# Internet Multicast Congestion Control: A Survey *

Y. Richard Yang Simon S. Lam

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188

## Abstract

*The increasing popularity of group communication applications such as teleconference and information dissemination services has led to a great deal of interest in the development of multicast transport protocols layered on top of IP multicast. However, these new transport protocols could cause congestion collapse if they are widely used but do not provide adequate congestion control. In this paper, we evaluate system models for multicast congestion control, and identify three key problems:* feedback implosion, congestion indicator filtering *and* fairness*. Current approaches to solve these problems are then discussed. Based on an analysis of the current approaches, we propose an algorithm to solve the* truncated TCP problem *to improve upon the current best approach.*

## 1 Introduction

Multicast improves the efficiency of multipoint data distribution by building a distribution tree from a sender to a set of receivers [5]. The increasing popularity of group communication applications such as teleconference and information dissemination services has led to a great deal of interest in the development of multicast transport protocols layered on top of IP multicast. However, these new transport protocols could cause congestion collapse if they are widely used but do not provide adequate congestion control. The success of the Internet relies on the fact that TCP sessions respond to congestion by reducing their load presented to the network. Therefore, the IETF reliable multicast criteria [17] require each multicast transport protocol proposal to include an analysis of whether the protocol has congestion avoidance mechanisms strong enough to cope with deployment in the global Internet.

The congestion control component of a transport protocol has two main objectives [8]:

- Avoid congestion collapse — A network congestion collapse occurs when the network is increasingly busy, but little useful work is getting done. Three scenarios that cause congestion collapse have been identified: unnecessarily-retransmitted packets [19, 14], fragmentation [15, 18], and packets that are discarded before they reach their receivers. To solve the third case, the sender must respond to the congestion and stop sending packets that cannot get through the network.

- Achieve fairness with competing traffics — There are many possible ways to define fairness [16, 1]. One popular notion of fairness is *max-min* [1] fairness. Another type of fairness definition is *global fairness*. Under this definition, each entity has an equal claim to the network's scarce resources (e.g., an entity traversing $N$ congested links is using more scarce resources than an entity traversing one congested link).

Several multicast congestion control protocols have been proposed recently. We can classify the approaches into 3 categories: single rate, replicated stream and layered. Single rate approach will send at one rate to the whole group. Under replicated approach, receivers will be partitioned into groups and each receiver joins one group. In layered approach, the data stream is organized in an incremental way, and a receiver incrementally joins higher groups according to its available bandwidth. In this paper, we will concentrate on the single rate approach because we consider it as a basic problem all three approaches have to solve.

The single rate multicast congestion controls proposed so far are Representative [6], LTRC [25], RLA [26], TFMCC [12], MTCP [24], Golestani [11]. We think it will be instructive to study the key problems and the current solution approaches. Based on this study, we will describe our own approach.

In particular, we want to answer the following questions:

- What are the feedbacks the sender receives? And how is the feedback implosion problem solved?

- What are the congestion indicators? How is the congestion indicator filtering problem solved?

- What kind of control parameter do they use: rate, or window? What is the control parameter estimation algorithm? Where the control parameter estimation algorithm is placed?

*Research sponsored in part by National Science Foundation grant no. ANI-9977267.

- What kind of fairness can they provide with respect to co-existence with TCP?

Multicast congestion control is still a new and active research area. Comprehensive performance evaluation of the current proposed approaches needs to be done to get a deeper understanding of the solution approaches. Our intention is to explore the similarities and differences between the approaches, and make our proposal to solve this difficult and important problem. It is important to emphasize that which approach to implement depends upon a number of variables, such as fairness requirements, network topology, and implementation complexity.

The organization of the paper is as follows. In the next section, we discuss the multicast congestion control system model. In section 3, we discuss the key problems that make multicast congestion control difficult and the current proposed solutions. A summary of the result is also presented at the end of this section. In section 4, we discuss the *truncated TCP problem*, and present our solution approach. We conclude the paper in section 5.

## 2 System Model

According to control theory, there are two types of control systems: feedback and open-loop. In a feedback control system, the result of the control is measured and the control parameter is adjusted on the fly. In an open-loop control system, a pre-determined control strategy is fixed without adjustment on the fly. To implement open-loop control in the Internet without causing congestion collapse, especially congestion collapse caused by packets discarded before they reach their receivers, we need a *virtual circuit* style of guarantee. This requires a session to make resource reservations ahead of time subject to admission control. It can then control its sending rate within the reservation, and does not need to respond to changing network conditions. However, the current Internet provides best effort service. Quality of service reservation using RSVP [28] has not been widely deployed. Without service reservation, open-loop congestion control is difficult to implement. Thus, up to date, all multicast congestion control proposals are based upon feedback control.

