Wave and Equation Based Rate Control Using Multicast Round Trip Time*

Michael Luby Vivek K Goyal Simon Skaria Gavin B. Horn
{luby,vivek}@digitalfountain.com sskaria@ics.uci.edu gavin@alumni.caltech.edu
Digital Fountain, Inc. Univ. of California, Irvine Pulsent Corporation

ABSTRACT

This paper introduces Wave and Equation Based Rate Control (WEBRC), the first multiple rate multicast congestion control protocol to be equation based. The equation-based approach enforces fairness to TCP with the benefit that fluctuations in the flow rate are small in comparison to TCP.

This paper also introduces the multicast round trip time (MRTT), a multicast analogue of the unicast round trip time (RTT). The MRTT is fundamental to the equation-based protocol that each receiver uses to adjust its reception rate. Each receiver independently measures its own MRTT without placing any added messaging burden on the receiver, the sender or the intermediate network elements. Benefits provided by the MRTT include those that the RTT provides to TCP, *e.g.*, reduced reception rates in reaction to buffer filling and fair sharing of bottleneck links. In addition, the use of MRTT is shown to synchronize and equalize the reception rates of proximate receivers and to cause reception rates to increase as the density of receivers increases.

Another innovation of WEBRC is the idea of transmitting data with waves: the transmission rate on a channel is periodic, with an exponentially decreasing form during an active period followed by a quiescent period. Benefits of using waves include insensitivity to large IGMP leave latency; a frequency of joins and leaves by each receiver that is small and independent of the receiver reception rate; the use of a small number of multicast channels; fine-grained control over the receiver reception rate; and minimal, at times nonexistent, losses due to buffer overflow.

Categories and Subject Descriptors

C.2.2 [**Communication Networks**]: Network Protocols

*This work was completed at Digital Fountain, Inc., 39141 Civic Center Drive, Suite 300, Fremont, CA 94538. S. Skaria and G. B. Horn were with Digital Fountain during their involvement in this work.

SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.

General Terms

Algorithms

Keywords

congestion control, multicast, multiple-rate, TCP-friendliness

1. INTRODUCTION

This paper introduces a natural notion of a multicast round trip time (MRTT) and an equation-based, multiple rate congestion control protocol—Wave and Equation Based Rate Control (WEBRC)—that uses the MRTT as a substitute for conventional unicast round trip time (RTT). Because the throughput of a TCP session depends strongly on RTT, having some measure of RTT is essential in making an equation-based rate control protocol "TCP-friendly." The lack of a suitable RTT measure is thus a major reason that previous multiple rate congestion control protocols have not been equation based and have not been fair to TCP over a wide range of RTTs.

In TCP, the RTT is not measured explicitly, except for setting some exceptional timeout values. Nevertheless, the mechanics of TCP make it reactive to changes in RTT. Since increases in RTT are often due to increasing buffer occupancy, it is good for network stability that the derivative of the packet injection rate of a TCP flow decreases as RTT increases. MRTT similarly increases with increasing queuing delay. In addition, the properties of MRTT within the context of WEBRC tend to equalize the reception rates of proximate receivers and increase receiver reception rates as the density of receivers increases.

Another major innovation of WEBRC is the transmission of packets in *waves*. Each WEBRC session uses several channels with packet injection rates that vary in time. There is one base channel with a low, fairly constant rate and several wave channels. The rate of each wave channel has the shape of a repeating wave. The rate quickly increases to a high peak value at the beginning of each wave and then decreases smoothly and gradually until a point where it drops to zero and stays at zero for some period of time. This wave pattern repeats itself again and again on each channel, and the starting times of the patterns on the different channels are equally spaced as shown in Fig. 1.

WEBRC is the first multiple rate congestion control protocol to be equation based. It draws heavily on the ideas in TFRC [5, 6]. At any point in time, a receiver is receiving packets from the base channel and some number of consecu-

