parameter called Maximum Allowed Cell Rate (MACR) [Rob94, Bar95]. $MACR$, is updated at the end of each time interval by the weighted sum of its previous value and the measured Δ. I.e.,:

$$MACR = MACR \cdot (1 - \alpha) + \Delta \cdot \alpha, \qquad \alpha \in [0, 1]$$

In our simulations τ is set to 100 cell time and α is set to 1/16. (As explained in Subsection 3.2 the value of α is set to be a negative power of 2 so that the multiplication by α is simply a shift operation.)

While this method seems to work reasonably it turns out that a more robust mechanism is necessary. Consider the following scenario: a sudden burst causes Δ to be significantly smaller (or even have a negative value, if the traffic destined to that output port exceeds the link capacity) for a short period of time. This sharp change in Δ causes $MACR$ to be significantly smaller which in turn causes the sessions to reduce their rates, and hence the network to be underutilized. Now, because a low rate session transmits fewer RM cells, it takes the system a long time to gain back the normal utilization. To this end, and to avoid the possibility of erroneous negative $MACR$, we prevent $MACR$ from dropping sharply, i.e.,:

$$MACR := \max(MACR \cdot (1 - \alpha) + \Delta \cdot \alpha, MACR \cdot dec\_factor),$$
$$dec\_factor \in [0, 1]$$

(In our simulations $dec\_factor = 3/4$).

In Fig. 1 the Phantom algorithm as described so far, is employed to control the rate of four sessions sharing one 150 Mbs link. The figure shows MACR, the queue length at the output port as well as the sessions' ACR.

Fig. 2 presents the behavior of the algorithm under the same scenario but the four sessions have a diverse range of RTTs.

In Fig. 3 the fast reaction of the scheme to the addition of a fifth session is shown.

In Fig. 4 the algorithm is examined with on/off traffic.

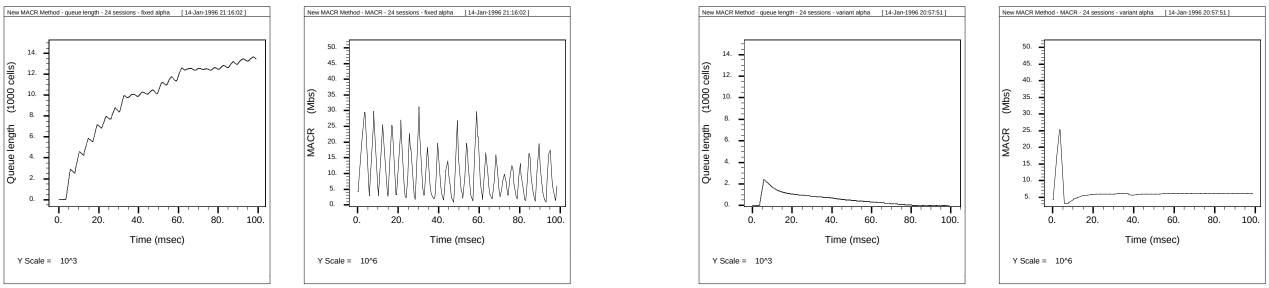
Sensitivity to queue length: Since the algorithm is insensitive to the absolute queue length it might happen that the queue length is growing without bound. Such a problematic case is likely to happen when many sessions are restricted by the same link. This phenomena is not unique to this algorithm, but is shared by many algorithms that are on the one hand insensitive to the queue length and on the other hand try to push the link utilization to a high level. (Such algorithms might have a temporary overutilization which results in increased queue length.) Fig. 5 illustrates such a scenario.

In order to solve this problem we extend the algorithm by a simple modification that makes it sensitive to the queue length. Instead of using a fixed value α in the computation of MACR we use two different variables α_{inc} and α_{dec}. The variable α_{inc} is used instead of α when $\Delta > MACR$ and α_{dec} is used when $\Delta \leq MACR$. Moreover, the actual value of α depends on the queue length, when the queue length is relatively small α_{inc} is relatively large and α_{dec} is small. On the other hand, when the queue length is large, α_{dec} is relatively large and α_{inc} is relatively small. This method shortens the convergence time of sessions and thus decreases the period of time in which the link is either underutilized or overutilized.

The values of α_{inc} and α_{dec} used in our simulations (Fig. 5) are presented in Table 1 as a function of the current queue length and the parameters QT (Queue Threshold, which is 50 cells in our simulations) and α.

queue length					α_{inc}	α_{dec}
		queue length	$\leq$	QT	α	$\alpha/16$
QT	$<$	queue length	$\leq$	$2QT$	$\frac{\alpha}{2}$	$\alpha/8$
$2QT$	$<$	queue length	$\leq$	$4QT$	$\frac{\alpha}{4}$	$\alpha/4$
$4QT$	$<$	queue length	$\leq$	$8QT$	$\frac{\alpha}{8}$	$\alpha/2$
$8QT$	$<$	queue length	$\leq$	$16QT$	$\frac{\alpha}{16}$	α
$16QT$	$<$	queue length			$\frac{\alpha}{32}$	$2 \cdot \alpha$

Table 1: Phantom - Computing α_{inc} and α_{dec} according to the current queue length.



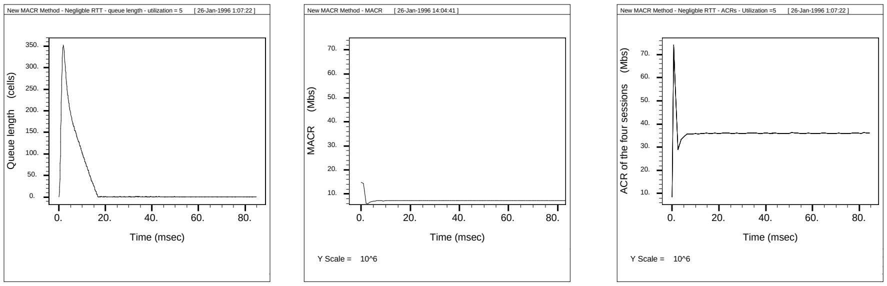
Queue length MACR
Constant α

Queue length MACR
α depending on queue length

Figure 5: Phantom - Twenty four greedy sources sharing a 150 Mbs link, each with a negligible RTT (0.01 ms). The two left hand diagrams illustrate the problematic case where α is constant. There is a "synchronization" effect between the sessions, i.e., whenever there is a small change in the value of $MACR$, it causes a large total change in the rate of all the sessions together. This change causes a large change in Δ and vice versa. Hence, there are large oscillations of the sessions rates which do not converge even after a long period of time and the queue length grows without bound. In the two right hand diagrams α_{inc} and α_{dec} are computed according to Table 1. The queue length peaks at 2,000 rather than continuously increasing, and $MACR$ converges nicely and smoothly to the desired value (which is $\frac{150}{24+1} = 6Mbs$). The large burst at the beginning of the simulation is caused by the "synchronized" burst of 24 sources. However, after this initial peek the value of $MACR$ is stable and the queue is drained.

Utilization: The underutilization of the algorithm when only very few sessions share a link, may be avoided by using the following modification to the basic scheme. Rather than restricting the sessions by Δ we may restrict them by a constant (called $utilization\_factor$) times $MACR$. For example, if we restrict the sessions by $3 \cdot MACR$, and we have three sessions sharing a 100Mbs link, then each session is allocated 30 Mbs (and the phantom session gets 10 Mbs). Our simulations show that this modification does not effect the other characteristics of our scheme. (The possible effect could be a larger transient queue length and larger oscillations in the value of $MACR$.) In Fig. 6 the simulation of Fig. 1 is repeated but the sessions rates are restricted by $5 \cdot MACR$ instead of just MACR.

Variance Consideration: When many sessions share one link with a high utilization factor there may be large oscillations in MACR and in the queue length (Fig. 8). To eliminate this



Queue length MACR Sessions' allowed rate

Figure 6: Phantom - Four greedy sessions with negligible RTT sharing a 150 Mbs link. The ratio between $MACR$ and the link restriction is 5. Each session stabilizes on a rate of $\frac{150}{4+\frac{1}{5}} = 35.7Mbs$ while $MACR$ stabilizes on a value of 7.14 Mbs. Hence, the utilization during steady state in this case is more than 95%. However, one can see that during convergence time, the queue length might have a temporary larger value (350 cells), compared with that in Fig. 1 (where no queue was built up).

phenomena we first approximate the standard deviation in Δ, and then take it into consideration in the calculation of α_{inc} and α_{dec}.

Following [Jac88] the standard deviation is approximated by the mean deviation (which is easier, as it does not require a square root computation) as follows:

$ERR := MACR - \Delta$

$\sigma := \sigma \cdot (1-h) + |ERR| \cdot h, \qquad h \in [0,1]$

(We set $h = 1/16$ in our simulations).

