Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet *

Huayan Amy Wang and Mischa Schwartz

Department of Electrical Engineering
Columbia University, New York, NY 10027.

Email: {whycu, schwartz}@ctr.columbia.edu

Abstract

There is an urgent need for effective multicast congestion control algorithms which enable reasonably fair share of network resources between multicast and unicast TCP traffic under the current Internet infrastructure. In this paper, we propose a quantitative definition of a type of bounded fairness between multicast and unicast best-effort traffic, termed "essentially fair". We also propose a window-based Random Listening Algorithm (RLA) for multicast congestion control. The algorithm is proven to be essentially fair to TCP connections under a restricted topology with equal round-trip times and with phase effects eliminated. The algorithm is also fair to multiple multicast sessions. This paper provides the theoretical proofs and some simulation results to demonstrate that the RLA achieves good performance under various network topologies. These include the performance of a generalization of the RLA algorithm for topologies with different round-trip times.

Key words: Multicast, flow and congestion control, Internet, RED and drop-tail gateways, phase effect.

1 Introduction

Given the ubiquitous presence of TCP traffic in the Internet, one of the major barriers for the wide-range deployment of reliable multicast is the lack of an effective congestion control mechanism which enables multicast traffic to share network resources reasonably fairly with TCP. Because it is crucial for the success of providing multicast services over the Internet, this problem has drawn great attention in the reliable multicast and Internet community. It was a central topic in the recent Reliable Multicast meetings [8], and many proposals have emerged recently [7, 15, 13, 1, 18, 3, 19]. In this introductory section, we first give an overview of the previous work and then we discuss our experience with the problem and introduce our approach.

The basic problem can be described as the following. Consider the transport layer of a multicast connection with one sender and multiple receivers over the Internet. The sender has to control its transmission rate based on the loss information obtained from all the receivers. We assume that there is TCP background traffic; the end-to-end loss information is the only mechanism to indicate congestion; the participating receivers have time-varying capacities; and different receivers can be losing different information at different times. The objective of the control algorithm is to avoid congestion and to be able to share network resources reasonably fairly with the competing TCP connections. The previously proposed multicast flow/congestion control algorithms for the Internet can be broadly classified into two categories: 1) Use multiple multicast groups. 2) A single group with rate-based feedback control algorithms.

The first category includes those proposals using forward error correction (FEC) or layered coding [18, 19]. They require setting up multiple multicast groups and require coordination between receivers, which are not always possible. Some limitations of this type of control are identified in [13].

Many of the proposed rate-based schemes share a common framework: The sender updates its rate from time to time (normally at a relatively large interval on the order of a second) based on the loss information obtained. It reduces its rate multiplicatively (usually by half, the same as TCP does) if the loss information indicates congestion, otherwise it increases its rate linearly. Different proposals differ in their ways of determining the length of the update interval and acquisition of loss information; they have different criteria to determine congestion, etc. It is largely agreed that, with no congestion, the rate should be increased linearly with approximately one packet per round-trip time, which is the same as TCP does. A critical aspect of these rate-based algorithms is when to reduce the rate to half, or how congestion is determined from the loss information from all the receivers. Most of the proposed algorithms are designed with the objective of identifying the bottleneck branches [1] of the multicast session and reacting only to the losses on the bottleneck links [13, 7]. Some also try to be fair to TCP [1, 15]. The algorithms have to be adaptive as well, i.e., be able to migrate to new bottlenecks once they come up and persist for a long time. In the following, we discuss two examples in detail.

The loss-tolerant rate controller (LTRC) proposal is based on checking an average loss rate against a threshold [13]. The algorithm tries to react only to the most congested paths and ignore other loss information. The algorithm

*This material is based upon work supported by the U.S. Army Research Office under grant number DAAH04-95-1-0188.

[1] By bottleneck branches, we refer to the branches where the bandwidth share of multicast traffic is the smallest among all multicast paths, assuming equal share of all connections going through the path.

identifies congestion and reduces the sender rate if the reported loss rate (an exponentially-weighted moving average) from some receiver is larger than a certain threshold. The rate is not reduced further within a certain period of time after the last reduction. It is not clear how to choose the loss threshold values for an arbitrary topology with any number of receivers to drive the system to the desired operating region.

The monitor-based flow control (MBFC) is a double-threshold-check scheme [15]. That is, a receiver is considered congested if its average loss rate during a monitor period is larger than a certain threshold (loss-rate threshold), and the sender recognizes congestion only if the fraction of the receiver population considered congested is larger than a certain threshold (loss-population threshold). Using the loss-population threshold to determine whether to reduce the rate or not is a means to average the QoS over all receivers, and is not aimed to work with the slowest receiver. As a special case, with the loss-population threshold set to minimum (one congested receiver is counted as congestion), the MBFC reduces to the case of tracing the slowest receiver, but, again, it is difficult to derive a meaningful threshold value to be able to single out the most congested receiver. If the threshold value is too small, there could be excessive congestion signals because different receivers could experience congestion at different times.

There are many other proposals which we cannot cover in this introduction. Although many of the proposals are claimed to be TCP-friendly based on the simulation results for certain network topologies, none have provided a quantitative description of fairness of their algorithms to TCP and a proof of their algorithms' ability to guarantee fairness.

We have carried out extensive simulations to study the interaction of TCP traffic with other forms of rate-controlled traffic in both unicast and multicast settings, with both drop-tail and RED (random early drop) gateways[2]. We summarize our major observations here and the details are discussed in the rest of this paper.

First of all, there is no consensus on the fairness issue between reliable multicast and unicast traffic, let alone a useful quantitative definition. Should a multicast session be treated as a single session which deserves no more bandwidth than a single TCP session when they share network resources? [8, 7]. Or should the multicast session be given more bandwidth than TCP connections because it is intended to serve more receivers? If the latter argument is creditable, how much more bandwidth should be given to the multicast session and how do we define "fairness" in this case? This paper addresses this problem and proposes an algorithm which allows a multicast session to obtain a larger share of resources when only a few of the multicast receivers are much more congested than others.

However, we believe that a consensus on the definition of relative fairness between multicast and unicast traffic is achievable once an algorithm shown to be "reasonably fair" to TCP is accepted by the Internet community. The toughest barrier to designing a fair multicast congestion control algorithm is that most of the current Internet routers are still of drop-tail type. A drop-tail router uses a first in first out (FIFO) buffer to store arriving packets when the outgoing link is busy. The FIFO buffer has a finite size and the arriving packet is dropped if the buffer is already full. Since drop-tail routers do not distinguish packets from different traffic flows, they do not enforce any fairness for the connections sharing resources through them. Also with drop-tail routers, the packet loss pattern is very sensitive to the way packets arrive at the router and is difficult to control in general. Since TCP packets tend to arrive at the router in clusters [21], any rate-based algorithm with an evenly-spaced packet arrival pattern may experience a very different loss rate from that of the competing TCP connections through a drop-tail gateway. Therefore, rate-based algorithms adjusting source transmission rate based on average loss rate cannot be fair to TCP in general. But, it has been pointed out that rate-based schemes are better suited for multicast flow/congestion control than window-based schemes [15]. This is true in general in terms of scalability and ease of design. However, if our design objective is to be fair to window-based TCP, rate-based schemes have difficulty, if not an impossibility, in achieving the goal without help from the networks.

For the algorithms assuming the same loss rate for the competing connections, RED gateways can be used to achieve the goal. The RED gateway is proposed as an active router management scheme which enables the routers to protect themselves from congestion collapse [5]. A RED gateway detects incipient congestion by computing the average queue size at the gateway buffer. If the average queue size exceeds a preset minimum threshold but below a maximum threshold, the gateway drops each incoming packet with a certain probability; if the maximum threshold is exceeded, all arriving packets are dropped. RED gateways are designed so that, during congestion, the probability that the gateway notifies a particular connection to reduce its window (or rate) is roughly proportional to that connection's share of the bandwidth through the gateway. Therefore, RED gateways not only keep the average queue length low but ensure fairness and avoid synchronization effects [6]. For our work, the most important fact about the RED gateway is that all connections going through it see the same loss probability. RED gateways also make fair allocation of network resources for connections using different forms of congestion control possible. Adoption of RED gateways will greatly ease the multicast congestion control problem, but the current Internet still uses mostly drop-tail gateways. Therefore, it is important to design an algorithm which works for drop-tail gateways and might work better for RED gateways.