Figure 1 shows our model of the feedback congestion control system.

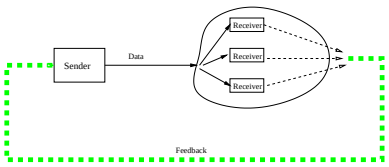


**Figure 1. Congestion Control System Model**

A sender congestion control component accepts data from its upper layer. It then uses its control parameter $c$ to decide if it can put more data into the network. If so, it sends data to the receivers using IP multicast. Since IP multicast will build a distribution tree to forward the data, only one copy of the data will traverse the path between the sender and any receiver. Therefore, we use a thin solid line to represent each data path. Receivers will send feedbacks to the sender. Feedback paths are represented using a thick dashed line in this model to show the possibility of multiple copies of feedback. The sender will change its control parameter $c$ according to the feedback.

The control parameter $c$ should be adjusted according to network traffic conditions. We call the algorithm to estimate $c$ the *estimation algorithm*. The sender should update $c$ upon receipt of each new estimaiton.

Figure 2 shows a model of the estimation algorithm. Overall, there are 4 important design parameters for the

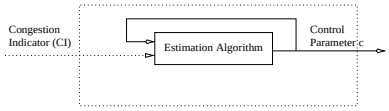


**Figure 2. Estimation Algorithm System Model**

control system:

1. *Control parameter c.* The parameter can be either window or rate. If it is a window, the sender can send new data into the network whenever there is space in the window. Otherwise, if the parameter is rate, the sender can inject new data no higher than the rate. The advantage of window control is that it integrates naturally with error control and flow control. Also, window control reacts to network changes faster [11]. TCP is an example that uses window as control parameter. The advantage of rate control is that it is a natural control parameter for certain applications, e.g. streaming media, which have intrinsic sending rates. Also rate control generates traffic into the network smoothly.

2. *Placement of estimation algorithm.* The sender may not be the entity that runs the estimation algorithm. For scalability reasons, it can be more efficient for each receiver to run a local estimation algorithm, and send its estimation to the sender.

3. *Congestion Indicator (CI).* Selecting the right CI can be a difficult problem. The selection criteria are: (1) Is it a good indication of congestion? (2) Is it easy to monitor? Below we discuss 4 types of congestion indicators: Many other types can also be designed.

*NACK/ACK.* The simplest CI is receiving (ACK) or missing (NACK) a packet. They are the easiest to monitor, and give the fastest response. Experience of TCP shows that they work fine for unicast congestion control. However, ACK/NACK reflects only instantaneous network congestion status. Drop-tail router and traffic phase effects [9] generate bursty packet loss. As we will see in the next section, responding to this type of signal may not be desirable in multicast case.

*Packets queued.* This type of congestion indicator estimates the number of packets queued in the network by estimating the change in round trip time (RTT) [3]. The advantage of this congestion indicator is that it can detect congestion before packet loss. However, making accurate estimation of packets queued is a difficult problem.

*Loss rate.* One way to smooth the bursty NACK/ACK congestion indicator is to use loss rate. The difficulty is how to calculate the loss rate. If the calculation period is too short, it will still be bursty. If the calculation is over a longer period, the control will be less responsive.

*Incoming rate.* Another way to smooth the bursty NACK/ACK is to measure the packet incoming rate. Congestion can be detected when the sender's sending rate is higher than the receiver's measured incoming rate. The difference between these two rates reflects the packet loss rate. However, calculating the incoming rate for bursty traffic is not straightforward.

4. *Estimation algorithm.* The control parameter $c$ will be continously updated according to the received congestion indicator. In this paper, we define two specific types of estimation algorithms. Other types of estimation algorithms can be designed and should be explored.

