MCA: A Rate-based End-to-end Multicast Congestion Avoidance Scheme

Jiang Li, Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180

***Abstract*—We propose MCA, a rate-based end-to-end multicast congestion *avoidance* scheme. Congestion avoidance [6] is different from congestion control in the sense that our scheme detects and responds to network congestion without necessarily inducing packet loss. Our scheme is a single-rate scheme and operates end-to-end, i.e., it goes at the rate allowed by the worst congested receiver and does not expect packet marking or other support from intermediate bottlenecks. Congestion is detected autonomously at receivers using the concept of *"accumulation"* and simple thresholding techniques proposed in our recent unicast work [8]. Congestion feedback to senders can be in the form of single-bit congestion indication (CIs) or as a multi-bit output rate measure. The feedback is sparse in the sense that at most one feedback is generated per measurement period (unlike multiple loss indications generated during packet loss). The source implements two key blocks: a filtering block to discriminate between competing feedback from receivers, and a congestion response block which implements a rate-increase/decrease policy. The two different feedback models (bit-based or explicit rate-based) leads to two different schemes: bit-based and explicit rate-based schemes. Simulation results show that both schemes avoid the drop-to-zero problem and are fair with unicast congestion avoidance schemes.**

I. Introduction

Multicast is the preferred transport mechanism for bulk data transfer to multiple receivers especially in multimedia applications and services on the internet. Applications like content distribution, streaming, multi-player games, multimedia multi-user chat/telephony, distance education etc could benefit from multicast. In this paper, we propose MCA, a rate-based end-to-end multicast congestion *avoidance* scheme. Congestion avoidance [6] is different from congestion control in the sense that our scheme detects and responds to network congestion without necessarily inducing packet loss. Our scheme is a single-rate scheme, i.e., it goes at the rate allowed by the worst congested receiver. Examples of single-rate multicast congestion control schemes include PGMCC [15], TFMCC [19], our earlier work LE-SBCC [16] and references within. These schemes are "congestion control" schemes in the sense that receivers wait for a packet loss which is signalled back to the source as loss-indications (LIs). If bottlenecks provide packet marking support (similar to TCP-ECN [14]), packet-loss may be avoided in the above schemes, and they could also be classified as "multicast congestion avoidance"schemes. Similar to other single-rate schemes, our scheme is aimed at small-medium scale receiver sets (up to 10,000 receivers). There exist another distinct class of congestion control schemes which are *multi-rate* (eg: RLC, FLID-DL [18], [5]).

This work was supported in part by ARO contract DAAD19-00-10559 and NSF ERC award EEC-9986821.

Our proposed scheme, MCA, uses a new concept of *"accumulation"* and simple thresholding techniques proposed in our recent unicast work [8] to achieve congestion avoidance on a *purely* end-to-end basis, i.e., with no marking support from interior bottlenecks. In this sense, our work is comparable to the unicast work of TCP Vegas [4] which assumes a similar model, albeit in a unicast, window-based TCP context. Just like TCP Vegas [12], [11], our scheme's congestion model detects congestion end-to-end as real queues are *being* built up. This model is inherently *incompatible* with the TCP model of *waiting for packet losses to occur* before detecting congestion. The only way to ensure compatibility is to have a packet marking scheme at bottlenecks which indicates congestion as the queues build up [10]. Therefore, in the absence of packet-marking support, our MCA scheme (like Vegas [12], [11]) *cannot directly compete* with TCP in the same queue and will be beaten down if it does. So, like Vegas, our scheme focusses on the conceptual rather than deployment issues. We discuss possible deployment scenarios briefly in the conclusion of this paper.

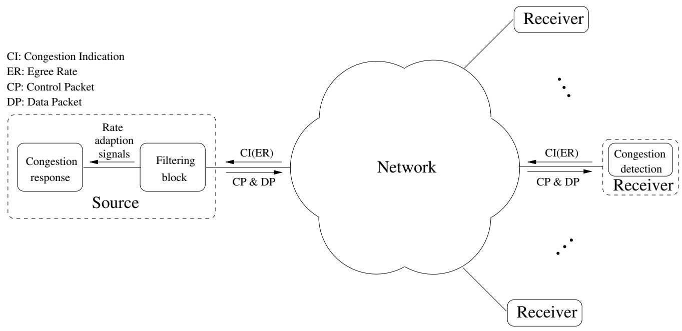


Fig. 1. MULTICAST CONGESTION AVOIDANCE: MODEL

MCA consists of three key building blocks (Figure 1): a *congestion detection* block at receivers, a *filtering block* at the source to discriminate between competing feedback from receivers, and a *congestion response* block which implements a rate-increase/decrease policy. Congestion detection is based upon our new "accumulation" measure. Congestion detection triggers feedback, which is sparse in the sense that at most one feedback is generated per measurement period (unlike multiple loss indications generated during packet loss in "congestion control" schemes). Congestion feedback to senders can be in the form of single-bit congestion indication (CIs) or as a multi-bit output rate measure. The two different feedback models (bit-based or explicit rate-based) leads to two different schemes: bit-based and explicit rate-based. These schemes essentially have different designs of the filtering and congestion response policy blocks. The explicit rate feedback can be leveraged to reduce the state requirements at the sender to O(1).