However, even with RED gateways, it is still very difficult to locate the bottlenecks of a multicast session based on loss information alone (refer to section 3.2 for details). Many proposals for reliable multicast flow control do try to locate the bottleneck links using some threshold-based mechanism, such as the LTRC (loss-tolerant rate controller) discussed above, but it is very difficult to choose a universal threshold which works for all kinds of network topologies. [16] has shown that a loss-threshold-based additive-increase-multiplicative-decrease multicast control algorithm is not fair to TCP with RED gateways.

Based on the above observations, we decided to choose a window-based approach to design a usable mechanism to do multicast congestion control in the current Internet infrastructure. Specifically, we propose a random listening algorithm which does not require locating the bottleneck link. The algorithm is simple and possesses great similarity to TCP; it ensures some reasonable fairness, defined later as "essential fairness", to TCP with RED gateways or drop-tail gateways in a restricted topology to be defined in the next section. Although the scheme inherits many of the identified drawbacks of TCP (some of them are alleviated in our multicast scheme), it might be the only way that the multicast sessions can potentially share bandwidth reasonably fairly

[2] Routers and gateways are used interchangeably in this paper. See [5] for definitions for drop-tail and RED gateways.

with TCP connections with drop-tail routers.

The rest of the paper is organized as follows: We propose a quantitative definition for fairness between multicast and unicast traffic in section 2. Our algorithm is presented in section 3, and we prove that it is essentially fair to TCP in section 4. In section 5 we present some simulation results indicating the performance of our algorithm sharing resources with TCP. We also briefly discuss a generalization of the algorithm which works for topologies with different round-trip times and its performance. Section 6 concludes the paper by addressing some possible future work.

2 Design Objectives

Our design of the multicast congestion control algorithm is motivated by the design of the TCP congestion control scheme [9, 12]. We summarize the basic properties of the TCP scheme in the following:

- *Probing extra bandwidth*: increase the congestion window by one packet per round-trip time until a loss is seen.
- *Responsive to congestion*: reduce the congestion window to half upon a detection of congestion (i.e., a packet loss).
- *Fair*: by using the same protocol, the TCP connections between the same source and destination pair (along the same route) share the bottleneck bandwidth equally in the steady state.[3]

Similarly we list our design objectives for the multicast congestion control algorithm to be:

- Able to probe and grab extra bandwidth.
- Responsive to congestion.
- Multicast Fairness: multiple multicast sessions between the same sender and receiver groups should share the bandwidth equally on average over the long run.
- Fair to TCP: the multicast traffic has to be able to share the bandwidth reasonably fairly with TCP in order to be accepted by the Internet community. This is a complex issue to be addressed in the rest of this section.

Note that our performance goals, including definitions of fairness, are focused on the average behavior in the steady state, assuming connections last for a long time. Our work in this paper is based on this assumption. We do not try to guarantee fairness to short-lived connections, but our algorithm does provide opportunities for them to be set up and to transmit data. This is a reasonable decision because the multicast session is presumably cumbersome with many links involved and thus it is impossible to react optimally to every disturbance, especially short-lived ones.

2.1 Observations

We observe that TCP fairness is defined and achieved only for the connections between the same sender and receiver, that is, the paths have to have the same round-trip times and the same number of congested gateways. It is well-recognized that the unfairness of the TCP congestion algorithm, such as biases against connections with multiple congested gateways, or against connections with longer round-trip times and against bursty traffic, is exhibited in networks with drop-tail gateways [5]. This observation leads us to define the relative fairness between multicast and TCP traffic on a *restricted* topology only, where the sender has the same round-trip time to all the receivers in the multicast group.

As we mentioned in the introductory section, there is no consensus on the issue of fairness between multicast and unicast traffic. But the following is obvious: An ideal situation is to be able to design a multicast algorithm which can control and adjust the bandwidth share of the multicast connection to be equal to some constant c times that of the competing TCP connection, with c being controllable by tuning some parameters of the algorithm. On the other extreme, the minimum requirements of reasonable fairness should include the following: 1) Do not shut out TCP completely. 2) The throughput of the multicast session does not diminish to zero as the number of receivers increases. Anything in between the ideal and the minimum could be reasonable provided that the cost to achieve it is justifiable. Loosely speaking, a useful definition for "essentially fair" could be the following: when sharing a link with TCP, the multicast session should get neither too much nor too little bandwidth; that is, some kind of bounded fairness. We quantify this definition next.

2.2 Concepts

First we introduce a *restricted topology*, referred to throughout this paper, on which the fairness concepts are defined. We also introduce the notation used throughout the paper. Consider a multicast session $\{S \rightarrow R_i, i = 1, 2, \ldots, N\}$ with one sender S and N receivers $R_1, \ldots, R_N$. The sender also has m_i separate TCP connections to each R_i along the same path (a branch of the multicast tree) [see figure 1], where m_i could be zero (no competing TCP connection). Imagine a virtual link (or a logical connection), L_i, between S and R_i. Note that the virtual links might share common physical paths. We assume that the round-trip times, RTT_i, of L_i are equal on the average.[4] Denote the minimum link capacity (or available bandwidth) along L_i by μ_i (pkt/sec).

We define the "*soft bottleneck*" of a multicast session, denoted by L_{sb}, as the branch with the smallest $\mu_i/(m_i+1)$.[5] That is, $L_{sb} = \arg\min_i\{\frac{\mu_i}{m_i+1}\}$. We say that the multicast is "*absolutely fair*" to TCP if the multicast session operates with an average throughput equal to $\min_i\{\mu_i/(m_i+1)\}$ in the steady state. In other words, "absolute fairness" requires that the multicast session be treated as a single session and equally share the bottleneck bandwidth with competing TCP connections on its soft bottleneck paths.

As we mentioned before, absolute fairness is difficult to achieve based on loss information alone. By relaxing the definition somewhat, we introduce an important concept called "*essential fairness*". We say that a multicast session is "*essentially fair*" to TCP if its average throughput, denoted by

[3] Generally speaking, the TCP connections share bandwidth equally as long as they have equal round-trip times and the same number of congested gateways on their path. But a slight difference in the round-trip times could result in very different outcome in bandwidth share due to the phase effect discussed in [5]; therefore, we restrict the fairness definition to the connections between the same source and destination pair along the same route which is the best way of ensuring equal round-trip times.

[4] Notice that the round-trip time includes both queueing delay and propagation delay. Therefore, it is time varying. In our analysis in this paper, we assume a nice property of round-trip time: it is uniformly distributed between pure propagation delay and propagation delay plus maximum queueing delay. It is based on the single bottleneck queue model.

[5] In contrast, a "*hard bottleneck*" would be the link with minimum capacity $\mu = \min_i \mu_i$. Also, there could be multiple soft bottlenecks with equal $\mu_i/(m_i+1)$.

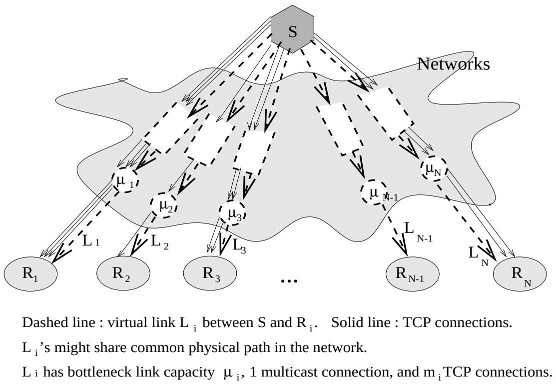

Figure 1: A restricted topology.

λ^{RLA}, in the long run is bounded by $a * \lambda^{TCP} < \lambda^{RLA} < b * \lambda^{TCP}$, where λ^{TCP} is the average throughput of the competing TCP connections on the soft bottleneck path, and a, b are functions of N such that $a \leq b < N$. Absolute fairness is a special case of essential fairness with $a = b = 1$ and $\lambda^{TCP} = \lambda^{RLA} = \min_i\{\frac{\mu_i}{(m_i+1)}\}$. b/a can serve as an indication of the tightness of the fairness measure. The flexibility of allowing an interval of fairness is necessary because absolute fairness might not be achievable in some networks whereas a fairness measure is needed. It is a reasonable representation of the vague term "reasonably fair", and would appear to be acceptable by many applications. The merit of the essential fairness concept lies in its boundedness, so the networks and applications can have some idea of what they can expect. Our definition can be used to measure and compare the fairness of existing multicast algorithms. We will prove later that the random listening algorithm we propose in this paper is essentially fair to TCP and it achieves more tightly bounded fairness with RED gateways than with drop-tail gateways.