   The first type of estimation algorithms is said to be *model-based*. In this case, the target control parameter $c_{ref}$ can be estimated from the congestion indicators using a model. For example, if the target is to achieve the same throughput as a TCP session under the same loss rate, a formula describing the achievable throughput of a TCP session as a function of loss rate can be used to calculate the estimated sending rate, using the measured loss rate. As an example of the formula, denote $T(p)$ to be the achieved throughput in $bytes/sec$, $S$ the packet size in $bytes$, $RTT$ the round trip time between the sender and the receiver, $t_o$ the timeout value, and $p$ the loss probability, the TCP formula can be expressed as [21]:

$$T(p) = \frac{S}{RTT\sqrt{\frac{2p}{3}} + t_o(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \quad (1)$$

   After estimating $c_{ref}$, the current $c$ is adjusted to approach $c_{ref}$. The estimation algorithm and the sender adjustment algorithm in this case are:

> Calculate $c_{ref}$ according to CI
> if current $c < c_{ref}$ then $c = c + \alpha * (c_{ref} - c)$
> else $c = c_{ref}$

**Figure 3. Model based estimation algorithm**

   The other type of estimation algorithm is the well-known Additive Increase Multiplicative Decrease (AIMD) [4] algorithm:

> if CI indicates congestion $c = c * a$
> if CI indicates no congestion in interval $T$ $c = c + b$

**Figure 4. AIMD estimation algorithm**

   where $a$ is a constant less than 1, and $b$ is another constant. In steady state (after slow start), TCP uses the AIMD estimation algorithm, with window size $wnd$ as control parameter, $a = 0.5$, $T = RTT$, and $b = 1$.

We have introduced the system model in this section. And we used TCP as an example. The behavior of TCP congestion control [14] is well-understood now. However, simply applying TCP congestion control to multicast does not appear to scale well. We will study the problems next.

## 3 Key Problems and Solution Approaches

With the above control model and terminology in mind, we next discuss the key problems and solution approaches.

### 3.1 Feedback Implosion Problem

The first problem facing multicast congestion control is the *feedback implosion problem*. For a sender to adjust its control parameter according to network traffic, it must receive feedback from the receivers (we assume no explicit network congestion support [23]). Therefore, we have to deal with the same feedback implosion problem as multicast error control [20, 7]. We identify two approaches to solve the feedback implosion problem: suppression-based, and structure-based.

- *Suppression.* In this approach, not all receivers will send their feedbacks to the sender. One solution is to choose some receivers as representatives, and only the representatives send their feedbacks [6]. The difficulty with this approach is how to select a suitable set of representatives. Another suppression approach is to adapt

the SRM [10] suppression algorithm designed originally for reliable multicast error control. In this algorithm, receivers control their feedbacks using random timers. The difficulty with this approach is how to set the timer. SRM[10] proposed to set the timer according to the distance (delay) between the sender and the receiver. This adds the requirement that every receiver has to measure the distance.

- *Structure-based.* Another approach to solve the feedback implosion problem is to organize the receivers into a structure, and feedbacks are propagated and *aggregated* through the structure. MTCP [24], TFMCC [12], and Golestani [11] all proposed to use a tree hierarchy to aggregate feedback traffic.

### 3.2 Congestion Indicator Filtering Problem

The second problem is the *congestion indicator filtering problem*. Unlike TCP, which just needs to control the single connection between a sender and a receiver, multicast congestion control has to support a wide range of operating parameters for each connection. As the number of receivers increases, the range of suitable transmission rates diminishes. Even worse, an inappropriate selection of congestion indicator and estimation algorithm may lead to low throughput.

An example will demonstrate the problem. Suppose we select NACK/ACK as the congestion indicator, and the TCP estimation algorithm. Since the sender may receive multiple NACKs for one packet, to avoid the worst case, we restrict the sender to *reduce its window only once for multiple NACKs of the same packet*. If no NACK is reported for a packet, the sender increases its window size linearly.

Because we simply adapted the TCP estimation algorithm, the TCP relationship between loss probability and throughput is still hold. Therefore, we can use the equation (1) to calculate the achieved throughput. From this equation, we can see that the higher the loss probability $p$, the lower the throughput. Consider a multicast session with $n$ receivers. Without worrying about the feedback implosion problem, suppose all receivers report their losses to the sender. According to mbone measurements [27], the individual receiver packet losses can be assumed to be independent. Assume the loss probability of a packet for each receiver is $p_i$. Then the probability $p$ of the sender receiving a loss report equals to the probability of at least one receiver losing a packet, which is $1-(1-p_i)^n$. Figure 5 shows the probability $p$ versus the number of receivers, assuming $p_i = 0.02$ and $p_i = 0.05$, which are on the low end of the mbone loss rates. And Figure 6 shows the achieved throughput $T(p)$. Also shown on Figure 6 is the throughput that a TCP unicast session can achieve between the sender and a receiver for the receiver's individual loss rate.