However, using this computation one can not distinguish between the case in which σ is large due to an external change, such as the addition or removal of a new session, and the case in which the large variation is caused by a small change in the calculation of $MACR$ that is then amplified by the large gain in the feedback loop (many sessions with large utilization factor). Note that each of the two cases requires a different treatment. In the first case Phantom should normally react with the large gain when updating $MACR$ (adapting to an external change in the traffic) while in the second case, the gain in the control loop should be reduced to flatten the oscillations in $MACR$, as suggested in Table 2.

To distinguish between the two cases we use two additional variables σ_{pos} and σ_{neg}. When $\Delta \geq MACR$, σ_{pos} is used and when $\Delta \leq MACR$, σ_{neg} is used. The variable σ is then set to the minimum between σ_{pos} and σ_{neg}. The pseudo code for the estimation of σ is presented in Figure 7.

The reasoning behind this approach is as follows: If the large variance is caused by an external change then only one of the variables $\sigma_{pos}, \sigma_{neg}$ reaches large values. However, if the large variance is caused by perpetual big oscillations then both variables are relatively large.

The right hand diagrams of Fig. 8 show the behavior of the algorithm employing the variance method. We have also tested the behavior of the algorithm in a dynamic changing environment with four greedy sessions and four on/off sessions in an environment identical to that of Fig. 4 and the results were exactly the same as in Fig. 4. Thus testifying that the variance method

Every τ seconds time interval do:

$ERR := MACR - \Delta$

If $ERR \geq 0$ then

$\sigma_{neg} := \sigma_{neg} \cdot (1-h) + ERR \cdot h$

Else

$\sigma_{pos} := \sigma_{pos} \cdot (1-h) + |ERR| \cdot h$

$\sigma := \min(\sigma_{neg}, \sigma_{pos})$

$ratio := f(\sigma)$ {According to Table 2}

$(\alpha_{inc}, \alpha_{dec}) := g(queue\ length, \alpha)$ {According to Table 1}

$\alpha_{inc} := \alpha_{inc} \cdot ratio$; $\alpha_{dec} := \alpha_{dec} \cdot ratio$

Figure 7: Phantom - Computation of α_{inc} and α_{dec} according to variance consideration.

σ	$ratio := f(\sigma)$
$\sigma \leq MACR$	1
$MACR < \sigma \leq 2\ MACR$	1/2
$2\ MACR < \sigma \leq 4\ MACR$	1/4
$4\ MACR < \sigma \leq 8\ MACR$	1/8
$8\ MACR < \sigma$	1/16

Table 2: Phantom - Computation of the ratio function.

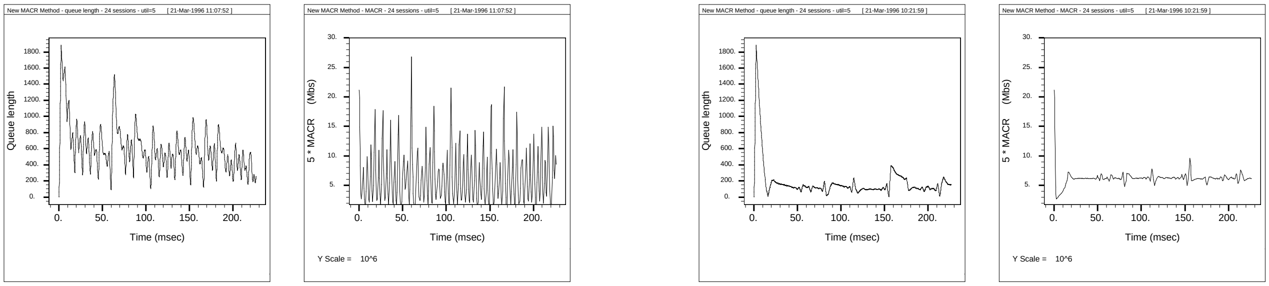
does not seem to affect the behavior of the algorithm when there are large external changes in the traffic.

Reducing the maximum queue length: Consider the following scenario: The MACR of a link that is shared by many sessions is suddenly increased from a small value to a relatively large value (e.g., some VBR connections aborted). In this case all the sessions that were bottlenecked on this link increase their rates to $MACR \cdot utilization\_factor$ at about the same time (assume large AIR), resulting in a surge of traffic much larger than the link capacity thus developing a large queue (see Fig. 9).

We employ two mechanisms to significantly reduce this large developing queue. First, (following the ideas of the TCP window flow control algorithm,) rather than allowing a session rate to jump at once to $MACR \cdot utilization\_factor$ we restrict the increase of its rate to at most doubling during each round trip time (by allowing the ER on the backward RM cell to be at most twice the value of the CCR on the RM cell). The price of this mechanism is a somewhat slower convergence when only a few sessions share the link. Fig. 10 examines this new approach in the same environment as the simulation of Fig. 9.

While the first method has significantly reduced the maximum queue length a careful observation of Fig. 10 reveals that the rate of an arbitrary session is still increasing when there is already a significant queue buildup (over utilization). Therefore we employ a second mechanism that monitors the available bandwidth more closely than $MACR$. The new measure is called $Fast\_MACR$ and is computed in the same way as $MACR$, i.e.:

$$Fast\_MACR := Fast\_MACR \cdot (1-\beta) + \beta \cdot \Delta, \quad \beta \in [0,1]$$



Queue length 5 · MACR
Without sensitivity to variance

Queue length 5 · MACR
$\alpha_{inc}, \alpha_{dec}$ sensitive to variance

Figure 8: Phantom - twenty four greedy sessions with negligible RTT sharing a 150 Mbs link where *utilization_factor* $= 5$. The reason for the large oscillations in Δ is the fact that a small change in the value of $MACR$ is multiplied by 5 and affects 24 sessions (i.e., the optimum value of $MACR$ is 1/121 of the link capacity). The two right hand diagrams demonstrate how the variance method overcomes this problem by drastically reducing the oscillations in $MACR$ and thus also reducing the queue length.

where β is larger than α ($\beta = 1/8$ in our simulations), thus $Fast\_MACR$ is more sensitive to fast changes in the available link bandwidth. Whenever $MACR$ is significantly larger than $FAST\_MACR$ we assume that $MACR$ is too large and set the NI (No Increase) bit in each backward RM cell. Any source that observes this bit set, may not increase its rate (on top of all the other restrictions, [Sat96]). Fig. 11 illustrates the effect of this method on the same scenario as in Fig. 9.

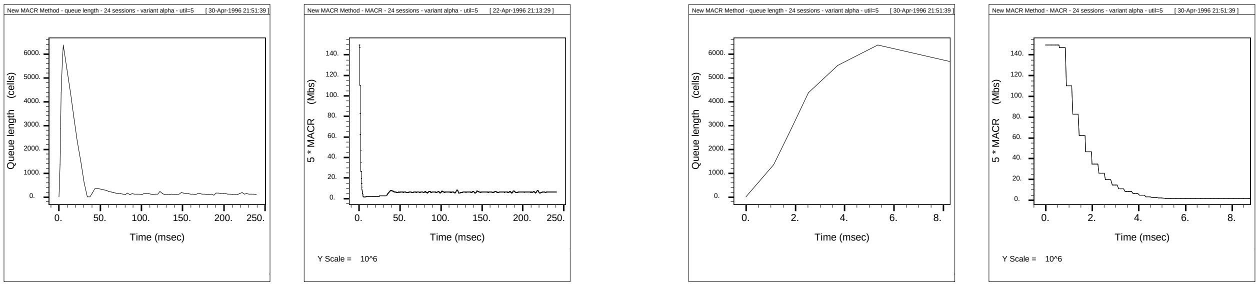
Fig. 12 is a detailed version of the pseudo code of the algorithm which is common to both ATM and TCP. Fig. 13 contains a part of the algorithm that is due to explicit rate environments. This part is implemented in ATM.

3.1 Convergence Analysis

We associate with each link a *Fair Share* variable, FS, that constitutes an upper bound on the rate of the sessions that cross this link. The rate of every session that crosses the link is either exactly FS or restricted elsewhere to a lower value. Let ls be the set of sessions whose rate is smaller than FS, $\Sigma\, ls$ the total rate of sessions in ls, $\#s$ the total number of sessions sharing the link, $\#ls$ the number of sessions in ls, and C be the link bandwidth. Max-min fairness is then achieved if $FS = \frac{C - \Sigma\, ls}{\#s - \#ls}$ [BG87, Jaf81, GB84, Cha94, Jai96]. Note that a set of flows is max-min fair if the flow of each session equals the minimum fair share along its path.