In summary, we have defined three key concepts for multicast sharing with unicast traffic on a restricted topology: soft bottleneck, absolute fairness and essential fairness. The definitions can be easily extended to the case with multiple multicast sessions between the same sender and receiver group. In the next section, we present a random listening algorithm which achieves the design objectives described in the beginning of this section.

3 Random Listening Algorithm (RLA)

In this paper, we focus on the congestion control problem. We assume that the sender has infinite data to send and the receivers are infinitely fast, so that the network is always the bottleneck. Hereafter, we refer to a *congested receiver*, meaning the path between the sender and the receiver is congested, i.e., experiences packet drops. We also define a *congestion signal* as an indication of congestion according to the algorithm; *congestion probability* as the ratio of the number of congestion signals the sender detected to the number of packets the sender sent; *congestion frequency* as the average number of congestion signals the sender detected per time unit. TCP considers packet losses as indications for congestion. In particular, one or multiple packet drops from one window of packets in TCP are considered as one congestion signal since they usually cause one window cut (or cause retransmission timeout) [4]. The number of window cuts is equal to the number of congestion signals in TCP in the ideal case without timeout event.

Our simulation experience convinced us that, with drop-tail gateways, algorithms might have to be "TCP-like" in order to be TCP-friendly. By TCP-like, we refer to the essential feature of the congestion window adjustment policy: increasing by one every round-trip time with no congestion and reducing by half upon congestion. But TCP-like alone is not enough to ensure TCP-friendliness. More importantly, to be fair to TCP, we have to make sure that the multicast sender and the competing TCP senders are consistent in their way of measuring congestion. RED gateways ensure that the competing connections sharing the same bottleneck link experience the same loss probability no matter what type of congestion control algorithms are used. However, the situation with drop-tail gateways is more complicated. We will show that, with some added random processing time to eliminate phase effects, we can design our algorithm to ensure that the competing connections see the same congestion frequency. This problem is examined in detail next.

3.1 TCP's Macro-effect with Drop-tail Gateways

With drop-tail gateways, a basic feature of TCP traffic is that the sender increases its transmission rate to fill up the bottleneck buffer until it sees a packet loss. Then the sender's transmission rate is sharply reduced to allow the buffer to be drained. The TCP congestion control policy results in a typical behavior of the buffer in front of the bottleneck router: the buffer occupancy periodically oscillates between empty (or almost empty) and full. Although this periodicity is neither necessarily of equal interval nor deterministic, depending upon the behavior of the cross traffic other than TCP, it is certainly the macroscopic behavior of the network routers carrying TCP traffic. We call the period starting from a low occupancy to a full buffer and then dropping back to a low occupancy, a "*buffer period*".

Through simulation, we find that the buffer period normally lasts much longer than two round-trip times, $2RTT$, and the buffer-full period [6], during which the buffer is full or nearly full, normally lasts around $2RTT$ or less in the steady state. During the buffer-full period within each buffer period, a sender could lose more than one packet. In our algorithm to be presented, we group the losses within $2RTT$ as one congestion signal. This way we approximately make sure of one congestion signal per buffer period if any packet is dropped. The reason for doing so is that it is not desirable to reduce a window multiple times due to closely spaced packet drops. TCP actually considers multiple packet drops within one window as one congestion signal.

Another phenomenon is that, with drop-tail gateways, the packet drop pattern is very sensitive to the packet arrival pattern. In particular, we find that the packet drop pattern is very sensitive to the interval between two consecutive packet arrivals at the bottleneck buffer. If the interval is slightly smaller than the service time of the bottleneck server, the next packet is more likely to be dropped when the buffer is nearly full. Otherwise, if it is slightly larger than the bottleneck server service time, the next packet is less likely to be dropped because one packet will leave the buffer in between. This is one type of phase effect identified in [5]. Phase effects do not take place in competing TCP connections when the round-trip times are exactly the same. But in a multicast session which consists of multiple links with different instantaneous round-trip times, adding a random

[6] This time interval roughly corresponds to the "drop period" defined in [5].

amount of processing time is necessary to avoid the phase effect. Therefore, a uniformly distributed random processing time up to the bottleneck server service time is added in our simulation with drop-tail gateways. The phase effect might not be significant in the real Internet because of mixing of different packet sizes, in which case our algorithm is expected to work well, sharing bandwidth reasonably fairly with TCP traffic.

With drop-tail gateways and added randomness to eliminate the phase effect, our TCP-like multicast algorithm is designed to make sure that the multicast sender sends packets in a fashion similar to the TCP senders. Then all senders have a similar chance to encounter packet drops in a restricted topology with equal round-trip times, provided that the congestion window sizes are large enough. That is, both the multicast and TCP senders see roughly the same number of congestion signals over a long period of time, or the congestion frequencies should be the same on the average over a large number of simulations. For the connections with smaller window sizes, they might experience fewer packet drops, which results in a desirable situation for our problem of bandwidth allocation between multicast and unicast traffic. The reason is explained in section 5.

3.2 Rationale for Random Listening

If the objective is to achieve absolute fairness between multicast and TCP traffic, we have to locate the soft bottlenecks of the multicast session and react only to the congestion signals from the soft bottleneck paths. However, it is difficult, if not impossible, to locate the soft bottlenecks based on the loss information alone. For the TCP connections to achieve the same average throughput, the larger the round-trip time is, the larger the window and hence smaller loss probability required. Therefore, for a multicast session with different round-trip times between the sender and receivers, it is not reasonable to expect the bottleneck would be the branch with the largest loss probability. Although, for the restricted topology with equal round-trip times, the soft bottlenecks are the branches with the largest loss probability, it is still difficult to locate them based on loss information alone. This is because either the sender or the receiver has to calculate a moving average of the loss probability for each receiver and the sender has to react to only the loss reports from the bottlenecks which have the largest loss probability. But, since losses are rare and stochastic events, a certain interval of time and enough samples are needed to make the loss probability estimate significant. It would take too long to locate the soft bottlenecks correctly; the wrong action based on the non-bottleneck branches could cause undesirable performance results. Based on these observations, we decided to trade off the absolute fairness (requiring locating the soft bottlenecks) with fast response.

Now examine figure 1. The multicast sender is receiving congestion signals from all congested receivers. Obviously the sender does not want to reduce its window upon each congestion signal. Otherwise, as the number of receivers increases, the number of congestion signals will increase, and the throughput of the multicast session will decrease as the number of receivers increases.

Suppose that the sender knows how many receivers, say n, are reporting congestion. An appealing solution would be to reduce the window every n congestion signals. To see why, consider a simple flat tree topology as in figure 1 with all the receivers, links and background TCP connections identical and independent, and all connections starting at the same time. Then the buffer periods are synchronized and the sender receives n congestion signals in each buffer period. Obviously it is desired that the sender only reduce its window once every buffer period. This deterministic approach is certainly a possible solution here. But in a more realistic network with not everything identical, where buffer periods are asynchronous and congestion signals come at different times, the sequence of congestion signals arriving at the sender could be very irregular, and thus the deterministic approach would not work well. On the other hand, in such a statistical environment, a random approach could be a good candidate to produce good average performance. This is the rationale we use to propose a random listening scheme to handle a complex stochastic stream of congestion feedback signals.

The basic idea is that, upon receiving a congestion signal, the sender reduces its window with probability $1/n$, where n is the number of receivers reporting frequent losses. Therefore, on the average, the sender reduces its window every n congestion signals. If all the receivers experience the same average congestion, the sender reacts as if listening to one representative of them. If the sender detects one receiver experiencing the worst congestion (on the soft bottleneck) and the others in better condition with less frequent congestion signals, it reduces the window less frequently than the TCPs on the soft bottleneck branch, resulting in a larger average window size of the multicast sender than that of the TCP connections on this branch. But we can prove that the multicast bandwidth share is bounded in terms of the TCP bandwidth share. Based on this idea, we propose a random listening algorithm to be presented next, with its performance to be discussed in the rest of this paper.

3.3 The Algorithm

The design closely follows the TCP selective acknowledgment procedure (SACK) [12]. We focus on the congestion control part of the algorithm. Here we only outline the essential part of the algorithm. The complete algorithm is implemented using Network Simulator (NS2) [17], and more information is available at [20].

The important variables are summarized below. Their meaning and maintenance are the same as in TCP unless specified differently here. The items preceded by a bullet are new to our algorithm.

- $cwnd$: congestion window size.
- $ssthresh$: slow start threshold.
- $srtt_i$: smoothed round-trip times between the sender and receiver i.
- $awnd$: moving average of the window size.