Simulation results show that both schemes avoid the drop-to-zero problem [19], [15], [2]. Drop-to-Zero is the problem of reacting to *more feedback* indications than necessary leading to a beat-down of the multicast flow's rate[19], [15], [2]. This occurs because the multicast flow receives feedback indications from multiple paths and may not filter them sufficiently. TCP-unfriendliness is the problem of reacting to *less feedback* than a hypothetical TCP flow would on the worst loss path [3], [15], [19]. Though the congestion detection model is incompatible with that of TCP (and we cannot directly demonstrate fairness with TCP) we demonstrate fairness with similar *unicast* congestion avoidance flows.

II. MCA: Scheme Description

We will propose two different multicast congestion avoidance schemes in this section. The bit-based scheme will be referred as *Bin-CI* scheme and the explicit rate-based as *ER-CI* in the following context.

We start the section describing the concept of "accumulation" using a fluid flow model, and develop an algorithm for measuring it in a multicast context at receivers. The accumulation concept is the basis of congestion detection in both schemes.

A. Accumulation

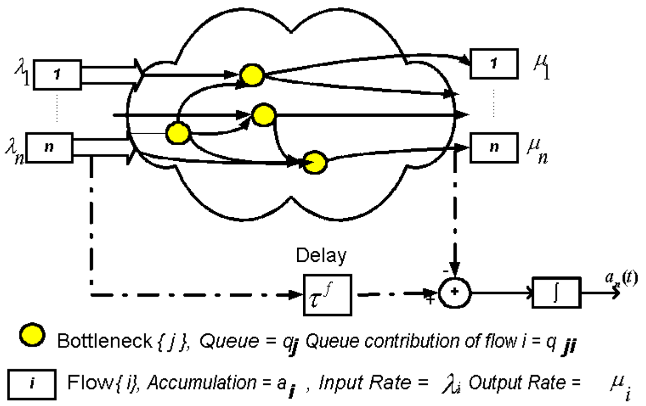


Fig. 2. FLUID FLOW MODEL: ACCUMULATION CONCEPT

We propose to use the concept of *accumulation* as a basic tool in our schemes. This concept was first developed in our earlier unicast work [8]. We summarize the core ideas here. This discussion below assumes unicast fluid flows, but we extend it to multicast later in this section.

Consider a network of queues fed by a system of *unicast* fluid flows { i } with input rates $\lambda_i(t)$ at the ingress, and output rates $\mu_i(t)$ at the egress (Figure 2). Assume infinite buffers at the bottlenecks. Let $\lambda_{ji}(t)$ be the input rate, and $\mu_{ji}(t)$ be the output rate of flow i w.r.t the j^{th} bottleneck at time t. The queue contribution of flow i at bottleneck j at time t, $q_{ji}(t)$ follows the dynamics:

$$\dot{q}_{ji}(t) = \lambda_{ji}(t) - \mu_{ji}(t)$$

Define the total accumulation of flow i, $a_i(t)$ by the dynamics:

J : The total number of bottleneck on the path of flow i.
d_r : The propogation delay between r^{th} and $(r+1)^{th}$ bottleneck. $d_J = 0$.

$$\dot{a}_i(t) = \sum_j \dot{q}_{ji}(t - \sum_{r=j}^{J} d_r) = \lambda_i(t - \sum_{r=0}^{J} d_r) - \mu_i(t) \qquad (1)$$

Here $\sum_{r=0}^{J} d_r = \tau_i^f$ is the forward propagation delay for flow i. Now, making some minor assumptions and discretizing Equation (1) as described in [8] we obtain:

$$a_i(k) = a_i(k-1) + (\lambda_i(k) - \mu_i(k))\tau \qquad (2)$$

In other words, accumulation for a loop is the *amount of fluid sent into the loop minus the amount of fluid leaving the loop measured over a correlated interval of time*. It can be measured by using correlated periods τ, staggered only by the fixed one-way delays. This can be done by sending synchronization data "*out-of-band*," i.e., the synchronization data experiences only the fixed one-way delays and not the queueing delays.

Given this assumption, accumulation also satisfies a *"per-loop"* property of being related to the *output rate* μ_i of that loop. Observe that the sum of all *output rates* cannot be larger than the sum of capacities of the network, a nice related *global property* which does not hold for the sum of all *input rates*.

Now, it can also be shown that:

$$a_i(k) > 0 \Leftrightarrow \exists j \;\; q_j(k) > 0, \text{ and } a_i(k) = 0 \Leftrightarrow \forall j \;\; q_j(k) = 0 \qquad (3)$$

In other words, for a network of fluid flows *a zero-threshold for the per-loop accumulation measure is equivalent to a zero-threshold on the real queue at some bottleneck*. Also, the *sum of all the per-loop accumulations is the sum of all queueing in the network*, another nice global property. These properties are rigorously developed in reference[8].