Using the above terms we define for the Phantom algorithm, $FS_{Phantom}$ for every link to be $\frac{C - \Sigma\, ls}{\#s - \#ls + \frac{1}{k}}$ where k is the *utilization_factor* used in the algorithm.

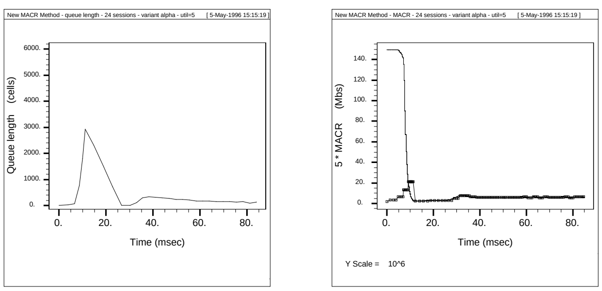
Clearly when the rate of every session is the minimum $FS_{Phantom}$ along its path, the network is in a stable state (assuming steady traffic requirements). Note that if $k = 1$ then the fair share of Phantom is exactly the same as that of the max-min allocation, when on every link there is an additional imaginary (phantom) greedy session. As k is increased the bandwidth consumed by the phantom session, in a stable state, is decreased while the rest of the bandwidth is *fairly* allocated between the other sessions.



Queue length $\quad$ $5 \cdot MACR$ $\quad$ Queue length $\quad$ $5 \cdot MACR$

Over long time $\quad$ Zoom during transient time

Figure 9: Phantom - 24 greedy sessions with negligible RTT sharing a 150 Mbs link and $utilization\_factor = 5$. The initial value of $5 \cdot MACR$ is the full link capacity, i.e., 150 Mbs. The initial rate of each session (ICR) is relatively small (1.7 Mbs). The large queue length (more than 6000 cells) is created during the transient time, until $MACR$ becomes close to its optimal value (6.2 Mbs).

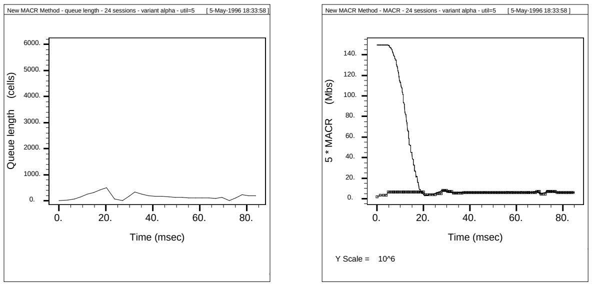


Queue length $\quad$ $5 \cdot MACR$ and the rate of an arbitrary session

Figure 10: Phantom - 24 greedy sessions with negligible RTT sharing a 150 Mbs link where $utilization\_factor = 5$. Sessions may at most double their rate each round trip time. The maximum queue length is now 3000 cells.

When employed in a network with a fixed set of greedy sessions Phantom behaves like control system with negative feedback. In a stable state that system either oscillates around, or converges to, the max-min fair allocation with an additional "phantom" greedy session on each link. It is a negative feedback system because when the spare capacity is too small (smaller than $k \cdot MACR$) the rate of the large sessions is decreased towards $k \cdot MACR$ and when the spare capacity is larger than the maximum session rate, the rate of the sessions is increased. The amount of oscillation in the control system depends on various parameters such as, the delay (i.e., RTT), gain factor (which roughly depends on the number of sessions). The heuristics presented in this section suppress the magnitude of the oscillations and brings the value of $k \cdot MACR$ close to that of $FS_{Phantom}$.

A simulation of the algorithm in a more complex configuration is presented in Appendix A.



Queue length $\quad$ $5 \cdot MACR$ and rate of an arbitrary session

Figure 11: Phantom - 24 greedy sessions with negligible RTT sharing a 150 Mbs link where $utilization\_factor = 5$. Sessions' rate is at most doubled each round trip time and the NI bit is set if $MACR$ is significantly larger than $Fast\_MACR$. The maximum queue length is now 500 cells. Note that as soon as congestion starts, the increase in the rate of the selected session stops.

3.2 Additional Remarks:

1. Phantom is set to acquire any available bandwidth as soon as it becomes available. Because, in calculating Δ we may subtract the total incoming traffic (from all traffic classes) from the total link capacity thus acquiring immediately any bandwidth that becomes available (the UBR traffic may be excluded depending on the relative priority).

2. In certain variations of Phantom forward RM cells may be eliminated, saving 3% of the bandwidth (forward RM cells are sent each Nrm-1 data cells, Nrm typically equals 32). However, backward RM cells would have to be originated by the destination every Nrm - 1 data cells received. Note that the current specifications ([Sat96]) allow a creation of RM cells at the destination.

3. If τ is set to be a positive power of 2 then the division operation that is used to calculate Δ can be implemented using a simple shift. Similarly, if α, β, h, $utilization\_factor$ and $decreasing\_factor$ are negative powers of 2 then the multiplication operations in the algorithm can be implemented using a simple shift.

4. The peak queue length is affected by the value of QT. If a further reduction of the peak queue length is desired then the value QT should be decreased. For example, if QT =16 (instead of 50, as in the simulations presented in this paper), it results in a peak queue length of 350 cells (compared with the 500 in Fig. 11 in the same settings, these simulations are not included).

The behavior of Phantom in the presence of VBR traffic is presented in Appendix B.

Constant: C { The total link capacity }
Parameters: τ, α, β, QT, $decreasing\_factor$, h, $utilization\_factor$, $init\_portion$
{In our simulations $\tau = 100$ cell times, $\alpha = \frac{1}{16}$, $\beta = \frac{1}{8}$, $QT = 50$ cells, $decreasing\_factor = \frac{3}{4}$,
$h = \frac{1}{16}$, $utilization\_factor = 1$ or 5, $init\_portion = 0.1$ }
Variables: $MACR$, Δ, α_{inc}, α_{dec}, $\sum traffic$, $Fast\_MACR$, ERR, σ_{neg}, σ_{pos}, $ratio$

Initialization:
$MACR := initi\_portion \cdot C$ {Initial settings of Max Allowed Cell Rate}
$Fast\_MACR := init\_portion \cdot C$
$\sum traffic := 0$

When a forward message M is received (either a TCP/IP packet or an ATM cell):
$\sum traffic := \sum traffic +$ size of M

Every τ seconds time interval do:
$\Delta := \min(C - \frac{\sum traffic}{\tau}, \frac{C}{utilization\_factor})$ {Traffic rate of last interval}

$ERR := MACR - \Delta$ {Compute α_{inc}, α_{dec} according to Fig. 7}
If $ERR \geq 0$ then {Computing the mean deviation.}
$\sigma_{neg} := \sigma_{neg} \cdot (1-h) + ERR \cdot h$
Else
$\sigma_{pos} := \sigma_{pos} \cdot (1-h) + |ERR| \cdot h$
$\sigma := \min(\sigma_{neg}, \sigma_{pos})$
$ratio := f(\sigma)$ {$f(\cdot)$ is given in Table 2}
$(\alpha_{inc}, \alpha_{dec}) := g(queue\ length, \alpha)$ {$g(\cdot)$ is given in Table 1}
$\alpha_{inc} := \alpha_{inc} \cdot ratio$; $\alpha_{dec} := \alpha_{dec} \cdot ratio$

If $\Delta > MACR$ then {Compute $MACR$}
$MACR := MACR \cdot (1 - \alpha_{inc}) + \Delta \cdot \alpha_{inc}$
Else
$MACR := \max(MACR \cdot (1 - \alpha_{dec}) + \Delta \cdot \alpha_{dec},$
$MACR \cdot decreasing\_factor)$

$Fast\_MACR := Fast\_MACR \cdot (1 - \beta) + \Delta \cdot \beta$ {Compute $Fast\_MACR$}

$\sum traffic := 0$

Figure 12: Phantom - Detailed Pseudo Code for computing and updating the parameters, in particular $MACR$. This is the part of the algorithm which is common to both explicit and implicit mechanisms. Figures 13 and 18 contain the rest of the code where the computed $MACR$ is used to effect the flow control.

When a backward RM cell BC is received:

$\{ER$, CCR and NI are three fields in BC, the received backward RM cell. $\}$

$BC.ER :=\min(utilization\_factor \cdot MACR\ ,\ BC.ER\ ,\ 2 \cdot BC.CCR)$

If $(MACR > 2 \cdot Fast\_MACR)$ then

$BC.NI :=$ True $\{MACR$ is too large, set the No Increase field on the Backward Cell.$\}$

Figure 13: Phantom - Addendum required for an implementation in ATM environment, where an explicit indication mechanism is used.