• $num\_trouble\_rcvr$: a dynamic count of the number of receivers which are reporting losses frequently.

• $pthresh$: a dynamically adjusted threshold to determine the probability of reducing the window upon a congestion signal. For a restricted topology, $pthresh = 1/num\_trouble\_rcvr$.

• π : a random number uniformly distributed in (0, 1), generated when a decision as to whether to reduce the window or not is needed.

• $last\_window\_cut$: the time when the cwnd was reduced to half last time.

• $cperiod\_start_i$: the starting time of a congestion period (i.e., the period in which packets are dropped) at receiver i. This is used to group the losses within two round-trip times into one congestion signal.

- *min_last_ack* : the minimum value of the cumulative ACK sequence number from all receivers. All packets up to this sequence number are received by all receivers.
- *max_reach_all* : the maximum packet number which is correctly received by all receivers. It could be different from *min_last_ack* because selective acknowledgment is used.
- *rexmit_thresh* : if the number of receivers requesting a retransmission of a lost packet is larger than *rexmit_thresh*, the retransmission is multicasted. Otherwise the retransmission is unicasted.

The skeleton of the RLA is the following:

1. *Loss detection method.* Our multicast receivers use selective acknowledgments using the same format as SACK TCP receivers [12]. A loss is detected by the sender via identifying discontinuous ack sequence numbers or timeout. To accommodate out-of-order delivery of data, the sender considers a packet P is lost if a packet with a sequence number at least three higher than P is selectively ACKed.
2. *Congestion detection method.* A congestion period starts when a loss is detected and the $cperiod\_start_i$ is beyond $2 * srtt_i$ ago; then $cperiod\_start_i$ is reset to the current time. The losses within $2*srtt_i$ of $cperiod\_start_i$ are ignored.
3. *Window adjustment upon a congestion detection.* Upon a congestion detected from receiver i by the above method:
   - update *num_trouble_rcvr*. If it is a rare loss from a receiver not considered as a troubled receiver (see rule 6 below), skip the following steps.
   - if *last_window_cut* is beyond $2 * awnd * srtt_i$, [7] $cwnd \leftarrow cwnd/2$. *forced-cut.*
   - else, generate a uniform random number π, if $\pi > pthresh$, ignore it. else $cwnd \leftarrow cwnd/2$. *randomized-cut.*
4. *Window adjustment upon ACKs.* Once a packet is ACKed by all the receivers, $cwnd \leftarrow cwnd + 1/cwnd$.
5. The window lower bound moves when *max_reach_all* increases, but the window upper bound should never exceed *min_last_ack* plus available receiver buffer size.
6. Update of *num_trouble_rcvr*: a congested receiver is considered as a *troubled receiver* only if the receiver's congestion probability is larger than a certain threshold, which is set to $1/(\eta * min\_congestion\_interval)$. η here is a constant, and is recommended to be set to 20. *min_congestion_interval* is the smallest of the exponentially-weighted moving average of the interval lengths between congestion signals from all receivers. This setting is justified in the proof in the next section. *num_trouble_rcvr* is the dynamic count of the number of troubled receivers. A detailed implementation instruction is available at [20].

[7] In ideal TCP with deterministic losses, *cwnd* has a maximum size of W and a minimum of $W/2$. *cwnd* is halved every $W/2$ round trips, or $RTT * W/2$ seconds. Here for the multicast connection using random listening approach, to avoid ignoring too many consecutive congestion signals due to the randomness of the algorithm, we choose to force the reduction of the congestion window if the previous window cut happens at least $2 * awnd$ round trips ago. The threshold value is *ad hoc* but works well from our simulation experience: Basically we don't want *cwnd* to grow too large or the forced-cut to happen too often.

Note that there are two different treatments to a congestion signal: forced-cut and randomized-cut. The forced actions are intended to protect the system by damping the randomness. Without the forced-cut step, the algorithm can possibly result in too long of a continuous increment of *cwnd*, which is not desirable.

We also implemented a retransmission scheme [8] to recover loss packets and a fast-recovery mechanism to prevent a suddenly widely-open window which is undesirable because it can cause congestion and a burst of packet losses. Many details in the implementation are not described here and can be found in [20]. Many of them are just straightforward extensions from the TCP algorithm. We believe it is beneficial to keep it as similar to TCP as possible. Then any changes to TCP or in networks to improve TCP performance can be easily incorporated and are likely to improve the performance of our algorithm as well.

4 Fairness of the RLA

In this section, we prove that our RLA is *essentially fair* to TCP. That is, with the restricted topology where a multicast session is sharing resources with unicast TCP connections, the multicast session gets a bandwidth share which is c times the share of a competing TCP connection on the soft bottleneck branch, where c is a bounded constant. We present a simple proof based on some gross simplifications of the system and the algorithm. Although a sophisticated proof based on advanced stochastic processes, similar to the proof for the TCP case [14], is possible, we choose a simple approach which is easier to understand and better illustrates our idea. We also prove the multicast fairness property of the RLA, one of the design objectives mentioned in section 2, using a simple two-session model.

We first present a simple estimation of TCP's performance adopted from [14], then we use the same idea and result to prove our theorems. The key part of the proofs is to show that the RLA results in an average window size bounded from above and below by functions of the congestion probability (the ratio of the number of congestion signals to the number of packets sent, see section 3) on the soft bottleneck branch. Since on each common link, the RLA sender and the competing TCP senders see the same loss probability with RED gateways [6], or the same congestion frequency with drop-tail gateways with phase effects eliminated (see section 3.1), a relation between congestion probabilities of the two types of traffic can be derived, based on which the bandwidth shares can be calculated.

4.1 Estimation of TCP Throughput

We consider TCP SACK here and use the approximation technique introduced in [14]. Although there are many subtleties in the implementation of fast retransmission and fast recovery, etc., we list the most important parts of the algorithm relevant to congestion control here.

The sender maintains *cwnd* and *ssthresh*, with the same meaning as defined in section 3. The sender also estimates the round-trip time and calculates the timeout timer based

[8] There could be different ways of doing retransmission as long as the retransmission traffic does not interfere too much with the normal transmission. In our implementation, the sender waits until it hears from all the receivers and it retransmits a lost packet by multicast if the number of receivers requesting it is larger than a threshold (*rexmit_thresh*) and by unicast otherwise. The receiver can also trigger an immediate retransmission of a lost packet by unicast if it sets a field in the packet.

on the estimation. The TCP congestion window *cwnd* evolves in the following way:

(1) Upon receiving a new ACK:
    if $cwnd < ssthresh$,
        $cwnd \leftarrow cwnd + 1$; *slow start phase.*
    else $cwnd \leftarrow cwnd + 1/[cwnd]$;
        congestion avoidance phase.

(2) Upon a loss detection:
    set $ssthresh \leftarrow cwnd/2$, and $cwnd \leftarrow cwnd/2$.

(3) Upon timeout,
    set $ssthresh \leftarrow cwnd/2$, and $cwnd \leftarrow 1$,

where $[\ x\ ]$ denotes the integer part of x.

We consider *cwnd* as a random process and are interested in its average value in the steady state since it is roughly proportional to the average throughput of the TCP connection. Assume perfect detection of packet losses, and that the slow-start and the timeout events can be ignored in the steady state analysis [10]. Suppose we run the algorithm for a long time and the resulting congestion probability (the number of window cuts over the number of packets sent) is p. The resulting average window size can be approximated in the following way [14]. Denote the *cwnd* right after receiving the acknowledgment with sequence number t by W_t. Then in the steady state, the random process W_t evolves as follows: given W_t, with probability $(1-p)$, $W_{t+1} = W_t + 1/W_t$; and with probability p, $W_{t+1} = W_t/2$. W_t stays constant between jumps upon acknowledgment arrivals. Now considering the average drift of W_t if $W_t = w$, denoted by $D(w)$, we have $D(w) = (1-p)/w - p * w/2$. Note $D(w) = 0$ if $w = \sqrt{2(1-p)}/\sqrt{p} =: w^*$. The drift is positive if $w < w^*$ and negative otherwise. Then the stationary distribution of W_t must have most of its probability in the region around w^*. This gives an *ad hoc* approximation of the average window size, $\overline{W}$, by

$$\overline{W} = \frac{\sqrt{2(1-p)}}{\sqrt{p}} \approx \frac{\sqrt{2}}{\sqrt{p}} \text{ if } p \ll 1, \qquad (1)$$

The unit is in terms of packet. Throughout this paper, we call this approximation "*proportional average* (PA) window size". It can be shown that $\overline{W}$ is a good approximation to the time average of the random process W_t and in fact is proportional to it. We adopt this simple approximation approach in our analysis since it is adequate for our purpose.