In summary, accumulation and output rate are quantities which can be *measured with only per-loop information*. But these measurements satisfy important global properties which allow us to build a *fully* distributed, transparent closed-loop building block using them. In particular, a simple congestion avoidance approach would be:

a) use simple thresholding techniques on the accumulation measure to detect epochs of congestion
b) use binary or output rate measures to guide the congestion response policy to achieve fine-grained control over input rate dynamics.

While the above discussion referred to unicast, the same approach can be applied to multicast if the machinery for accumulation measurement can be instrumented, which is the focus of the following sub-sections.

B. Accumulation Measurement

To perform accumulation measurement in real world, we relax three key assumptions made in the previous section. First, we send synchronization (or control) data "*in-band*" instead of "*out-of-band*", i.e., it sees both fixed delays and queuing delays. Second, we send packets instead of fluid. Therefore, to account for the randomness introduced, the thresholding procedure has to be amended and a new re-synchronization procedure is performed at the end of congestion periods. Third, we develop the scheme for multicast, i.e., divide functionality between the

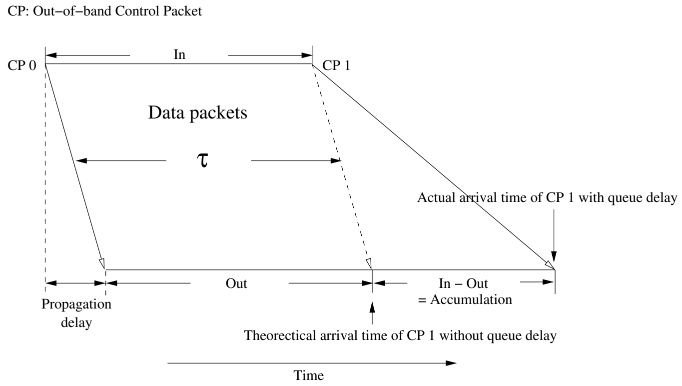


Fig. 3. ACCUMULATION W/ IN-BAND CONTROL PACKETS

source and receivers such that the reverse control traffic is minimized. Forward control traffic is multicast to all receivers.

Figure 3 illustrates the measurement of accumulation using *in-band* control packets. Assume that the first control packet (CP0) sees no queue (and hence only fixed delays). The second control packet (CP1) is multicast after the measurement period τ, which contains the count of bytes sent (*"in"*). The receiver also measures a variable *"out"* which is the number of bytes seen from the receipt of CP0 for a period τ. If there were no queuing delays anywhere, CP1 should have arrived at this point. Therefore the number of bytes *in - out* is the accumulation, which is measured at the time of receipt of CP1. Observe that this measure works correctly in a rate-based system where packets are sent uniformly and the input burstiness is also control by a rate-shaper.

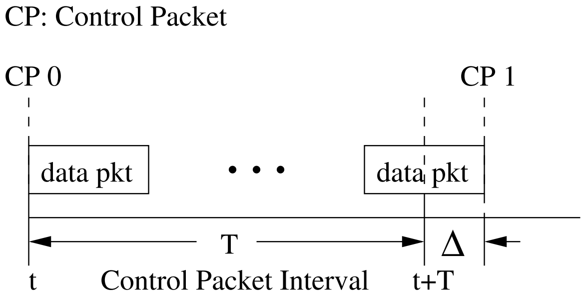


Fig. 4. ERRORS IN ACCUMULATION DUE TO PACKETIZATION

Figure 3 still assumed a fluid model. Packetization introduces randomness and burstiness in the system. In particular, Figure 4 shows that when the measurement period τ expires, a packet may be in transmission which leads to a delay in the transmission of the control packet (CP). On the average this can lead to a measurement error in both *"in"* and *"out"* of 0.5 packets each, even if the size of the CP is accounted for. (In fact, this is also a side-effect of using *"in-band"* CPs.) Moreover, in a packetized system the rate control at the source is done using leaky bucket shapers which can have a minimum bucket depth of one packet. Hence, even for a perfectly smoothed packetized transmission, bottlenecks can have a steady state *average* queue of one packet when underloaded. Therefore, the fluid flow equation (3) which implied a zero-threshold on accumulation no longer holds.

However, we can use the following hysterisis technique: detect congestion epochs if accumulation goes larger than two packets (H_thresh), and subsequently declare end of congestion epoch when accumulation falls below 1.5 packets (L_thresh). This technique essentially adds the average steady state errors in accumulation measurement (0.5 + 0.5 + 1pkt) and sets the congestion detect threshold above the sum of these errors. If there are other sources of noise which affect accumulation (eg: scheduling noise at operating systems or at bottlenecks), the thresholds should be set higher. This technique is hence conservative in detecting congestion, i.e., a receiver may unilaterally detect congestion even if there is no network congestion (eg: in multi-bottleneck cases). Higher thresholds reduce the probability of such errors, at the price of larger worst-case queues. We find through simulation that the setting above works very well.