4 Applying Phantom to flow control in TCP

4.1 TCP flow control techniques

In the past few years several new techniques for flow control of TCP traffic in IP networks were introduced and employed. We distinguish between flow control mechanisms that operate at the end stations and those that operate at the routers/gateways.

Source end stations mechanisms: Two such algorithms are currently proposed, Reno, suggested by Jacobson [Jac88], and Vegas, by Brakmo and Paterson [BP95]. Both algorithms operate by dynamically adjusting the TCP window size. If the network is observed not congested the window size is increased while if the network is observed congested, the window size is decreased. The basic idea is to consider the network congested if packet loss is detected. In Vegas there are several additional mechanisms one of which is a more refined consideration of the round trip delay (the time interval between the departure of a packet and the arrival of its acknowledgment) in order to detect congestion.

Router mechanisms: Obviously when the router buffer overflows some packets are dropped. Another option for the router is to send an ICMP Source Quench message [Ste94, BP87], which causes the source to decrease its window size. This option however, is rarely used as the Source Quench message may overload an already congested network. Furthermore, this mechanism is biased against certain kinds of traffic as shown in [FJ92].

A more sophisticated strategy, called RED (Random Early Detection) was presented by Floyd and Jacobson [FJ93], where either packets are dropped or Source Quench messages are sent when the router's queue length exceeds a certain threshold. The packet to be dropped (or the source to which a Source Quench message is sent) is selected according to some probability distribution on the packets. The RED approach reduces the queue lengths at the router and overcomes some of the bias described in [FJ92]. Yet, the resulting mechanism still does not always guarantee fairness (as discussed in the sequel).

For networks in which each router sends packets in a round robin order between the sessions, another router mechanism was presented by Keshav [Kes91]. In this method the source measures the maximum queuing delay by sending two consecutive packets and measuring the time difference at which they return to the source. This mechanism may achieve high degree of fairness and does not depend on any specific network topology, however its implementation complicates the router architecture (requires a queue per session) and requires the cooperation of all the routers.

For further discussion on routers congestion control see the survey of Mankin and Ramakrishnan [MR91].

Three problems are likely to occur in TCP traffic flow control that does not use explicit rate indication:

Unfair allocation: With the exception of [Kes91] the above mechanisms do not guarantee fairness. For example when two sources that use Vegas [BP95], get different window sizes, and both have the same delay thresholds (α, β) then, there is no mechanism that would balance them. The current mechanisms would either increase both or decrease both. Another source for unfairness in allocation is the bias against sessions that pass through many routers (analogous to the "beat down" phenomena in ATM, [BdJ94]). An unfair behavior of Reno in an environment of drop tail routers is depicted in the left hand side of Fig. 14 and Fig. 17.

Congestion detection: Some of the above mechanisms base their estimate of the queuing delay on the knowledge of the round trip delay (RTT). However, RTT is the sum of the queuing delays and propagation delays along the path. As the source uses only the total delay it is almost impossible to separate the two components, leaving a large degree of uncertainty about the queuing delay.

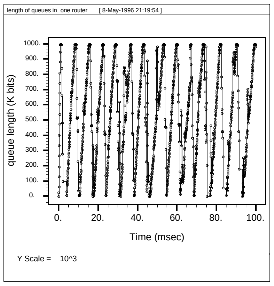
Sensitivity to parameters: Some of the above schemes are sensitive to variations in parameters values. For example, out of tune time threshold parameters may cause network underutilization (see the discussion in [BP95] for example). Alternatively, distinct parameters for different sessions may cause severe unfairness. E.g., two sessions using Vegas and sharing one router such that the lower time threshold (α) of the one is larger than the upper time threshold (β) of the other. Then convergence is achieved only when the first session uses the entire link bandwidth.

4.2 Employing Phantom in TCP

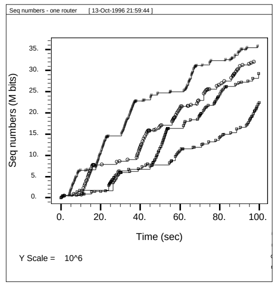
Here we show how the employment of the Phantom approach may solve some of the problems described above. We suggest and discuss different possibilities of applying the basic Phantom scheme to the flow control of TCP traffic over IP networks. Our suggestions introduce minimal overhead, while achieving smooth interaction with current TCP equipment, thus it can be gradually introduced into TCP environments. For the implementation of Phantom, TCP routers would have to compute $MACR$, either as explained above or by measuring the amount of unutilized link bandwidth (which is an approximation for the value of $MACR$).

There are two basic approaches in the application of Phantom to TCP:

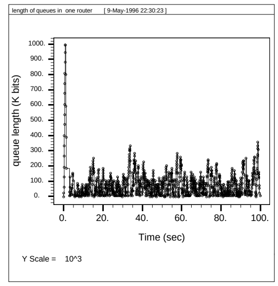
Without modifying the routers: The source periodically polls the routers on the path to the destination using e.g., SNMP. In each polling the source fetches the MACR values of those routers, and adjusts its rate according to the minimum MACR. This approach is simple, however it consumes additional network resources and may suffer from delay problems.



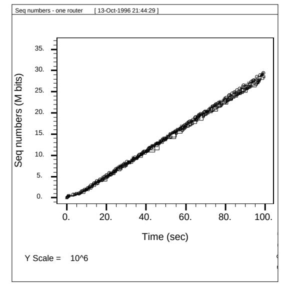
Queue length



Goodput of different sessions



Queue length



Goodput of different sessions

Reno, Drop tail router without selective discard

Reno, Drop tail router with selective discard

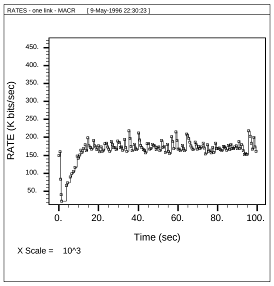
Figure 14: Four TCP sessions sharing a 1.5 Mbs link, each with a small RTT (40 ms). The two left hand diagrams illustrate the problematic case of a drop tail mechanism as is. Though all the sessions go via the same path, the link bandwidth is not equally shared (different sessions get different goodput). Note the sharp changes in the queue length. When the selective discard mechanism is used (two right hand side diagrams), the allocation is more fair and the queue length is moderate (400,000 bits, except for the initial peak).

Modifying the routers: In this approach the source employs either Reno or Vegas [Jac88, BP95] and in addition it indicates its current rate (CR) in the IP (or TCP) header. Thus requiring (1) a modification of the TCP/IP header and (2) simple modifications in the routers. This modification enables the routers to identify sources whose current rate is relatively large (by comparing the sources current rate to the $MACR$). We suggest four different ways by which the routers "tell" the source to decrease its rate in case of congestion:

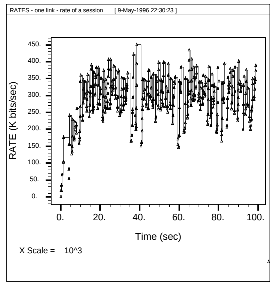
1. **Explicit Rate Indication:** Simply implement in TCP a mechanism similar to the ABR rate base flow control. That is, the routers indicate the Explicit Rate (ER) on backward packets and the source station transmits according to that rate (in addition to the constraints imposed by the window size mechanism).

2. **Selective Discard:** The router discards any packet for which the indicated rate (CR) is larger than $utilization\_factor \cdot MACR$. The pseudo code of this mechanism is presented in Fig. 18. This mechanism avoids congestion even in drop tail routers while reducing both the bias discussed in [FJ92] and the "beat down" problem. The right hand sides of Fig. 14 and Fig. 17 illustrate the behavior of this mechanism.

3. **Selective Source Quench:** A slight modification of the previous approach, where the router sends Source Quench messages to sources whose connection rate is above $utilization\_factor \cdot MACR$. The source reacts to the receipt of a Source Quench message as if a packet was dropped [BP87], and hence reduces its window size. Note that these messages might consume scarce network bandwidth at time of congestion. Alternatively, if an $EFCI$ (Explicit Forward Congestion Indication) bit is used on the IP header, the router can set this bit. See [Flo94] for discussion of the use of a Source Quench mechanism and



MACR



Session's measured rate

Figure 15: Four TCP sessions sharing a 1.5 Mbs link, each with a small RTT (40 ms) as in Fig. 14. We observe the values of $MACR$ and the rate of an arbitrary session when the Selective discard mechanism is implemented. Note that the session rate is not stable due to the inherent burstiness of the window mechanism. However, the average rate is about twice the $MACR$ ($utilization\_factor = 2$).

the addition of a new EFCI bit. [RF99] also discusses the Explicit Congestion Notification in IP.