Also note that the above simple derivation gives a result similar to the popular formula for TCP throughput estimation, $bandwidth = 1.3/(RTT\sqrt{p})$ (packets), as in [11], with a slightly different constant. Comparison of the two formulas shows that the average throughput is roughly proportional to the ratio of the average window size to the average round-trip time. Both formulas only work for the cases with small loss probability. Therefore, in the rest of our paper, we only consider the cases with $p < 5\%$ (used in [11]), called moderate congestion. The performance of TCP (and TCP-like algorithms) deteriorates in heavy congestion because of frequent timeout events. Maintaining fairness is then not as important an issue as long as no one is completely shut out.

4.2 Estimation of RLA Throughput

Applying the above drift analysis technique to our RLA algorithm proposed in section 3, we can derive the following proposition.

Proposition: Consider a restricted topology (with TCP background traffic and the RLA used by the multicast sender) and n receivers persistently reporting congestion. The congestion probabilities seen by the multicast sender from the n receivers are p_i, $i = 1...n$, and congestion is moderate so that $p_{max} = \max_i(p_i) < 5\%$. Denote the proportional average of the congestion window size by $\overline{W}$. Then $\overline{W}$ satisfies the following:

$$\frac{\sqrt{2(1-p_{max})}}{\sqrt{p_{max}}} < \overline{W} < \sqrt{n} * \frac{\sqrt{2(1-p_{max})}}{\sqrt{p_{max}}}. \qquad (2)$$

Due to space limitations, we cannot show the complete proof which is available in reference [20]. The basic idea and methodology are illustrated using the following simple case of two receivers with independent loss path (see figure 2(a)).

(a) independent losses (b) common losses

Figure 2: Two simple cases with two receivers only.

In figure 2(a), the sender sees independent congestion signals from receivers 1 and 2, denoted by R1 and R2, respectively. We assume that all traffic is persistent and thus, in the steady state, $n = num\_trouble\_rcvr = 2$ and $pthresh = \frac{1}{2}$. Therefore, if the sender detects a congestion signal at time t_0, it cuts its window by half with probability $\frac{1}{2}$; if the outcome turns out to be ignoring the congestion signal, the lost packet will be ACKed later (at time t_1) and will cause the congestion window to increase by $\frac{1}{W}$, based on the fourth rule of the RLA (see section 3.3). In our proof, we ignore the possible difference between the congestion window sizes at time t_0 and t_1 since $\frac{1}{W}$ is small for a relatively large window size. Now for each packet sent by the sender, the possible outcomes are (w.p. stands for with probability):

1. No congestion signal from both receivers, w.p. $(1-p_1)(1-p_2)$;
   then $W \leftarrow W + \frac{1}{W}$.
2. Cause one congestion signal from R1, w.p. $p_1(1-p_2)$;
   then $W \leftarrow W + \frac{1}{W}$ w.p. $\frac{1}{2}$, or
   $W \leftarrow \frac{W}{2}$ w.p. $\frac{1}{2}$.
3. Cause one congestion signal from R2, w.p. $(1-p_1)p_2$;
   then $W \leftarrow W + \frac{1}{W}$ w.p. $\frac{1}{2}$, or
   $W \leftarrow \frac{W}{2}$ w.p. $\frac{1}{2}$.
4. Cause two congestion signals from both receivers w.p. $p_1 p_2$;
   then $W \leftarrow W + \frac{1}{W}$ w.p. $\frac{1}{4}$, or
   $W \leftarrow \frac{W}{2}$ w.p. $\frac{1}{2}$, or
   $W \leftarrow \frac{W}{4}$ w.p. $\frac{1}{4}$.

The positive drift of W is

$$\frac{1}{W}\{(1-p_1)(1-p_2) + \frac{1}{2}p_1(1-p_2) + \frac{1}{2}(1-p_1)p_2 + \frac{1}{4}p_1 p_2\},$$

and the negative drift of W is

$$\frac{W}{2}\{\frac{1}{2}p_1(1-p_2) + \frac{1}{2}(1-p_1)p_2 + \frac{1}{2}p_1 p_2\} + \frac{3W}{4}\frac{1}{4}p_1 p_2.$$

The neutral point gives the approximation for the average window size to be

$$\overline{W}^2 = \frac{4\{1 - \frac{1}{2}(p_1 + p_2) + \frac{1}{4}p_1 p_2\}}{p_1 + p_2 - \frac{1}{4}p_1 p_2}. \tag{3}$$

It is easy to check out by simple algebraic manipulation that, for any $p_{max} > 0$, $\overline{W} > \sqrt{2(1 - p_{max})}/\sqrt{p_{max}}$ holds.

To prove the upper bound in equation 2 with $n = 2$, denote $p_2 = x * p_1$, and without loss of generality assume $0 < x \leq 1$, that is, $p_{max} = p_1 > 0$. Then the following holds:

$$\overline{W}^2 = \frac{4\{1 - \frac{1}{2}(1+x)p_1 + \frac{x}{4}p_1^2\}}{(1+x)p_1 - \frac{x}{4}p_1^2} < \frac{4(1-p_1)}{p_1},$$

if $x \geq f(p_1) = p_1/(2 - 1.5 * p_1)$. Note that $f(p_1)$ is an increasing function of p_1 for $0 < p_1 < 1$. Therefore, for $p_1 < 5\%$, x larger than 0.03 is sufficient for $x \geq f(p_1)$ to hold. This condition is ensured in the RLA algorithm by controlling the way the variable "*num_trouble_rcvr*" is dynamically counted. The RLA algorithm counts a receiver as a "troubled receiver" only if the interval lengths between the congestion signals are smaller than $\eta * min\_congestion\_interval$, that is, its average congestion probability is larger than $\frac{1}{\eta} * p_{max}$. We recommend in our algorithm to take $\eta = 20$, or $\frac{1}{\eta} = 0.05$ which leaves more room than the above 0.03 bound. Protocol designers can choose a proper value for η based on the above analysis.

Note that we did not consider the forced-cut action in the RLA algorithm, which is rarely invoked (as shown in the simulation results). The effect of the forced-cut could be a slightly smaller average window size which does not affect our results in any significant way.

With more complex algebraic manipulation involved, the results can be extended to a case with n receivers with independent loss paths. Using the same approach, for a topology with common losses only (see figure 2(b) for an illustration of the case with two receivers), we can prove equation 2 holds. The general case stated in the Proposition can be proved using the above results and the following Lemma:

Lemma: A higher degree of correlation in loss due to common path results in a larger average congestion window size if the RLA is used.

The proof is omitted due to space limitations. Intuitively, for the same congestion probability, correlation in the congestion signals results in more window increments and less window cuts on the average. This is because congestion signals come in groups in the correlated case. This has the potential of causing a deep cut in *cwnd* at once, while the independent congestion signals come one at a time but more frequently, and cause potentially more window cuts.

We deliberately choose the bounds in the form of equation 2 because $\sqrt{2(1-p)/p}$ is related to the average window size of the competing TCP connections. Now we are ready to proceed to show that the RLA is essentially fair to TCP.

Theorem I: Consider a restricted topology with RED gateways. If there are n receivers persistently reporting congestion and the largest congestion probability is less than 5%, the RLA algorithm is essentially fair to TCP with $a = \frac{1}{3}$ and $b = \sqrt{3n}$. [9]

Due to space limitations, here we only outline the major steps of the proof. First, since RED gateways ensure that on each link, all competing connections see the same loss probability [6], we denote the largest loss probability, occurring at the soft bottleneck branches, by p_l. We can derive the relations between p_l and the corresponding congestion probabilities, p_c^{RLA} for the multicast connection, and p_c^{TCP} for the TCP connection. Then, using these relations and the Proposition, we can derive the following inequality:

$$\frac{2}{3}\overline{W}_{WTCP} < \overline{W}_{RLA} < \sqrt{3n}\,\overline{W}_{WTCP}. \tag{4}$$

Second, we have to consider the round-trip times in order to estimate the throughput. Here we have to notice that in the multicast RLA, a packet is considered acknowledged only if the sender has received ACKs from all the receivers. Then the round-trip time for each packet in the RLA is always the largest among the round-trip times on all the links when the packet is sent. Denote the average round-trip time for RLA by $\overline{RTT}_{RLA}$ and that for TCP by $\overline{RTT}$ (recall each of the branches in the restricted topology (figure 1) has equal average round-trip times). Using the approximation that the round-trip time is equal to a fixed propagation delay plus a varying queueing delay, we can derive the following:

$$\overline{RTT} \leq \overline{RTT}_{RLA} \leq 2\overline{RTT}. \tag{5}$$

Finally, combining the bounds for average window size and average round-trip times, we have the following:

$$\frac{\overline{W}_{WTCP}}{3\overline{RTT}} < \frac{\overline{W}_{RLA}}{\overline{RTT}_{RLA}} < \frac{\sqrt{3n}\,\overline{W}_{WTCP}}{\overline{RTT}} \tag{6}$$

That is, the long term average throughput of the multicast sender is no less than a third of the TCP throughput on the soft bottleneck branch, and no more than $\sqrt{3n}$ times that. Therefore, the RLA is essentially fair to TCP according to the definition of essential fairness in section 2.