Another aspect of packetization is that since bottlenecks have steady state queue even in underload, the initial control packet (CP0 in Figure 3) may not see zero queuing delay. In other words, our assumption of synchronization at the first control packet may be erraneous. Also, as a side effect of the hysterisis scheme described above, the receiver could end a congestion epoch with a non-zero accumulation. To counter these issues, we introduce the notion of *"re-synchronization"* as illustrated in Figure 5. We begin with the default assumption that we have synchronized correctly at CP0, and then measure successive intervals of length τ based upon this assumption. If control packets arrive at the receiver before the corresponding interval timer expires, then we re-synchronize (not shown in the figure) and set accumulation to zero. Also, if we detect end of congestion epoch, we re-synchronize.[1] Figure 5 shows a case when the re-synchronization happens perfectly, i.e., accumulation is zero, and the re-synch point is the same as the theoretical arrival of the control packet. In practice, any residual positive accumulation is carried over to the next epoch. The pseudo-code of the accumulation measurement and CI generating algorithm is presented below.

C. Accumulation Measurement and CI Generating Algorithm

Suppose we begin at time t_0. Let T be the value of control packet interval. The behavior of the source is simply to send out a control packet (CP) to the receivers at $t_0 + iT$ ($i = 0, 1, 2...$). The receivers execute the following algorithm whenever a CP arrives, with the variable $accu$ recording the accumulation:

t	:	current time
i	:	the sequence number of CP
T	:	control packet interval
t_s	:	time of the most recent synchronization point (SP)
seq_s	:	CP sequence number of the most recent SP
$accu$	:	accumulation in bytes
$accu_g$	:	global accumulation in bytes
H_thresh	:	high threshold of accumulation.
L_thresh	:	low threshold of accumulation.

(*Synchronization point (SP)* is the point at which we assume no packet backlog on the path from the source to the receiver.)

1. If the CP is the very first one since t_0,
   Set: $accu_g = 0$, $accu = 0$, $t_s = t$, $seq_s = i$. (SP)
   Return.
   Endif

[1] Believing that the periods between synchronizations won't be long, we assume that clock skew can be ignored.

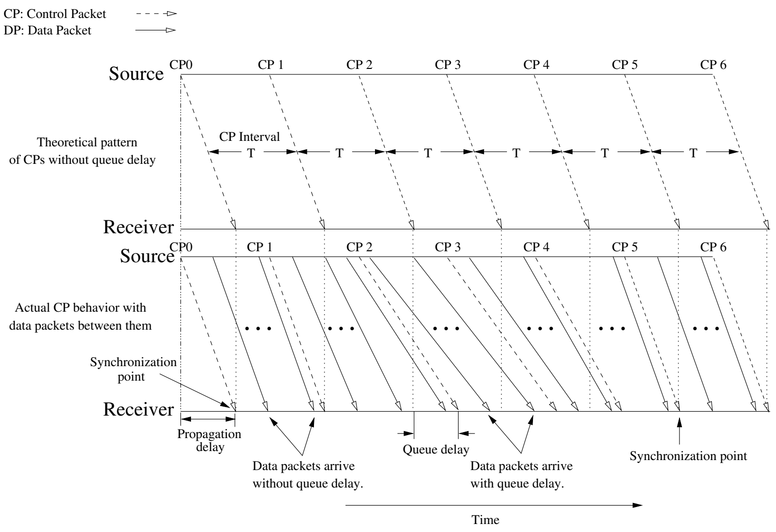


Fig. 5. CONGESTION EPOCHS: SYNCHRONIZATION POINTS AND ACCUMULATION

2. If $t < t_s + (i - seq_s)T$,
   Set: $accu_g = 0$, $accu = 0$, $t_s = t$, $seq_s = i$. (SP)
   Return.
   Endif
3. Set $accu$ = the bytes received within $(t_s + (i - seq_s)T, t] + accu_g$.
4. If $accu \geq H\_thresh$,
   Send a CI back to the source.
   Else if $accu > L\_thresh$,
   Do nothing.
   Else if $accu$ has ever exceeded H_thresh since t_s,
   Set: $accu_g = accu$, $accu = 0$, $t_s = t$, $seq_s = i$. (SP)
   Else
   Do nothing.
   Endif
   Return.

The algorithm above assumes zero packet loss. To make it robust, a receiver also send CIs upon detection of packet losses. Although packet losses may decrease the accumulation seen by the receiver and hide congestion, the congestion detection by packet losses compensate for it. Besides, the error of $accu$ measured at the arrival of i^{th} CP won't be carried over to next measurement if the receiver does not see any loss during $(t_s + (i + 1 - seq_s)T, t]$. In addition, at the arrival of i^{th} CP, if the receiver has seen any losses during $(t_s + (i - seq_s)T, t]$, it won't do re-synchronization.

However, if the route between the source and the receiver changes to be longer (although it does not happen too often), our scheme won't be able to re-synchronize. That is an issue for our future research.