4. **Selective RED:** Here the router applies the RED mechanism (Random Early Detection). However, only packets whose rate is larger than *utilization _ factor*$\cdot MACR$ may be dropped. Since we are selective in the packets we drop, we increase the fairness of the RED routers by reducing the "beat down" phenomena though this phenomena is not eliminated (in simulations, not presented here, it is seen that a session that crosses several routers is more vulnerable to such a drop even when the Selective RED mechanism is implemented).

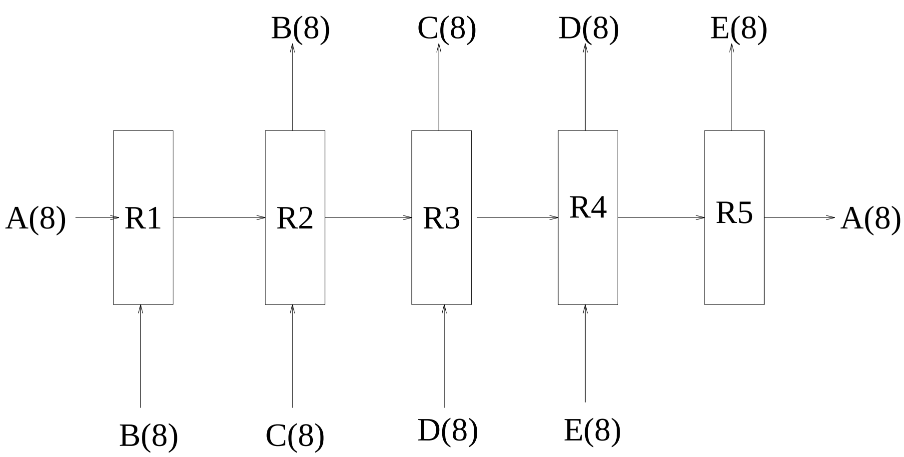
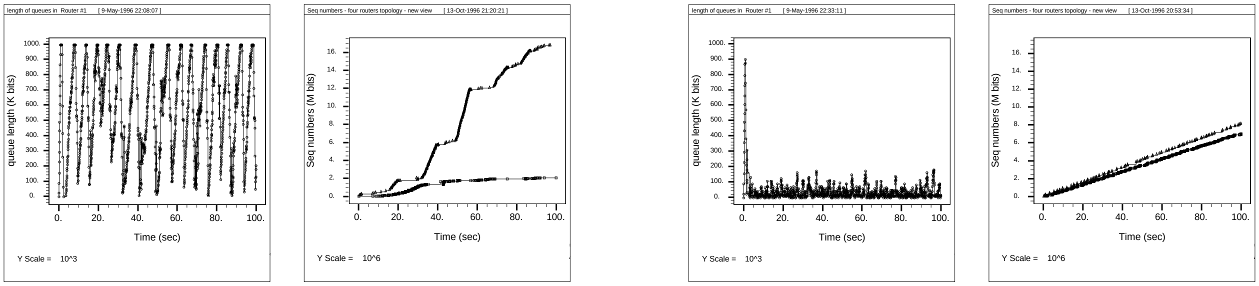


Figure 16: A configuration to measure the "beat down" phenomena. There are eight "long distance" sessions that cross four links and on each link there are eight additional "local" sessions. Each link has a bandwidth of 1.5 Mbs and a delay of 10 msec.

5. **Priority queue mechanism.** Here we suggest a new novel approach. The key idea is that the router would treat "bad" sessions differently, i.e., would give priority to the well behaving sessions. Sessions whose rate exceeds $utilization\_factor \cdot MACR$ would be



Queue length | Goodput of different sessions | Queue length | Goodput of different sessions

Reno, Drop tail routers without selective discard | Reno, Drop tail routers with selective discard

Figure 17: TCP sessions in the configuration described in Figure 16. In the left hand diagrams we see that the rate of a "local" session (from group B) is much larger than the rate of a "long distance" session (from group A). Note in the right hand side that the selective discard mechanism ensures a high level of fairness (almost equal allocation between "local" and "long distance" sessions) while keeping a moderate queue length.

considered misbehaved and would get a worse service. This approach might be appropriate especially to environments where the sources implement Vegas, and hence, are very sensitive to the measured delay. This attitude is only in its first steps and we are looking for different policies according to which the different treatment is given.

The suggested mechanisms can be used in only part of the network routers or in only part of the end stations as it is possible to operate both new and old methods in the network without that they prohibit each other. Note, that the "new" TCP source code is an extension of either Vegas or Reno, and uses their window adjustment policy (the only change done by the source is the use of an additional field to indicate the current rate). Hence, the current "old" routers have identical treatment to packets of both "new" and "old" end stations. To this end, the "new" router contains two queues, `old-queue` for packets that do not follow the new algorithm and `new-queue` for packets that do. Packets are selected for transmission from the two queues in a round robin order.

An additional motivation to implement the newly suggested flow control mechanism in TCP is that TCP traffic might traverse ATM networks. The use of a consistent flow control mechanism in both TCP and ABR over ATM may improve the network utilization.

4.3 Simulations of Selective Discard

We use BONES [ALT94] for our simulations. The TCP end systems implement Reno according to the pseudo code specified in Section 21 in [Ste94]. We assume greedy sources where size of packets is 512 bytes. Each source computes its rate as the ratio between the size of payload transmitted and acknowledged by the destination in a time interval, and the length of the time interval. The amount of transmitted payload is the difference in the number of bytes acknowledged at the end and the beginning of the interval. To avoid noisy peaks in this measurement we use the standard exponential averaging. The pseudo code appears in Fig. 19.

When a forward message M is received:

$\sum traffic := \sum traffic +$ size of M $\{$As indicated in Fig. 12$\}$

If $(M.CR > utilization\_factor \cdot MACR)$ then $\{CR$ is the source current rate.$\}$

Discard M

Figure 18: Selective Discard - Addendum required for an implementation of Phantom (as presented in Fig. 12) in IP environment.

Parameters: τ , γ

$\{$ In our simulations $\tau = 100ms$ and $\gamma = 1/16$. $\}$

Initialization:

$Curret\_Rate := 0$; $Curr\_Ack := 1$; $Max\_Ack := 1$

Whenever an ACK is received:

If $ACK.Seq > Max\_Ack$ then $Max\_Ack := ACK.Seq$

At the end of each time interval of length τ:

$Rate := \frac{Max\_Ack \; - \; Curr\_Ack}{\tau}$

$Curret\_Rate := Curret\_Rate \cdot (1 - \gamma) + Rate \cdot \gamma$

$Curr\_Ack := Max\_Ack$

Figure 19: Phantom - Computation of the Current Rate by a TCP source-end-system.

The *goodput* of a session in any point of time is the total amount of "good" data that has been received by the sessions' destination so far. We use the *goodput* measure for illustrations (Fig. 14 and Fig. 17) rather than the rate of traffic measured by the source at any point of time. This approach of illustration is used as the rate of TCP sessions is very bursty due to the manner in which the window is going up and down (see Fig. 15 for an illustration of the rate burstiness).

As for Phantom, the parameters are set as in Section 2 except for QT which is 8192 bits, τ which is 100 msec and $ratio$ (denoted also as $f(\sigma)$) which is one quarter of the value in Section 2 (due to the window mechanism in TCP, the variance (burstiness) is much higher than in ABR). The $utilization\_factor$ is set to 2, i.e., when using the selective discard mechanism, each packet with an indicated rate that is more than $2 \cdot MACR$ is dropped.

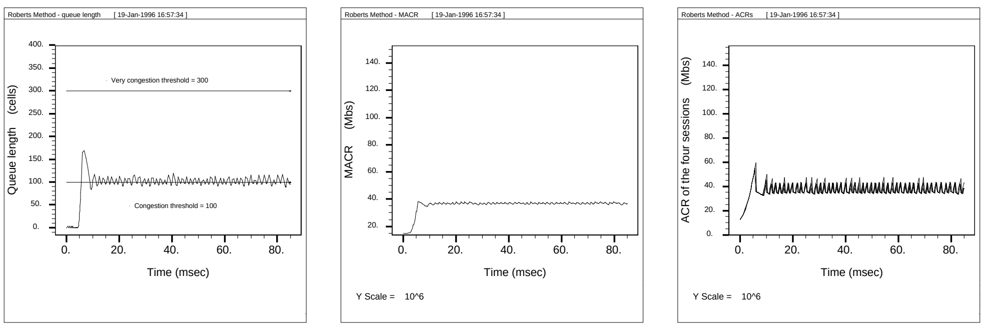
Fig. 14 illustrates the behavior of the drop tail mechanism and the selective discard mechanism in a one link configuration. Figure 15 shows the values of $MACR$ and the rate of an arbitrary session when the selective discard mechanism is implemented.