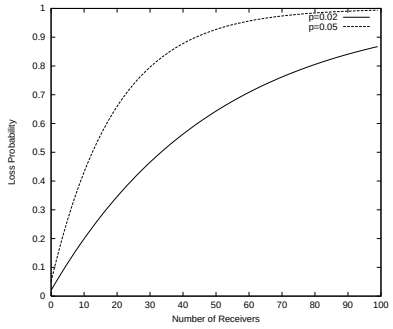


**Figure 5. Loss Probability versus number receiver $n$ ($p_i = 0.02$ or $0.05$)**

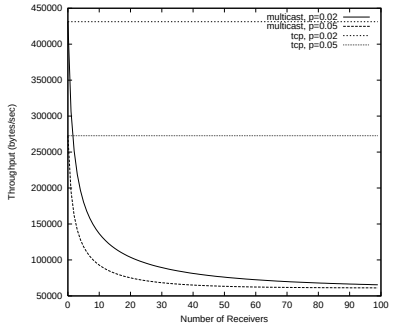


**Figure 6. Achieved Throughput**

What we have learned from this example is that asking the sender to respond to all congestion indicators will reduce the throughput to zero as the number of receivers increases. Rather, a filter scheme has to be added to solve this *reduce-to-zero problem*. So far, several approaches have been proposed to filter the congestion indicator:

- *Representative.* The Representative approach we discussed to solve the *feedback implosion problem* will also reduce the impact of this problem. Because its congestion indicators are restricted to be from only a small set of representatives, it reduces the impact of this problem. However, as previously discussed, how to select the right size and right set of receivers as representatives needs further research.

- *Suppression timer.* In LTRC [25] (Loss Tolerant Rate Controller), the sender will respond to only one loss report in a time period $TD$. The time $TD$ is a measure of the time it takes for a rate change to "flush" through the system. The paper gives a method to determine $TD$. The difficulty of this approach is to make the tradeoff between responsiveness and the reduce-to-zero problem. The effectiveness of this approach still

needs further study.

- *Lossiest receiver.* LPM [2] discussed an approach in which the sender responds only to the receiver with the highest loss rate. In our terminology, they selected loss rate as congestion indicator, and the congestion indicators are filtered using the maximum function. However, losses on a given link tend to be fractal because of the self-similar nature of Internet traffic [22] and the phase effects of drop-tail routers [9]. Therefore, it will be difficult to solve the reduce-to-zero problem in trying to filter the congestion indicator using maximum function.

- *Probability approach.* RLA [26] proposed a probability approach. In this approach, the sender responds to a congestion indicator with a probability, therefore the name *random listening algorithm*. In this way, it is responding to the average instead of the maximum of the receiver's loss rates.

To truly evaluate the above approaches, we need to not only study their effectiveness to solve the filtering problem, but also study their fairness property. Because of congestion indicator filtering, they may achieve a fairness property different from that of TCP. This leads us to the fairness problem of multicast congestion control.

### 3.3 Fairness Problem

The third difficulty of multicast congestion control is the *fairness* problem. Over the past couple of years, a key challenge in defining multicast congestion control is the lack of an agreed upon definition for fairness.

In the introduction section, we discussed several types of fairness: max-min fairness, and global resource fairness. From the format of the adjustment algorithms, [11] defined two other types of fairness: rate-oriented and window-oriented. Rate-oriented fairness tries to achieve equal throughput at the bottleneck resource. Window-oriented fairness achieves throughput proportional to the inverse of round trip time. Compared to the definitions of fairness in the introduction section, we observe that max-min fairness and rate-oriented fairness refer to the same type of fairness; also global resource fairness and window-oriented fairness refer to the same type of fairness. Therefore, we can call the fairness of TCP either window-oriented fairness or global resource fairness. Over the years TCP has become the standard transport protocol as well as the widely used protocol (90-95% of the bytes or packets) in the Internet. For this reason, it has been strongly argued that multicast congestion control should be TCP-friendly [17]. Therefore, the fairness problem to solve in multicast congestion control is to select the congestion indicator(s) and estimation algorithm so that it will achieve TCP fairness.

One seemingly obvious way to achieve this objective is to use the TCP congestion indicator (NACK/ACK) and estimation algorithm (AIMD algorithm) to do multicast congestion control. However, in the previous subsection we have shown this will lead to very low throughput. So far, two approaches show promise to solve both of the *congestion indicator filtering problem* and the *fairness* problem.