D. Bin-CI Scheme

We now describe filtering and congestion response policies at the source. If the feedback upon congestion detection is a single bit, i.e., a binary congestion indication (CI), the scheme is called *Bin-CI*. We shall see that the tradeoff between explicit rate feedback and single bit feedback is simplicity of feedback vs complexity of state at the source. For this *Bin-CI* scheme, we largely leverage our prior work (LE-SBCC) [16] for a similar case of binary feedback carrying packet loss indications (LIs) instead of congestion indications (CIs). There are a cascade of three filters (CI2CE, MaxLPRF, ATF) into which CIs are fed, and two modules of RTT estimation and rate adaption, as follow.

CI2CE FILTER Whenever a CI from receiver i arrives, the source checks the current time t_2. Let t_1 be the time when last CI from i was accepted. If $t_2 - t_1 \leq RTT + 2 \times \sigma$ [2], the CI is rejected. Otherwise, it is accepted as a new congestion event (CE) from receiver i and passed to the next filter MaxLPRF.

MaxLPRF FILTER Let the total number of CEs from receiver i be X_i. Any CE is passed with probability of $(\max X_i)/\sum_i X_i$, i.e., the MaxLPRF passes on the average, $\max X_i$ CEs out of a total of $\sum_i X_i$ CEs.

In addition to the above probabilistic behavior, MaxLPRF maintains two accounting variables: P_d and V_d. P_d is set zero at initialization and *each time the rate is reduced in the rate adaptation module (see below)*. Whenever a CE arrives, P_d is incremented by $(\max X_i)/\sum_i X_i$. If P_d was below 1 prior to incrementing and is at least 1 after incrementing, we set V_d as the current data transfer rate. These accounting variables are used in the rate-adaptation module (see below).

ATF FILTER When a CE arrives at ATF, the current time t_2 is checked against the time t_1 when last CE (from any receiver) was passed by ATF. The new CE is passed if $t_2 - t_1 \leq RTT + 4 \times \sigma$. This guarantees that at most one rate deduction is performed in any one RTT.

RTT ESTIMATION At the arrival of a CI, (1) if it is triggered by a CP, the RTT sample RTT_s is the difference between the current time t and the departure time of the CP triggering the CI, (2) if it is triggered by a packet loss and not a retransmitted one, RTT_s is the difference between t and the transmit time of the lost packet. With RTT_s, the RTT is updated as $RTT = 7/8 \cdot RTT + 1/8 \cdot RTT_s$.

RATE ADAPTATION During the periods of no congestion (i.e. no CE), the data transfer rate V_s is incremented by $S/SRTT$ every $SRTT$ [3] (where S is the data packet size). An exponentially weighted moving average (EWMA) of the rate-increments, V_e is maintained as $V_e = \alpha V_e + (1-\alpha)S/SRTT$ [4].

[2] σ is the mean deviation of RTT samples.
[3] $SRTT = RTT + 2 \times \sigma$
[4] α is $7/8$ in our simulations.

If a CE passes the filter cascade, the rate V_s is adjusted in the following way:

1. If $P_d < 1$, $V_s = \beta(V_s - V_e)$.
2. If $P_d \geq 1$, $V_s = \beta(V_d - V_e) - (V_s - V_d)$.

In our simulations, β is 0.9. When $P_d < 1$, V_s is deducted by the amount of V_e first, and then multiplicatively reduced by β. This ensures that, with a high probability, drain capacity is provisioned for the accumulation incurred. When $P_d \geq 1$, it means that the source responds late (since the ATF may have filtered a CE passed by MaxLPRF). Therefore, the source goes back to the rate V_d where it should have been, responds as described earlier, and cancels excessive increment due to late response.

E. ER-CI Scheme

The *ER-CI* scheme leverages the multi-bit egress rate (ER) information in the feedback message (also called as a CI for convenience) to reduce the state requirements at the source to O(1). The filtering block in the *ER-CI* scheme calculates an EWMA of the ER fed back by receivers. The EWMA estimate $V_{fe} = \gamma V_{fe} + (1-\gamma)V_f$ where V_f is the egress receiving rate in the feedback message. An ER error estimate σ_v is also calculated: $\sigma_v = \gamma\sigma_v + (1-\gamma)|V_{fe} - V_f|$ [5]. Let R be the the receiver whose CI was accepted most recently. An arriving CI is accepted if any of the following conditions is met:

(1) The CI is the very first one received by the source,
(2) The CI is from receiver R,
(3) V_f satisfies $V_f < V_{fe} - 3\sigma_v$,
(4) $t_2 - t_1 > RTT + 7\sigma$, $V_s \geq V_{fe} + 3\sigma_v + S/SRTT$,
where V_s is the data transfer rate, S is packet size,
t_2 is the current time,
t_1 is the time when last CI was accepted.

When a CI is accepted, the source data transfer rate V_s is updated as $V_s = min(V_s, \beta V_f)$, $(\beta < 1)$. The decrease factor β is the same as that of the *Bin-CI* scheme. If there is no congestion detected or the CI is filtered by the algorithm above, V_s grows by $S/SRTT$ every $SRTT$.