Theorem II: Consider a restricted topology with drop-tail gateways and the phase effect eliminated. If there are n receivers persistently reporting congestion and the largest congestion probability is less than 5%, the RLA is essentially fair to TCP with $a = \frac{1}{4}$, and $b = 2n$.

This theorem can be proved similarly to theorem I, using the fact that, with drop-tail gateways and the phase effect eliminated, the competing RLA and TCP traffic see the same congestion frequency. The proof is omitted due to space limitations.

4.3 Remarks

In the above two theorems, we proved that the RLA is essentially fair to TCP with the restricted topology with equal round-trip times. Note that the bounds in the theorems are widely separated for sizable n; this is because they work for all situations including the cases with extremely unbalanced congestion branches. The algorithm actually delivers desirable performance in the following way: if all the troubled receivers have the same degree of congestion, the RLA results in a throughput no larger than four times that of the competing TCP throughput for any n (this can be proved [20]); on the other extreme, if there is one most congested receiver and the other $n-1$ receivers experience only minor congestion just enough to be counted as troubled receivers, the actual throughput of the RLA is close to the upper bound which is in the order of n for drop-tail gateways. That is, the multicast connection on the soft bottleneck branch gets $c = O(n)$ times the smallest throughput among the competing TCP connections. This might be desirable because this

[9] See section 2 for the definition of a and b.

single bottleneck is slowing down the other $n-1$ receivers. If this is not desirable, the RLA can implement an option to drop this slow receiver. For the situations in between the above two extreme cases, the RLA gives reasonable performance; this is demonstrated in the simulation results in section 5.

In summary, the RLA achieves a higher share of bandwidth than the TCPs on the soft bottleneck branches when only a few receivers in the multicast session are much more congested than others. This is reasonable because the multicast session serves more receivers and it should suffer less on a single highly congested bottleneck.

4.4 Multicast Fairness of RLA

The RLA is fair in the sense that the senders of competing multicast sessions between the same sender and receiver group will have the same average *cwnd* in the steady state. Consider a simple case with two competing sessions with n receivers in each session on the same topology of the form in figure 1. The *cwnd*'s of the two senders are correlated random processes. The problem can be modeled as a randomly moving particle on a plane, with x and y axes being the *cwnd* of sender 1 and 2, respectively (see figure 3). This model is a generalization of the deterministic model for unicast congestion control used in [2], where the authors proved that the linear increase/multiplicative decrease scheme converges to the fair operating point. Our algorithm is a generalization of the unicast algorithm to multiple receivers and introduces randomness. We will show that although the *cwnd* does not converge to a single point, the desired operating point (equal share of the bottleneck bandwidth, see figure 3) is a recurrent point and most of the probability mass would be focused on the general area of this point.

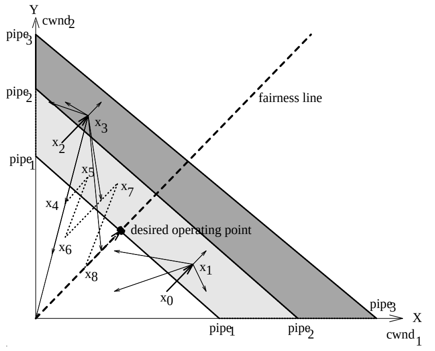

Figure 3: Fairness of RLA to each other.

In our analysis below, we assume there is no feedback delay.[10] Since the two sessions share exactly the same path, the two senders get the same congestion signals. The senders are informed of congestion by receiver i if $cwnd_1 + cwnd_2$ exceeds the pipe size of virtual link L_i which is the largest RTT value times the available bandwidth of the link (see figure 3). Otherwise the sender is informed of no congestion. Focus on the troubled receivers, recalling there are n of them per session. We order the pipe sizes of these troubled links as $pipe_1 < pipe_2 < \ldots < pipe_k$ and there are $n_i (i = 1, \ldots, k)$ receivers with pipe size $pipe_i$ and $\sum_1^k n_i = n$. In figure 3, $cwnd_1 + cwnd_2 < pipe_1$ in the white region with no congestion here; $pipe_1 < cwnd_1 + cwnd_2 < pipe_2$ in the lightly shaded region and the senders receive n_1 congestion signals once they enter this region; and $pipe_2 < cwnd_1 + cwnd_2 < pipe_3$ in the dark shaded region and the senders receive $n_1 + n_2$ congestion signals once they enter this region, and so on... If there is no congestion, both *cwnd*'s increase linearly which results in an upward movement of the particle along the 45^o line (see the movement of x_0 to x_1 in figure 3). In the congested region with a certain number of congestion signals fed to the senders, each sender independently generates a random number to decide whether to increase or cut the window. In our model, the particle randomly chooses one of the moving directions which are combinations of increasing or cutting of each window. For example, in figure 3, assuming $n_1 = 1$, after a round-trip time, $x_1 = (cwnd_1, cwnd_2)$ can move to $(cwnd_1 + 1, cwnd_2 + 1)$, or $(cwnd_1/2, cwnd_2/2)$, or $(cwnd_1 + 1, cwnd_2/2)$, or $(cwnd_1/2, cwnd_2 + 1)$.

Obviously the movement of the particle is Markovian since the next movement only depends on the current location of the particle. From this Markovian model, we can draw several conclusions about the system. First, the desired operating point (see figure 3) is a recurrent point. This is because from any starting point, there exists at least one convergent path to the desired point (e.g., $x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_8 \rightarrow \ldots$ along the dotted line in figure 3) with positive probability. Note that, in our model, there is a positive probability for the *cwnd* to grow to infinity, but this does not happen in the real system because we incorporated a forced-cut mechanism in the algorithm.

Secondly, the average *cwnd*'s of the two senders are the same. This is obvious because the two senders get the same congestion signals and react randomly but identically and independently, that is, the roles of the two senders are interchangeable. In other words, if we switch the x and y axes, the moving particle follows the same stochastic process, and the marginal distributions along the axes are the same which gives the same mean value.

Finally, most of the probability mass would be focused on the general area of the desired operating point in figure 3. To illustrate the idea, we consider a simple case where all of the n links have the same pipe size *pipe*. Then the plane is divided into two regions: a non-congested region with $cwnd_1 + cwnd_2 < pipe$ and the rest a congested region with n congestion signals arriving to the sender upon each packet loss. Since in the RLA, the losses within two RTTs are grouped into one congestion signal, we consider a discrete-time version of the system with the time unit being two round-trip times, i.e., $\Delta t = 2RTT$. Denote *cwnd* of the kth multicast session by W_k, $k = 1, 2$. If there is no congestion, $W_k(t + \Delta t) = W_k(t) + 2$ (because $\Delta t = 2RTT$). Upon congestion with n congestion signals,

$W_k(t + \Delta t) = W_k(t) + 2$, w.p. $(1 - \frac{1}{n})^n =: p_0$.

$W_k(t + \Delta t) = W_k(t)/2^i$, w.p. $\begin{pmatrix} n \\ i \end{pmatrix} (1 - \frac{1}{n})^{n-i} (\frac{1}{n})^i =: p_i$

for $i = 1, \ldots, n$, and $k = 1, 2$. W_1 and W_2 control the movement of the particle along x and y axes, respectively. The average drift along the x axis is 2 if $W_1 + W_2 < pipe$; or $2 * p_0 - \sum_{i=1}^n (1 - W_1/2^i) * p_i$ if $W_1 + W_2 \geq pipe$. The time unit is $2RTT$. The average drift along the y axis is symmetric and can be obtained by replacing W_1 with W_2 in the above equation. The drift diagram with $n = 3$ and $pipe = 10$ is drawn in figure 4; the drift is scaled down by a factor of 5 to make the picture clear.