- *Golestani's approach.* In this approach [11], the congestion indicator is still the NACK/ACK. However, to avoid the *congestion indicator filtering problem*, it is not the sender but each receiver that runs its own copy of estimation algorithm. Therefore, it solves the *congestion indicator filtering problem*. Because each receiver runs the same AIMD estimation algorithm as TCP does, it achieves the same type of fairness TCP achieves. The problem with this approach is that no specific estimation algorithm has been described. Also there is still no experimental validation yet.

- *TCP-formula approach.* The basic idea of this approach [12] is very similar to the above one. In this approach, it is also the individual receiver who implements the estimation algorithm. However, in this case, what was proposed is rate instead of window control. Each individual receiver uses the TCP throughput formula to calculate the achievable throughput as if a TCP unicast connection were running between itself and the sender. With feedbacks from receivers, the sender will select the minimum of the rates and adjust its sending rate. The disadvantage of this approach is that each receiver has to measure both loss rate and round trip time. As we previously discussed, a tradeoff between responsiveness and accuracy must be made in these measurements.

### 3.4 Summary of Approaches

In this subsection, we use a table to summarize the current approaches.

## 4 Our Proposed Approach

### 4.1 Motivation

We have studied the current approaches in the previous section. We said that Golestani and TFMCC are two of the promising approaches. It is interesting to observe that these two approaches choose different control parameters: window control in Golestani and rate in TFMCC. As we discussed in the system model section, each type of control parameter has its own advantages and disadvantages. Therefore, the first motivation of our work is to design an

| Approach | Feedback implosion | Congestion indicator | CI filtering | Control parameter | Estimation algorithm | Control Placement | Fairness |
|---|---|---|---|---|---|---|---|
| LRA [26] | n/a | NACK ACK | random listening | window | AIMD | sender | essentially fair |
| MTCP [24] | tree structure | packets queued | min {receiver window size} - max {tree node buffer size} | window | AIMD | receivers, tree nodes | n/a |
| LTRC [25] | n/a | loss rate | suppre-ssion | rate | AIMD | receivers | n/a |
| Representative [6] | suppression | receive rate, pkt queued | representative | rate | AIMD | sender | n/a |
| LPM [2] | n/a | loss rate | max {receiver loss rate} | rate | AIMD | sender | max-min |
| TFMCC [12] | tree structure | loss rate | min {receiver TCP formula throughput} | rate | Model (TCP formula) | receivers | TCP |
| Golestani[11] | tree structure | NACK ACK | min {receiver window boundary} | window | AIMD | receivers | TCP |

**Table 1. Summary of Current Multicast Congestion Control Approaches**

algorithm to take the best of both approaches: smooth traffic of rate control and fast response of window control.

The second motivation of our design is to solve the *truncated TCP model* problem. In either Golestani or TFMCC approach, we call the receivers that feedback the lowest formula throughput or or smallest window size the bottlenacked receivers, and other receivers non-nonbottlnecked receivers. In TFMCC, the assumption is that the TCP throughput formula is still valid for non-dominant receivers. In an environment of FIFO scheduling and large-scale statistical multiplexing, it is reasonable to assume that the loss rate and delay experienced by a receiver are independent of the sender's sending rate. Therefore, the TCP formula may hold for non-bottlenecked receivers. However, in a small-scale statistical multiplexing environment, the loss rate a receiver experienced will depend on the sender sending rate. For non-bottlenecked receivers, because the sender is sending at a lower rate than the path allows, the receivers may experience lower loss rate or no loss at all. However, the receiver will still use the TCP formula to calculate the throughput, which assumes TCP behavior. Therefore, its estimated throughput can be much higher than the path allows. For example, the allowable throughput for a receiver $r_1$ is $2Mbps$. However, because of the limitation imposed by another lower throughput receiver, the sender is only sending at a rate of $1Mbps$. Therefore, $r_1$ may observe zero loss and its calculated TCP throughput is infinity. The same problem can also happen for the Golenstani's approach. In network environment with dynamic receiver membership, this overestimation can lead to a large amount of packet loss when the bottlenecked receivers leave. However, solving this problem is not very difficult. One way to solve it is to put an upper bound on the estimate of TCP throughput or window size by non-bottlenecked receivers. Take the TFMCC for example, the receiver should monitor the sender sending rate, and then, if its estimate is higher than the current sender sending rate, it can imply that it is a non-bottlenecked receiver. It should set its throughput estimate to the current sender's sending rate plus an increase step.

### 4.2 Algorithm Specification

We specify our approach according to the 4 design parameters we discussed in the system model section:

1. *Control parameter $c$.* We choose rate as the control parameter because it generates traffic smoothly.