The filter block is required because too many rate reductions $(V_s = min(V_s, \beta V_f))$ would eliminate the chance of rate increment and hence beat-down a multicast flow having many branches. The filter block attempts to keep track of only one of the receivers seeing the smallest egress rate (ER). Condition (1) is for initialization. Condition (2) means that if the receiver R whose CIs the source accepted most recently keeps sending CIs, the source will always accept them. To avoid neglecting other receivers, condition (3) accepts feedback from other receivers provided that the ER fed back $V_f < V_{fe} - 3\sigma_v$, i.e., the ER fed back is statistically significant. Condition (4) is a statistical safeguard against the case of the receiver R disappearing, while the other feedback rates remain within the range $[V_{fe} - 3\sigma_v, \infty)$. Condition 4 implies that the source has not seen CIs from the receiver R for a long enough period $(RTT + 7\sigma)$, as well as that the transfer rate has been increased significantly $(V_s \geq V_{fe} + 3\sigma_v + S/SRTT)$ and congestion feedback is being received from some other receiver. The number 7σ comes from a calculation excluded here for simplicity.

[5] γ is $7/8$ in our simulations.

III. Simulation Results

We ran several ns-2 simulations to verify the performance of our scheme. The simulations include (1) Simple Multicast Configuration, (2) Multiple Bottlenecks (Linear Network), (3) Drop-to-Zero Avoidance Testing. In these simulations, the data packet size is 1000 bytes. The bottleneck buffer size is 1MB which is sufficient to avoid any packet losses in all simulations. Queue graphs show that the real queue is far smaller. When the bottleneck buffer size is smaller, there can be packet losses while our scheme still performs well. For space reasons, we do not present those results here.

A. Simple Multicast Configuration

Consider the simple multicast topology in Figure 6. At time=0, there is only one multicast flow, with the source on Node 1, two receivers on Node2 and 3 respectively. At t=30s, another multicast flow is added with the same source-receiver set. At t=60s, the third multicast flow is added, again with the same source-receiver pattern.

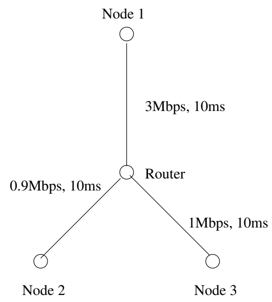


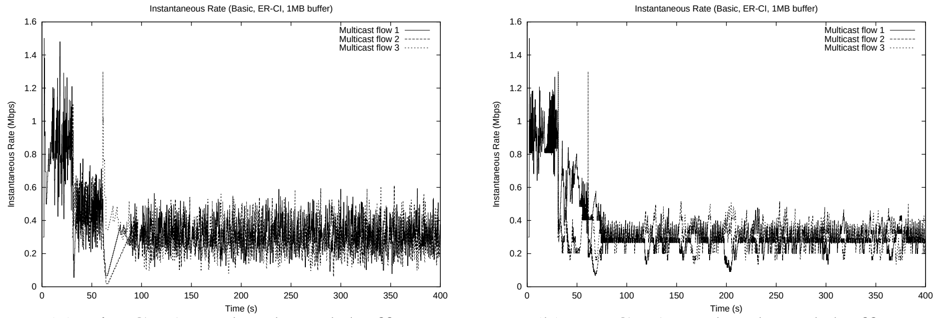
Fig. 6. SIMPLE MULTICAST CONFIGURATION

Figures 7(a) and (b) shows the performance of *Bin-CI* and *ER-CI* respectively. In both cases, observe that for t in [0s,30s], the rate oscillates around 0.9 Mbps. For t in [30s, 60s] the rates are around $0.9/2 = 0.45$ Mbps showing that the two multicast flows compete fairly. For $t > 60s$, the rates oscillate around $0.9/3 = 0.3$ Mbps, again with the rates being shared fairly.

The queue sizes of different simulations are shown in Figure 8 and the utilization/queue data is summarized in Table I. We can observe that the bandwidth utilization is high (above 80%) while the average queue length is low (up to 20 packets). Also observe that the utilization in the *ER-CI* scheme is in general higher than that in the *Bin-CI* scheme.

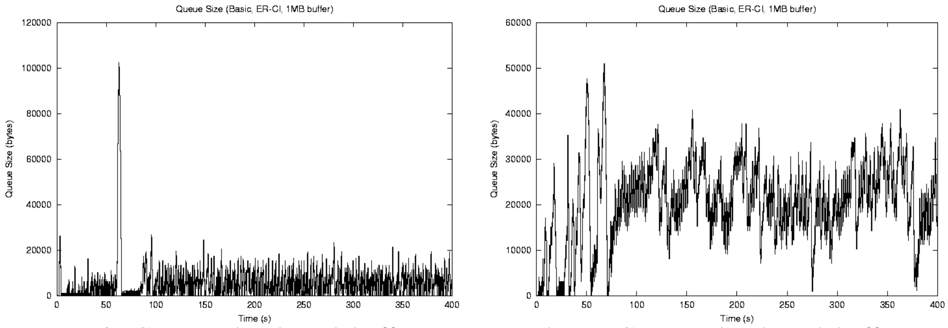
B. Multiple Bottlenecks: Linear Network

The linear network is a popular multi-bottleneck configuration [8]. We extend this configuration for multicast as shown in Figure 9. There are three flows running on the configuration of Figure 9. One multicast flow goes from Node 1 to Node 4 and 5, two single-receiver multicast flows go from Node 2 to Router 2 and from Node 3 to Node 4 respectively. In this configuration, if the flow that traverses multiple bottlenecks gets a larger share, it reduces the overall network capacity. Different notions of fairness define how much the long flow can get. *Proportional fairness* implies that the the long (multicast) flow should get one-third of the bottleneck bandwidth whereas *Max-min fairness* suggests a share of one-half.