Figure 16 presents a configuration where the "beat down" phenomena is likely to occur. Figure 17 illustrates the behavior of the drop tail mechanism and the selective discard mechanism in this configuration.

5 Comparison with other ATM rate based algorithms

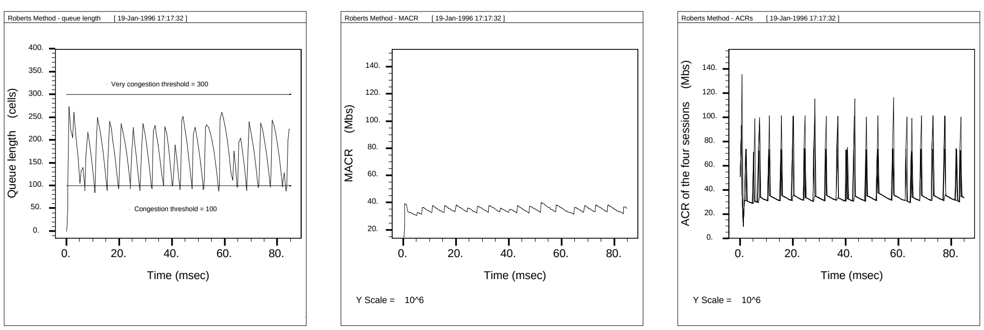
We compare Phantom to three other constant space rate-based flow control algorithms that have been proposed in the ATM forum, EPRCA, suggested by Roberts [Rob94], APRC, suggested by Siu and Tzeng [ST94] and CAPC, suggested by Barnhart [Bar94]. We give a brief description of each algorithm and compare it to Phantom. Another well known constant space rate-based flow control algorithm is OSU, suggested by Jain et. al. ([JKV94]). However, its advanced versions - ERICA/ERICA+ ([JKVG95, JKG+95, KJF+97]) maintain a counter per session. (The interested reader should look at a more complete survey by Jain [Jai96, JKV97].)

All three algorithms compute for each output port in each switch a fair share parameter, (called MACR in EPRCA and APRC and ERS in CAPC). This parameter is used to delimit the rate of sessions passing through the output port.



Queue length MACR Sessions' allowed rate

Very small increase at source ($AIR \cdot Nrm = 4.2Mbs$)



Queue length MACR Sessions' allowed rate

Larger increase at source ($AIR \cdot Nrm = 42.5Mbs$)

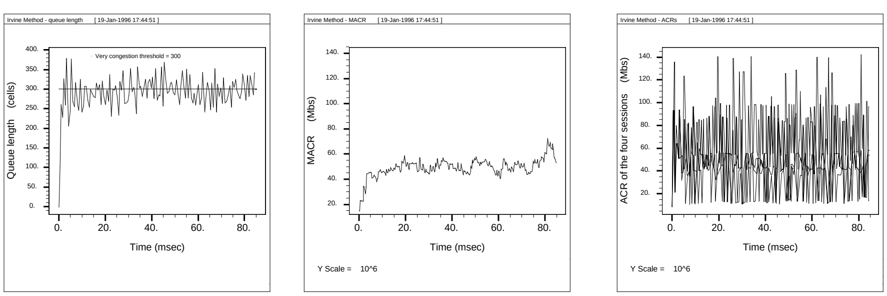
Figure 20: EPRCA - Four greedy sources sharing a 150Mbs link, RTT is negligible. Congested threshold is 100 cells and Very congested threshold is 300 cells. Values of other parameters are as recommended in [Rob94].

5.1 EPRCA and APRC

The EPRCA (Enhanced Proportional Rate Control Algorithm) [Rob94] was proposed by Roberts at the July 1994 ATM Forum meeting. In this algorithm the fair share parameter (MACR) is computed for each output port from the information received on the forward RM cells generated by the session's source end system. The computation of MACR, and of the sessions' rate limitations, depend on the queue state, which could be either *not congested*, if the queue length is smaller than a certain threshold, *congested*, if above the threshold, or *very congested*, if above a higher threshold. If the queue is *not congested*, the sessions' rates are not limited. However, when the queue is either *congested* or *very congested* then the rates are restricted by some fraction of MACR. That is, the queue size plays a major role in the control loop. Furthermore, it influences the MACR indirectly through RM cells that go from the switch to the end-system and back to the switch. This extra delay in the control loop is a source for inherent oscillations in the algorithm as can be seen in Fig. 20. In the first part of Fig. 20 sessions increase their rate, when allowed, by a very small amount, while in the second part they may increase their rate more aggressively (the more aggressive increase rate is the one used in all other simulations in this paper). Aside from the oscillations, the extra delay in the control loop may cause an unfair sharing of the bandwidth between sessions with vastly different round trip delays [CGBS94, JKVG94, CRBdJ94].

In a modification of EPRCA, proposed by Siu and Tzeng called APRC (Adaptive Proportional Rate Control with intelligent congestion indication) [ST94] the definition of *congested* state is changed. Rather than being a function of the queue length it is now a function of the rate at which the queue length is changing. That is, if the derivative of the queue length is positive then the link is said to be *congested.*

Fig. 21 illustrates the behavior of APRC. Since the algorithm is insensitive to the absolute value of the queue length, the queue length might increases until it becomes *very congested* (because when in state *not congested* the queue length might increase more than it is decreasing while in state *congested*). Therefore, in some scenarios the queue length might often exceed the very congested threshold.



Queue length MACR Sessions' allowed rate

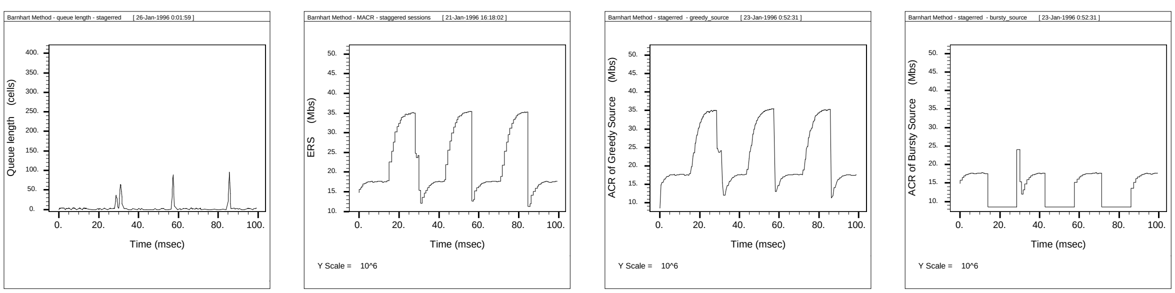
Figure 21: APRC - Four greedy sources sharing a 150Mbs link, RTT is negligible. Very congested threshold is 300 cells, values of other parameters are as recommended in [ST94].

5.2 CAPC

CAPC (Congestion Avoidance using Proportional Control) was proposed by Barnhart [Bar94]. It uses the fraction of unused capacity to control the algorithm actions. In this respect it is analogous to Phantom that uses the absolute amount of unused bandwidth to control the algorithm. CAPC estimate of the fair share is called ERS (Explicit Rate for Switch, which corresponds to our MACR). The main parameter used to control the changes of ERS is the ratio, r, between the rate of incoming traffic and the target link capacity. (Target link capacity is taken at 95% of the actual capacity.) Let $delta = 1 - r$, if $delta$ is positive then ERS is multiplied by an increase factor, otherwise ERS is reduced by a multiplicative decrease factor. In addition, if the queue length exceeds a certain threshold, the switch instructs the source end-systems to reduce their rates (using the CI field, a binary indication provided by the ATM standard on RM cells).

Fig. 22 illustrates the behavior of CAPC in an environment with on/off sessions where the algorithm parameters are as recommended in [Bar94]. The configuration is analogous to that in Fig. 4, Section 2. As can be seen CAPC has longer convergence time while its queue is relatively smaller during that time.

The larger value of the queue length in Phantom stems from the faster reaction of Phantom. We do not consider the momentary large queue as a problem, because Phantom is sensitive to the queue length (which enables it to smoothly and quickly drain the queue) and because the queue length consumed by Phantom can be reduced (as illustrated in Section 2).



Queue length | ERS (MACR) | Greedy sessions' allowed rate | Bursty Sessions' allowed rate

Figure 22: CAPC - Four greedy sessions and four on/off sessions sharing a 150 Mbs link with negligible RTT.