2. *Placement of estimation algorithm.* Estimation algorithm will be run by each individual receiver.

3. *Congestion indicator.* Receiving and missing of a packet, which can be detected by sequence number.

4. *Estimation algorithm.* Instead of using the TCP formula to estimate the sending rate, each receiver runs the standard TCP congestion *window* estimation algorithm. Then, the rate estimation $R_i$ of receiver $i$ is derived from its congestion window size $cwnd_i$ and

round trip time $RTT_i$:

$$R_i = \frac{cwnd_i}{RTT_i}$$

Also, as discussed above, the congestion window size will be limited by $incoming\ rate * (1 + \alpha) * RTT$, where $\alpha (> 0)$ is the limit on how much the estimation can be higher than the current sending rate. This limit is to prevent the truncated TCP model problem

We are in the process of evaluating the stability and fairness of our approach using an analytical method. We have also done some simulations to evaluate its performance. Initial result shows it as a very promising approach. A prototype implementation using the reliable multicast protocol LGMP [13] is in progress. We will report our results in another paper.

## 5 Conclusion

Multicast congestion control is still a new and active research area. In this paper, we identified 3 key problems and discussed the current proposed solutions. We also make a proposal to improve upon the current best approaches. Because of the complexity of multicast congestion control, many problems have to be solved to form a complete solution. For example, one problem we did not discuss is how to limit the local recovery bandwidth in reliable multicast congestion control [25]. Another high priority task to be done is a comprehensive performance evaluation of the current proposed approaches.

## References


[1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Second Edition, 1992.

[2] S. Bhattacharyya, D. Towsley, and J. Jurose. The loss path multiplicity problem in multicast congestion control. In *Proceedings of IEEE INFOCOM '99*.

[3] L. Brakmo, S. O'Malley, and L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of ACM SIGCOMM '94*, Vancouver, Canada, May 1994.

[4] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989.

[5] S. Deering. Host extensions for IP multicasting. *RFC1112*, Jan. 1989.

[6] D. DeLucia and K. Obraczka. Multicast feedback suppression using representatives. In *Proceedings of IEEE INFOCOM '97*, 1997.

[7] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: A survey of protocols, functions and mechanism. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, 1997.

[8] S. Floyd. *Congestion Control Principles, INTERNET DRAFT*, Jan. 2000.

[9] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3(3):115–156, Sep 1992.

[10] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[11] J. Golestani and K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *Proceedings of IEEE INFOCOM '99*, 1999.

[12] M. Handley and S. Floyd. *Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC), Working Draft*, Dec. 1998.

[13] M. Hofmann. Enabling group communication in global networks. In *Global Networking'97*, Alberta, Canada, June 1997.

[14] V. Jacobson. Congestion avoidnace and control. In *Proceedings of ACM SIGCOMM '88*, 1988.

[15] C. Kent and J. Mogul. Fragmentation considerd harmful. *ACM Communications Review*, 17(5):110–120, Aug 1987.

[16] S. Low. Optimization flow control with on-line measurement. In *Proceedings of the 16th International Teletraffic Congress*, Edinburgh, UK, June 1999.

[17] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. IETF criteria for evaluating reliable multicast transport and application protocols. *RFC2357*, June 1998.

[18] J. Mogul, C. Kent, C. Patridge, and K. McCloghrie. *IP MTU Discovery Options, IETF RFC 1063*, July 1988.

[19] J. Nagle. Congestion control in IP/TCP internetworks. *ACM Communications Review*, 14(4):11–17, Oct 1984.

[20] K. Obraczka. Multicast transport mechanism: A survey and taxonomy. submitted to IEEE Communications Magazine, 1997.

[21] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, 1998.

[22] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[23] K. K. Ramakrishnan and S. Floyd. *A Proposal to add Explicit Congestion Notification (ECN) to IP, IETF RFC 2481*, Jan 1999.

[24] I. Rhee, N. Ballaguru, and G. N. Rouskas. MTCP: Scalable TCP-like congestion control for reliable multicast. In *Proceedings of IEEE INFOCOM '99*, 1999.

[25] M. T. A loss tolerant rate controller for reliable multicast. Technical report, NASA-IVV-97-011, August 1997.

[26] H. A. Wang and M. Schwartz. Achieving bounded fairness for multicast traffic and TCP traffic in the internet. In *Proceedings of ACM SIGCOMM '98*, 1998.

[27] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the mbone multicast network. *CCR*, Sept. 1993.

[28] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource ReSerVation Protocol. *IEEE Network Magazine*, 9(5), 1993.