(a) Bin-CI, 1MB bottleneck buffer (b) ER-CI, 1MB bottleneck buffer

Fig. 7. SIMPLE MULTICAST CONFIG: RESULTS (RATES)



(a) Bin-CI, 1MB bottleneck buffer (b) ER-CI, 1MB bottleneck buffer

Fig. 8. SIMPLE MULTICAST CONFIGURATION: RESULTS (QUEUES)

TABLE I

AVERAGE QUEUE SIZE AND UTILIZATION

	Average Bottleneck Queue Size (bytes)	Bottleneck Utilization
Bin-CI, 1MB bottleneck buffer	5745.93	83.0221%
ER-CI, 1MB bottleneck buffer	20359.8	98.0692%

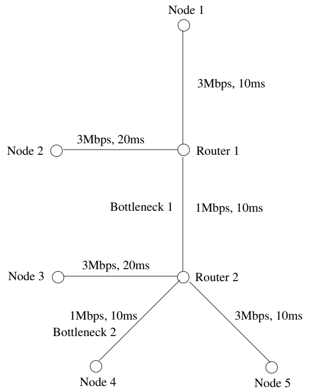


Fig. 9. MULTIPLE BOTTLENECKS: LINEAR NETWORK

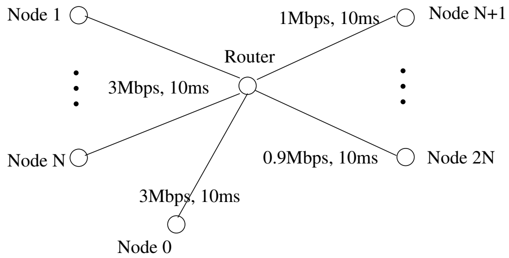


Fig. 11. STAR TOPOLOGY CONFIGURATION

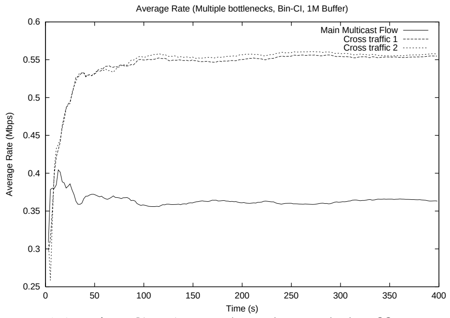
The average rate [6] graph (Figure 10) shows that the multicast flow gets more than 1/3 of the bottleneck bandwidth (the proportional fairness share), but less than 1/2 of the bottleneck bandwidth (the max-min fairness share).

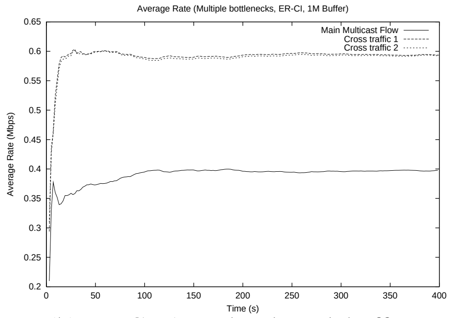
C. Star Topology: Drop-to-Zero Avoidance Testing

To test the immunity of the scheme to the drop-to-zero (DTZ) problem [15], we use the star topology (Figure 11). The DTZ problem occurs when multiple paths in a multicast tree experience different congestion levels asynchronously, and the source reacts to more congestion feedback than necessary leading to a beat-down of transmission rate.

In the star topology, node i sends data to Node $i + N$ ($i = 1..N$), $N = 16$, thus generating background traffic on each path. A multicast flow has as its source, Node 0, and receivers, Node $N + 1$ to $2N$. The links between Router and Node j ($j = N + 1..3N/2$) have bandwidth of 1Mbps; and the links between Router and Node j ($j = 3N/2 + 1..2N$) have bandwidth of 0.9Mbps. The multicast flow should compete fairly with those single-receiver flows on 0.9Mbps bottlenecks. Indeed, figure 12 shows that the multicast and the uni-

[6] Average rate = (amount of data sent between time 0 and t) / t, where t is the sampling time.