Since CAPC uses the binary indication bit in very congested states, it is prone to the unfair behaviors indicated in [BdJ94]. Sessions that cross many switches are more likely to be "beaten down" and to get a smaller allocation than sessions that cross only a few switches. Unlike Phantom, in CAPC it is necessary to know the exact bandwidth allocated to ABR, a quantity which may change over time due to the addition and deletion of CBR and VBR sessions. A good feature that CAPC and some versions of Phantom share, is that they do not require forward RM cells.

6 Conclusion

This paper introduces Phantom, a simple constant space flow control mechanism. The key idea behind this scheme is to bound the rate of sessions that share a link by the amount of unused bandwidth on that link. We have shown how to employ the method within the ATM framework for ABR traffic flow-control to achieve good flow control algorithms as indicated by our simulations. Some properties that the new scheme enjoys are: fair allocation, smooth convergence, simple hardware requirements and fast reaction to changes in the offered traffic.

In addition we have shown how the new method may be used for IP traffic flow-control. The most attractive implementation of Phantom in TCP is called *Selective Discard*. In this approach if the rate of a session is larger than the unused bandwidth of the link, then its packets are discarded. It turns out that Selective Discard improves the degree of fairness in TCP/IP, by dramatically reducing the penalty that sessions that cross multiple links suffer in the current flow-control algorithms for TCP/IP. To the best of our knowledge this is the first scheme that achieves this degree of fairness in an environment that is based on implicit indication, without employing specialized hardware mechanism (such as [Kes91]). Selective Discard does not interfere with the dynamic adjustment of the TCP window and hence, it can be gradually implemented together with current TCP equipment. However, the implementation of Selective Discard requires an addition of one field to either TCP or IP header (in which the current rate of a session would be marked).

Acknowledgments
We would like to thank K.K. Ramakrishnan and the four anonymous Sigcomm 96 referees for their very helpful comments.

References

[ALT94] ALTAGROUP. Bones designer core library reference. Technical report, December 1994.

[Bar94] A. W. Barnhart. Explicit rate performance evaluation. Technical Report ATM-FORUM/94-0983R1, October 1994.

[Bar95] A. W. Barnhart. Example switch algorithm for section 5.4 of tm spec. Technical Report ATM-FORUM/95-0195, Hughes Network Systems, February 1995.

[BdJ94] J. C. R. Bennett and G. T. des Jardins. Comments on the july prca rate control baseline. Technical Report ATM-FORUM/94-0682, July 1994.

[BF95] F. Bonomi and K. Fendick. The rate based flow control for available bit rate atm service. *IEEE Networks*, 9(2):24–39, March/April 1995.

[BG87] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice Hall, 1987.

[BP87] R. Braden and J. Postel. Requirements for internet gateways. Technical Report RFC 1009, IETF, June 1987.

[BP95] L. S. Brakmo and L. L. Paterson. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE JSAC*, 13,8:1465–1480, October 1995.

[CCJ95] A. Charny, D.D. Clark, and R. Jain. Congestion control with explicit rate indication. In *Proc. ICC*, June 1995.

[CGBS94] Y. Chang, N. Golmie, L. Benmohamed, and D. Su. Simulation study of the new rate-based eprca traffic management mechanism. Technical Report ATM-FORUM/94-0809, September 1994.

[CH96] A. K. Choudhury and E. L. Hahne. Dynamic queue length thresholds in a shared memory atm switch. In *Proc. 15th IEEE INFOCOMM*, pages 679–687, March 1996.

[Cha94] A. Charny. An algorithm for rate allocation in a packet-switching network with feedback. Technical Report MIT/LCS/TR-601, MIT Laboratory for Computer Science, April 1994.

[CR96] A. Charny and K.K. Ramakrishnan. Scalability issues for distributed explicit rate allocation in atm. In *Proc. 15th IEEE INFOCOMM*, pages 1182–1189, March 1996.

[CRBdJ94] A. Charny, K.K. Ramakrishnan, J. C. R. Bennett, and G. T. des Jardins. Some preliminary results on the eprca rate-control scheme. Technical Report ATM-FORUM/95-0941, September 1994.

[FH99] S. Floyd and T. Henderson. The newreno modification to tcp's fast recovery algorithm. Technical Report RFC 2582, Experimental, IETF, April 1999.

[FJ92] S. Floyd and V. Jacobson. On traffic phase effects in packet-switches gateways. *Internetw.: Res. and Exper.*, 3(3):115–156, September 1992.

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[Flo94] S. Floyd. Tcp and explicit congestion notification. *ACM Computer Communication Review*, 24,5:10–23, August 1994.

[Gaf82] E. Gafni. *The Integration of Routing and Flow Control for Voice and Data in a Computer Communication Network.* PhD thesis, Massachusetts Institute of Technology, Dept. of Elec. Eng. and Comp. Science, Cambridge, MA, August 1982.

[GB84] E. Gafni and D. P. Bertsekas. Dynamic control of session input rates in communication networks. *IEEE Trans. on Automatic Control*, AC-29(11):804–823, November 1984.

[Hay81] H. Hayden. Voice flow control in integrated packet networks. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, Cambridge, MA, 1981.

[Jac88] V. Jacobson. Congestion avoidance and control. In *Proc. SIGCOMM*, pages 314–329, August 1988.

[Jaf81] J. M. Jaffe. Bottleneck flow control. *IEEE Trans on Communication*, COM-29,1(7):954–962, July 1981.

[Jai96] R. Jain. Congestion control and traffic management in atm networks: Recent advances and a survey. *Computer Networks and ISDN Systems*, 28,13:1723–1738, October 1996.

[JKG+95] R Jain, S. Kalyanaraman, R. Goyal, Sonia Fahmy, and Fang Lu. ERICA+: Extensions to the ERICA switch algorithm. Technical Report ATM-FORUM/95-1346, The Ohio State University, October 1995.

[JKV94] R. Jain, S. Kalyanaraman, and R. Viswanathan. The OSU scheme for congestion avoidance using explicit rate indication. Technical Report ATM-FORUM/94-0883, The Ohio State University, September 1994.

[JKV97] R. Jain, S. Kalyanaraman, and R. Viswanathan. The OSU scheme for congestion avoidance in atm networks: Lessons learnt and extensions. *Performance Evaluation, (North-Holland), Special Issue on Traffic Control in ATM Networks*, 31(1-2):67–88, November 1997.

[JKVG94] R Jain, S. Kalyanaraman, R. Viswanathan, and R. Goyal. Rate based schemes: Mistakes to avoid. Technical Report ATM-FORUM/94-0882, The Ohio State University, September 1994.

[JKVG95] R Jain, S. Kalyanaraman, R. Viswanathan, and R. Goyal. A sample switch algorithm. Technical Report ATM-FORUM/95-0178R1, The Ohio State University, February 1995.

[Kes91] S. Keshav. A control-theoretic approach to flow control. In *Proc. SIGCOMM*, pages 3–16, September 1991.

[KJF+97] Shivkumar Kalyanaraman, Raj Jain, Sonia Fahmy, Rohit Goyal, and Bobby Vandalore. The erica switch algorithm for abr traffic management in atm networks. Technical Report Submitted to IEEE/ACM Transactions on Networking, The Ohio State University, November 1997. http://www.cis.ohio-state.edu/ jain/papers/erica.htm.

[KVR95] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. An efficient rate allocation algorithm for atm networks providing max-min fairness. In *Proc. HPN*, September 1995.

[Mos84] J. Mosely. *Asynchronous Distributed Flow Control Algorithms*. PhD thesis, MIT, June 1984.

[MR91] A. Mankin and K. K. Ramakrishnan. Gateway congestion control survey. Technical Report RFC 1254, IETF, August 1991.

[RF95] A. Romanow and S. Floyd. Dynamics of TCP Traffic over ATM Networks. *JSAC*, 13(4):633–641, May 1995. An earlier version appeared in SIGCOMM '94, August 1994, pp. 79-88.

[RF99] K. K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ip. Technical Report RFC 2481, Experimental, IETF, January 1999.

[RJ90] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Trans. Comput System.*, 8(2):158–181, 1990.

[Rob94] L. Roberts. Enhanced PRCA (proportional rate-control algorithm). Technical Report ATM-FORUM/94-0735R1, ATM Systems, August 1994.

[Sat96] S. Sathaye. Atm forum traffic management specification version 4.0. Technical Report ATM-FORUM/95-0013R10, February 1996.

[ST94] K. Y. Siu and H. Y. Tzeng. Adaptive proportional rate control (APRC) with intelligent congestion indication. Technical Report ATM_FORUM/94-0888, University of California, Irvine, September 1994.

[Ste94] W. Stevens. *TCP/IP Illustrated*. Addison-Wesley, 1994.