The drift diagram shows that the particle controlled by the two congestion window sizes along the two axes has a

[10] This assumption is necessary to allow us to use a simple and neat analysis. Our simulation results indicated that the fairness we claimed here still holds when propagation delay is involved.

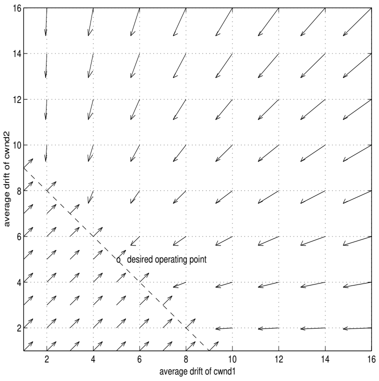

Figure 4: Average drift diagram of two competing $cwnd$'s.

trend to move towards the desired operating point. Figure 5 is the density plot of the occurrence of the point $(cwnd_1, cwnd_2)$ during one simulation run; [11] the higher numbers of occurrence, the darker the area. It shows that most of the probability mass is in an area centered around the desired operating point which is (20, 20) in this case.

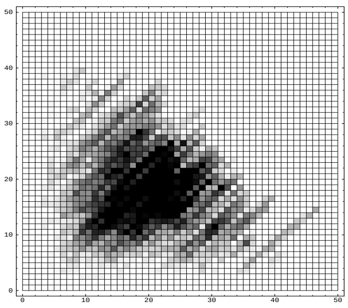

Figure 5: Density plot of the occurrence of $(cwnd_1, cwnd_2)$.

5 Performance Evaluation

A version of the RLA is implemented in Network Simulator (NS2) for simulation purposes to test the RLA performance under various network topologies. Here we present some of the simulation results in a four-level tertiary tree network topology (see figure 6), where the links and nodes are labeled with the first number index indicating their level and the second indicating an order in each level.

We describe most of the simulation parameters used in the simulations shown below. In figure 6, all senders (RLA or TCP) are located at the root node S, all receivers at leaf nodes R1 through R27. The nodes in between are gateways; they could be either drop-tail or RED type. All nodes have a buffer of size 20 packets. In the case of RED gateways,

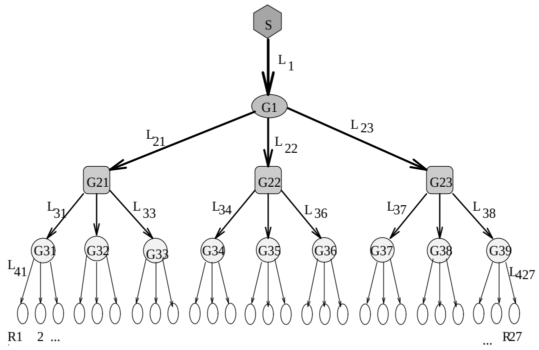

Figure 6: Four-level tertiary tree.

the minimum threshold is 5 and the maximum threshold is 15. (Other parameters are the default values used in the standard NS2.0 RED gateway). The one-way propagation delays of the first three level links are all 5 ms, and those of the last level links are 100 ms. We tested the situations with bottleneck links at different levels to study the effect of independent or correlated losses. All the non-bottleneck link speeds are set to 100 Mbps. Data packet size is set to 1000 bytes. All simulations have 27 receivers (except for the case with different round-trip times), and all receivers are troubled receivers. In the simulation results presented here, $rexmit\_thresh$ is set to 0 (i.e., all retransmissions are multicasted). All simulations are run for 3000 seconds and statistics are collected after the first 100 seconds. The simulation results are briefly summarized in the following.

5.1 Multicast Sharing with TCP

The results for drop-tail gateways are shown in figure 7 and figure 9 for RED gateways. There are 5 cases with different soft bottleneck locations. The second row (most congested links) in the figures indicates the soft bottleneck location. The corresponding link speeds are set so that the soft bottleneck bandwidth share is $\min_i \frac{\mu_i}{m_i+1} = 100$ packets per second (recall m_i is the number of background TCP connections between the sender and receiver i). We list the performance of the RLA, including the average throughput in packets per second, average congestion window size, average round-trip time (for those packets correctly received without retransmissions), the number of congestion signals the multicast sender detected from all receivers, the number of window cuts and the number of forced window cuts, over the entire simulation period (after the first 100 seconds). We also list the worst and the best case TCP performance (WTCP and BTCP rows in the figures) among the competing TCP flows.

As we can see from figure 7, the RLA achieves reasonable fairness with TCP even with drop-tail gateways. Comparing cases 1, 2 and 3, we can see that a higher correlation among the packet losses results in a larger average window size and a higher throughput. This agrees with our Lemma in section 4.2. The throughputs of the RLA and TCP connections in all cases satisfy the essential fairness requirement with $a = 1/4$ and $b = 2n$. In fact, the bounds are quite loose for these cases. The actual performance of the RLA algorithm in most cases is much more "reasonable" than the bounds indicated, in the sense that in most cases the RLA can achieve a tighter bounded fairness. With the simulation setup, the measured essential fairness has bounds $a = 1$ and

[11] The simulation setup consists of two multicast sessions with 27 receivers in each in a topology of the form shown in figure 1. There is one TCP session from the sender node to each of the receiver nodes. All receivers have the same capability. Each path has a delay bandwidth product of 60 shared by 2 multicast and 1 TCP sessions. Therefore, each session is supposed to get an average $cwnd$ of 20.

$b = 3$ in these cases, which is very reasonable performance and acceptable for many applications.

Case		1	2	3	4	5
Most Congested Links		L1	L3i, i = 1, ..., 9	L4i, i = 1, ..., 27	L4i, i = 1, ..., 5	L21
RLA	thrput (pkt/sec)	144.1	105.1	94.6	153.0	224.6
	$\overline{\text{cwnd}}$	33.9	27.2	26.0	40.0	53.7
	$\overline{\text{RTT}}$ (sec)	0.234	0.267	0.270	0.264	0.238
	# cong signals	23247	19797	17007	12759	11754
	# wnd cut	840	719	651	482	442
	# forced cut	0	0	0	0	0
WTCP	thrput (pkt/sec)	81.8	83.0	79.2	68.2	74.5
	$\overline{\text{cwnd}}$	20.2	22.0	22.8	17.9	18.9
	$\overline{\text{RTT}}$ (sec)	0.233	0.251	0.269	0.252	0.238
	# wnd cut	879	722	658	842	899
BTCP	thrput (pkt/sec)	89.6	87.8	80.3	170.7	570.7
	$\overline{\text{cwnd}}$	21.9	23.3	23.2	43.8	134.8
	$\overline{\text{RTT}}$ (sec)	0.233	0.251	0.270	0.244	0.231
	# wnd cut	818	688	646	405	225

Figure 7: Simulation results with drop-tail gateways.

We can also see from figure 7 that the number of window cuts taken by the RLA sender is roughly $\frac{1}{27}$ of the congestion signals the multicast sender detected, as desired. In figure 8, we consider the congestion signals from each receiver separately and list the worst, best and average number of congestion signals the sender detected from each of the receivers on the links with the same level of congestion. The number is over the entire simulation period (2900 seconds). We also list the results for the competing TCP connections. It demonstrates that the TCP sender and the RLA sender see roughly the same number of congestion signals on each branch on average. Therefore, they see the same congestion frequency, as we argued in section 3.1. Note that the discrepancies between the RLA and TCP congestion frequencies are larger in cases 4 and 5. This is because their congestion window sizes are very different (refer to figure 7 for window sizes). In these cases, a larger window likely incurs more losses. Although it breaks the assumption of equal congestion frequencies, it creates a desired balance: the larger the window, the more losses and then the window is more likely to be reduced to half and vice versa. This balance actually helps to achieve a tighter bounded fairness as we observed in the simulations.

	Case	RLA Branch			TCP		
		Worst	Best	Average	Worst	Best	Average
1	all links	861	861	861	879	818	851
2	all links	762	713	707	722	688	709
3	all links	650	609	630	657	646	652
4	more congested	952	925	938	842	819	831
	less congested	384	351	367	413	405	409
5	more congested	1082	1082	1082	899	869	886
	less congested	112	112	112	302	225	271

Figure 8: Statistics of the number of congestion signals.

Figure 9 shows the corresponding results for the RED gateways. All simulation setups are the same except that the gateway type is changed to RED, and we do not use random overhead in these simulations because RED gateways eliminate the phase effect. The results show that with RED gateways, the fairness between multicast and TCP is closer to absolute, especially in case 1. This is expected as suggested by the bounds derived in section 4 and is also intuitive because RED gateways are designed to enforce fairness.