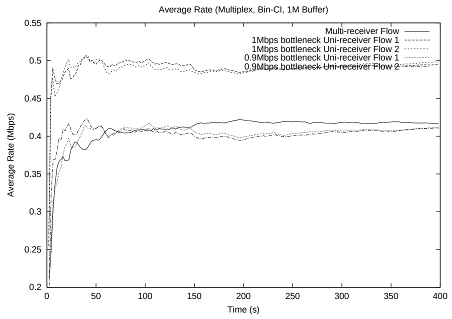




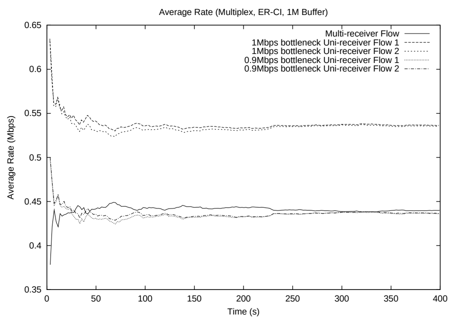
(a) Bin-CI, 1MB bottleneck buffer (b) ER-CI, 1MB bottleneck buffer

Fig. 10. LINEAR NETWORK: RESULTS (AVG. RATES)

cast flows sharing the 0.9Mbps bottleneck achieve equal rates around [0.4,0.45]Mbps. This demonstrates fairness and the drop-to-zero immunity of the schemes.



(a) Bin-CI, 1MB bottleneck buffer



(b) ER-CI, 1MB bottleneck buffer

Fig. 12. STAR CONFIGURATION: 16 FLOWS MULTIPLEXED (AVG. RATES)

IV. CONCLUSION

We have proposed two schemes (*Bin-CI* and *ER-CI*) for rate based multicast congestion *avoidance*. Both schemes leverage the concept of *accumulation* developed in our recent work. By accumulation measurement extended to multicast, receivers detect congestion without necessarily inducing packet loss and send congestion indications (CIs) back to source for the purpose of rate control. The source then filters the CIs and adapt its transfer rate according to rate control policy.

While the *Bin-CI* requires only binary congestion indication (CIs) from receivers and needs O(N) state complexity at the source, the *ER-CI* requires explicit output rate in CIs but only maintains states of O(1) at the source. Both schemes work very well in our simulations, i.e. they (1) do not suffer from drop-to-zero problem, and, (2) achieve high bottleneck utilization and low average queues, (3) are proportionally fair. Although our results focus on lossless interior network, we have extended our schemes to be robust to packet losses. We expect the deployment scenarios for such schemes to be at ISPs who can control their infrastructure (i.e. can manage buffers, isolate non-congestion avoidance flows) and want to gain efficiencies due to multicast on an edge-to-edge basis.

REFERENCES

[1] S. Bhattacharyya. et al, "A Novel Loss Indication Filtering Approach for Multicast Congestion Control," *J. of Comp. Commns*, Feb '01.
[2] S. Bhattacharyya, D. Towsley and J. Kurose, "The Loss Path Multiplicity Problem in Multicast Congestion Control," *INFOCOM*, March '99.
[3] S. Bradner et al, "IETF criteria for evaluating reliable multicast transport and application protocols," *RFC 2357*, June '98.
[4] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE JSAC,* Vol 13, No. 8, Oct 1995
[5] J.W. Byers, et al, "FLID-DL Congestion Control for Layered Multicast," *NGC*, Nov '00.
[6] D. Chiu and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks," *Journal of Computer Networks and ISDN*, Vol. 17, No. 1, June 1989, pp. 1-14
[7] S. Floyd et al, "Equation-Based Congestion Control for Unicast Applications," *SIGCOMM*, Aug '00.
[8] D. Harrison, S. Kalyanaraman, S. Ramakrishnan, "Overlay Bandwidth Services: Basic Framework and an Edge-to-Edge Closed-Loop Building Block," preprint, 2001. Available from: http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html
[9] V. Jacobson, "Congestion avoidance and control," *SIGCOMM*, Aug '88
[10] S.H. Low, "A Duality Model of TCP and Queue Management Algorithms," *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, Sep 2000, Monterey, CA.
[11] S.H. Low, L.L. Peterson, and L. Wang, "Understanding Vegas: A Duality Model," *Proceedings of ACM SIGMETRICS*, Boston, MA, June 2001
[12] J. Mo, et al, "Analysis and comparison of TCP Reno and Vegas," *INFOCOM*, Apr '99.
[13] J. Padhye et al, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *SIGCOMM*, Aug '98.
[14] K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," *RFC 2481*, Jan '99.
[15] L. Rizzo, "PGMCC: A TCP-friendly Single-Rate Multicast Congestion Control Scheme," *SIGCOMM*, Aug '00.
[16] P. Thapliyal, Sidhartha, J. Li, S. Kalyanaraman, "LE-SBCC: Loss-Event Oriented Source-based Multicast Congestion Control," *Multimedia Tools and Applications*, 2002, to appear. Available from: http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html
[17] T. Speakman et al, "PGM reliable transport protocol specification," *Internet Draft*, March '00.
[18] L. Vicisano, L. Rizzo and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," *INFOCOM*, Apr '98.
[19] J. Widmer, M. Handley, "Extending Equation-based Congestion Control to Multicast Appliations," *SIGCOMM*, Aug 2001