[TW96] D. H. K. Tsang and W. K. F. Wong. A new rate-based switch algorithm for ABR traffic to achieve max-min fairness with analytical approximation and delay adjustment. In *Proc. 15th IEEE INFOCOMM*, pages 1174–1181, March 1996.

[Woj94] L. Wojnaroski. Baseline text for traffic management sub-working group. Technical Report ATM-FORUM/94-0394R5, October 1994.

[Zha89] L. Zhang. A new architecture for packet switching network protocols. Technical Report MIT/LCS/TR-455, MIT Laboratory for Computer Science, August 1989.

A Simulations of Phantom in a GFC

The configuration shown in Fig. 23 is called **GFC** (Generic Fairness Configuration). This configuration is taken from [Woj94] and is frequently used as an environment for testing flow control algorithms. The behavior of of Phantom in a Generic Fairness configuration is illustrated in Fig. 24.

B Simulations of Phantom in the presence of VBR

The simulation presented in Fig. 25 is for a one link that is shared by four ABR sessions and one VBR session. The rate of the VBR session is set as follows: Every 90 msec the average is chosen in random. During the period of time when the average is fixed the rate is rotating around this average. The rotation is achieved by random small changes.

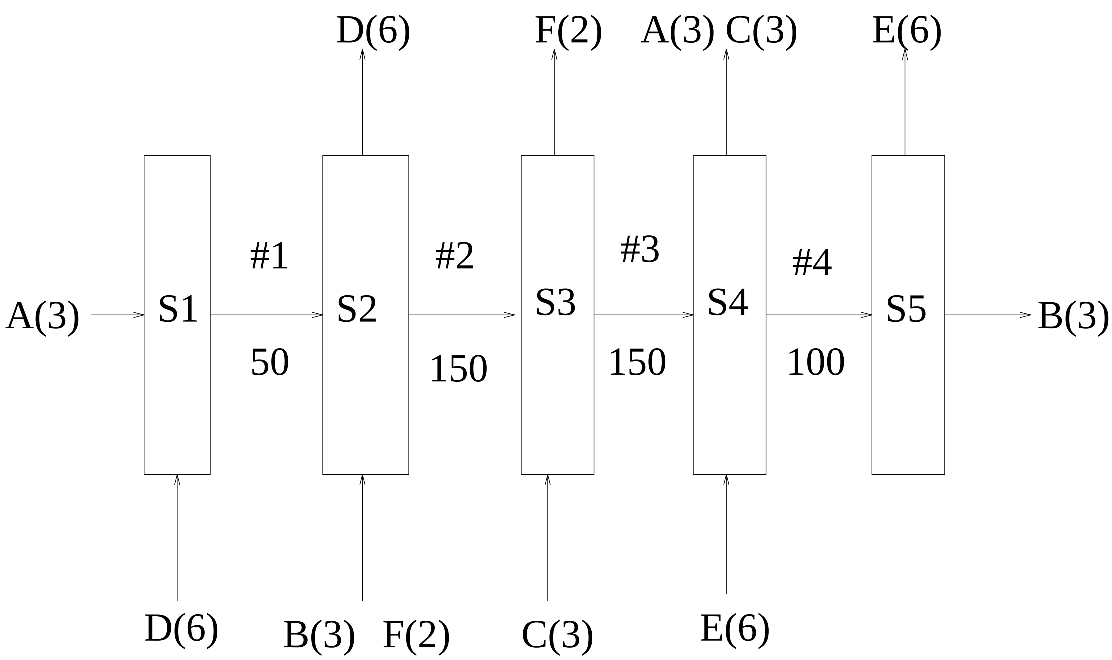
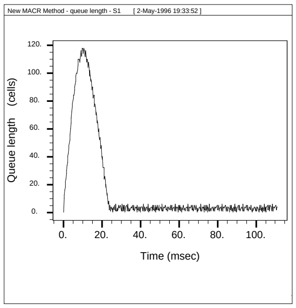
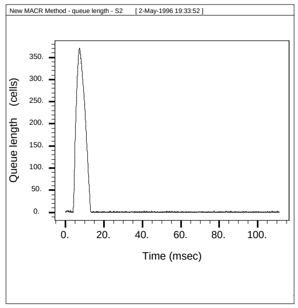


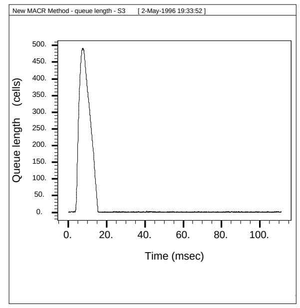
Figure 23: Description of the Generic Fairness Configuration. All link speeds are in Mbs. A(x) is the quantity of X VC's going to destination A. The FS (Fair Share) of link $\#1$ equals 5.56 Mbs as it restricts 9 sessions (A VCs and D VCs) while $FS_{Phantom}(\#1) = 5Mbs$. Similarly, as link $\#4$ restricts the B VCs and the E VCs then $FS(\#4) = 11.11Mbs$ while $FS_{Phantom}(\#4) = 10Mbs$. $FS(\#2) = \frac{150-50}{2} = 50Mbs$, $FS(\#3) = \frac{150-50}{3} = 33.33Mbs$, $FS_{Phantom}(\#2) = \frac{150-45}{3} = 35Mbs$ and $FS(\#3)_{Phantom} = \frac{150-45}{4} = 26.25Mbs$.



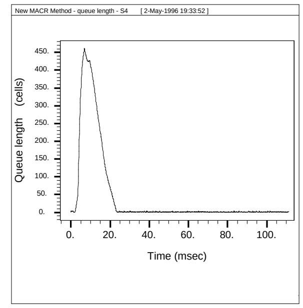
queue S1



queue S2

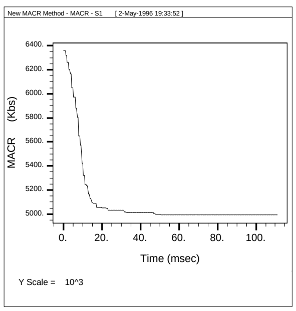


queue S3

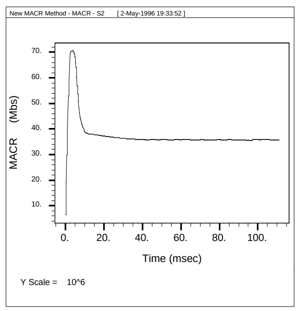


queue S4

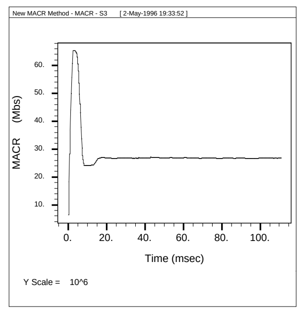
The value of the queue length in the different switches



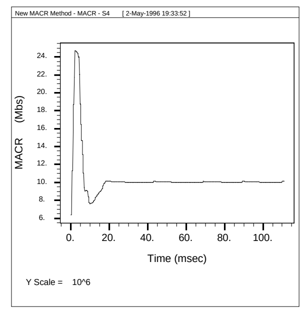
MACR S1



MACR S2



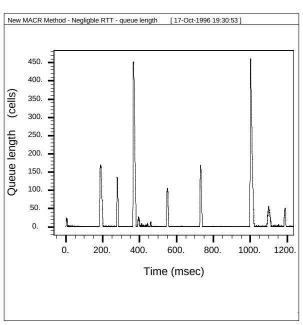
MACR S3



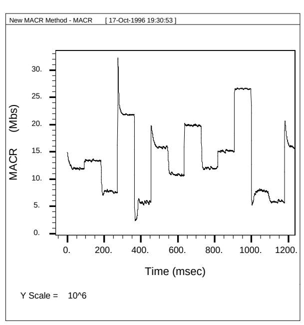
MACR S4

The value of MACR in the different switches

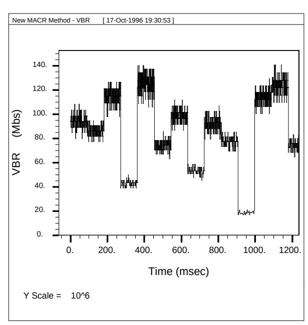
Figure 24: The behavior of Phantom in GFC configuration where $utilization\_factor = 1$ and the all links delays are negligible (0.003 ms). The queue length and the $MACR$ are measured in Switches S1-S4 (note that there is no flow control problem in Switch S5).



Queue length



MACR



VBR Session' rate

Figure 25: Phantom - Four greedy sessions with negligible RTT (0.01 ms) sharing a 150 Mbs link together with a VBR session.