Case		1	2	3	4	5
Most Congested Links		L1	L3i, i = 1, ..., 9	L4i, i = 1, ..., 27	L4i, i = 1, ..., 5	L21
RLA	thrput (pkt/sec)	118.0	103.7	88.3	141.0	209.2
	$\overline{\text{cwnd}}$	27.6	27.0	25.9	36.3	49.6
	$\overline{\text{RTT}}$ (sec)	0.233	0.264	0.283	0.261	0.236
	# cong signals	25272	19188	19895	13939	12132
	# wnd cut	949	729	721	545	454
	# forced cut	0	0	0	0	0
WTCP	thrput (pkt/sec)	84.9	81.7	74.1	67.1	73.1
	$\overline{\text{cwnd}}$	20.7	21.4	21.1	17.3	18.4
	$\overline{\text{RTT}}$ (sec)	0.232	0.249	0.265	0.250	0.236
	# wnd cut	845	741	714	891	902
BTCP	thrput (pkt/sec)	88.9	86.1	74.0	166.2	576.4
	$\overline{\text{cwnd}}$	21.5	22.6	21.1	41.8	135.7
	$\overline{\text{RTT}}$ (sec)	0.232	0.249	0.265	0.243	0.231
	# wnd cut	812	707	702	433	178

Figure 9: Simulation results with RED gateways.

5.2 Multiple Multicast Sessions

To test the multicast fairness property of the RLA algorithm, we have simulated the above scenarios with two overlapping multicast sessions from the sender to the same receivers. In all cases, the two multicast sessions share bandwidth almost equally and have roughly the same average window size. In particular, in the topology of case 3 mentioned above, the two multicast senders achieve throughputs of 65.1 and 65.9 pkt/sec respectively, and average window sizes of 19.9 and 20.1 packets respectively.

5.3 Different Round-Trip Times

This paper has focused on the restricted topology with equal round-trip times where fairness is meaningfully defined. But, in reality, most multicast sessions comprise receivers located at different distances from the sender. These cases have to be addressed properly in order for an algorithm to be accepted. We have a generalized version of the RLA algorithm presented in section 3 to work for the cases with different round-trip times. The basic idea is to set $pthresh = f(\frac{rtt_i}{rtt_{max}})/num\_trouble\_rcvr$. In our experiment, we are using the function of the form $f(x) = x^2$, because it has been shown that, for TCP-like window adjustment policy, the average throughput is proportional to $(\overline{RTT})^k$, where $1 \leq k < 2$ and $k = 2$ if there are no queueing delays [5, 10]. Note that in the case of equal round-trip times, the above $pthresh$ is the same as in the original RLA. In the case of different round-trip times, the receiver with a smaller round-trip time has a much smaller $pthresh$, that is, a much larger fraction of the congestion signals is ignored.

In the case of different round-trip times, we are not able to provide any theoretical proofs of bounded fairness. But our initial experimental results show the generalized algorithm is promising in providing a reasonable share of bandwidth among multicast and TCP traffic. Here we present a set of simulation results with the same topology as in the above simulations but adding the nodes G31 through G39 also as receivers which are of significantly different round-trip times from the leaf nodes since the level four links have a one-way propagation delay of 100 ms. Here we show simulation results for two cases with the bottlenecks at level 2 links or level 3 links respectively. Both cases have a total of 36 receivers, all troubled receivers. The results are summarized in figure 10 and they show a reasonable share of bandwidth between the multicast and TCP traffic.

Case	Most Congested Links	RLA						WTCP				BTCP			
		thrput (pkt/sec)	$\overline{\text{cwnd}}$	$\overline{\text{RTT}}$ (sec)	# cong signals	# wnd cut	# forced cut	thrput (pkt/sec)	$\overline{\text{cwnd}}$	$\overline{\text{RTT}}$ (sec)	# wnd cut	thrput (pkt/sec)	$\overline{\text{cwnd}}$	$\overline{\text{RTT}}$ (sec)	# wnd cut
1	L2i, i = 1,...,3	167.6	39.1	0.24	32118	609	0	78.0	19.7	0.238	856	83.2	20.8	0.238	814
2	L3i, i = 1,...,9	161.6	36.5	0.264	41175	721	0	64.2	17.4	0.253	879	67.7	18.2	0.253	844

Figure 10: Results with different round-trip times.

6 Conclusions and Future work

In this paper, we introduced a quantitative definition for essential fairness between multicast and unicast traffic. We also proposed a random listening algorithm (RLA) to achieve essential fairness for multicast and TCP traffic over the Internet with drop-tail or RED gateways. RLA is simple and achieves bounded fairness without requiring locating the soft bottleneck links. Although our RLA is based on the TCP congestion control mechanism, it is worth noting that the idea of "random listening" can be used in conjunction with other forms of congestion control mechanism, such as rate-based control. The key idea is to randomly react to the congestion signals from all receivers and to achieve a reasonable reaction to congestion on the average over a long run. There are many interesting possibilities which are worth exploring in this direction.

Due to space limitations, many details of the algorithm and simulation results are not shown in this paper. Our ongoing work is to carry out more and larger scale simulations and to refine the algorithm based on the experience gained from the simulations.

Acknowledgement

This work was inspired by a summer project the first author worked on at Lucent Bell Labs. She would like to thank Dr. Zheng Wang for the basic inspiration. Many thanks are also due to Dr. Sanjoy Paul, Dr. Ramachandran Ramjee, Dr. Jamal Golestani, all of Bell Labs, for useful discussions.

References

[1] F. Brockners. Notes on FEC supported congestion control for one to many reliable multicast. Working Draft, http://www.zpr.uni-koeln.de/~frank/doc/, Sept. 1997.

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

[3] D. DeLucia and K. Obraczka. A congestion control mechanism for reliable multicast. RM meeting, Sept. 1997. http://www.east.isi.edu/rm/delucia.ps.

[4] K. Fall and S. Floyd. Simulation-based comparisons of tahoe, reno, and sack tcp. *ACM Comput. Commun. Review*, 26(3):5–21, July 1996.

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

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

[7] Mark Handley. A congestion control architecture for bulk data transfer, Sept. 1997. available from http://www.east.isi.edu/rm/cannes-meeting.html.

[8] IRTF. Minutes of the reliable multicast irtf research group meeting, Sept. 1997. available from http://www.east.isi.edu/rm/notes-rev0.html.

[9] V. Jacobson. Congestion avoidance and control. In *Proc. SIGCOMM'88*, pages 314–329, 1988.

[10] T.V. Lakshman and U.Madhow. The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Networking*, 5(3):336–350, June 1997.

[11] J. Mahdavi and S. Floyd. Tcp-friendly unicast rate-based flow control. Technical note sent to the end2end-interest mailing list, 1997. Available via http://www.psc.edu/networking/papers.

[12] M. Mathis, et. al. Tcp selective acknowledgement options, April 1996. RFC 2018.

[13] T. Montgomery. A loss tolerant rate controller for reliable multicast. Technical Report NASA-IVV-97-011, West Virginia University, August 1997.

[14] T.J. Ott, J.H.B. Kemperman, and M. Mathis. The stationary behavior of ideal tcp congestion avoidance. Obtain via pub/tjo/TCPwindow.ps using anonymous ftp to ftp.bellcore.com, August 1996.

[15] T. Sano, N. Yamanouchi, T. Shiroshita, and O. Takahashi. Flow and congestion control for bulk reliable multicast protocols – toward coexistence with tcp. RM meeting, Sept. 1997. Available via http://info.isl.ntt.co.jp/chisho/rmtp/infocomm98.ps.gz.

[16] D. Sisalem, F. Emanuel, and H. Schulzrinne. The direct adjustment algorithm : A tcp-friendly adaptation scheme. Preprint, August 1997.

[17] UCB/LBNL/VINT. Network simulator. a discrete event simulator targeted at networking research, available at http://www-mash.cs.berkeley.edu/ns/ns.html.

[18] L. Vicisano and J. Crowcroft. One to many reliable bulk-data transfer in the mbone. In *Proc. HIPPARCH'97*, Uppsala, Sweden, 1997.

[19] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Proc. INFOCOM'98*, CA, April, 1998.

[20] H. Wang and M. Schwartz. Achieving bounded fairness for multicast and TCP traffic in the Internet. Complete version. Available via http://www.ctr.columbia.edu/~whycu/res.html.

[21] L. Zhang, S. Shenker, and D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *Proc. SIGCOMM'91*, pages 133–147, 1991.