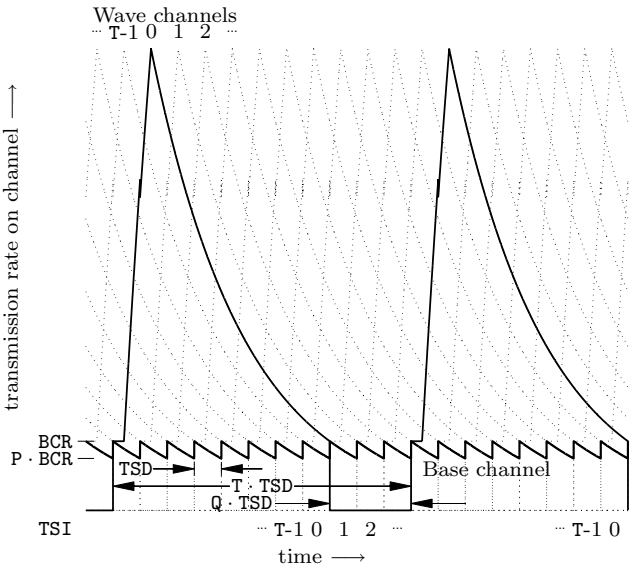


Figure 1: Rates on WEBRC channels under a fluid transmission model. The base channel rate has small variation over a period of `TSD` seconds. Each of `T` wave channels has much more variation over a period of `T·TSD` seconds. Each wave channel is quiescent for `Q · TSD` seconds in each period.

tive waves that are the lowest-rate active waves at that moment. The number of waves depends on the receiver's current target reception rate. The receiver continually updates its target reception rate using its measured loss event rate and MRTT. Neglecting losses, joining the next higher-rate wave increases the reception rate by a known multiplicative factor and refraining from joining allows the reception rate to decrease exponentially. These mechanisms lead to very fine-grained control of the reception rate.

WEBRC has several advantages over previous multiple rate congestion control protocols such as Receiver-driven Layered Congestion Control (RLC) [18] and Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) [1]. The key advantages are that the frequency of joins and leaves by each receiver is small and independent of the receiver reception rate, the number of multicast channels used is small, the receiver reception rate control is fine-grained, losses due to buffer overflow are minimal (potentially nonexistent) and fairness to TCP is very good.

RLC and FLID-DL place explicit signals into packets to indicate to receivers when they may increase their reception rates. This keeps the increase linear on average despite geometrically increasing reception rate steps. One issue with this step-based approach to congestion control is that, because increases in reception rate are in geometrically increasing steps, packet loss can be substantially higher than for a TCP session with a similar average reception rate. In contrast, the wave- and equation-based approach that WEBRC uses ensures that packet loss rates are similar to those for a TCP session with a similar average reception rate.

With FLID-DL the amount of IGMP and PIM SM control traffic generated by each receiver is so high that it is considered to be impractical to implement in many networking architectures. Because of the use of waves, WEBRC can use time slots that are 20 times as long as FLID-DL times slots (10 sec instead of 0.5 sec) and thus the frequency of IGMP join and leave messages is reduced by a factor of 20.

The resources the sender uses to support a WEBRC session are the same independent of the number of receivers in the session. Furthermore, the reception rate of each receiver in a session adjusts to the available bandwidth between the receiver and the sender, and thus receivers on paths with more available bandwidth are not slowed down by receivers on paths with less available bandwidth.

A significant impediment to the universal adoption of IP multicast is the lack of a standardized congestion control protocol that is fair to TCP flows and other flows with similar average throughput, produces only a similar number of packet losses as a concurrent TCP flow, places minimal multicast message processing burden on routers, has high bandwidth utilization, and enables unlimited scalability by completely avoiding any per-receiver state in the sender. WEBRC provides all of these properties, and thus it is a candidate protocol for widespread deployment. A WEBRC draft has been submitted to the IETF Reliable Multicast Transport working group as a multiple rate congestion control building block [10].

WEBRC is primarily intended for reliable download applications where the packets sent to the different channels contain encoded material that is useful to the receiver to recover the original content, independent of which packets are received [2, 18]. Thus, WEBRC is a suitable congestion control protocol to use with the ALC approach to reliable content delivery [8]. ALC combines the LCT building block [9] with the FEC building block [12]. The LCT block provides higher-level session support, whereas the FEC block provides coding that enables reliable content delivery as described in [13]. WEBRC can also be used for unicast and for other types of applications, such as streaming.

2. EQUATION BASED RATE CONTROL

It is well understood that the use of end-to-end congestion control is essential for the stability of the Internet [4]. For unicast applications, end-to-end congestion control is usually provided by the TCP congestion avoidance protocol. Because of the prevalence of TCP, unicast UDP connections are often asked to be "fair" to TCP. A detailed discussion of fairness and a survey of techniques can be found in [19].

In TCP, congestion control is entangled with the reliability mechanism. Reliability is based on maintaining within the sender's *congestion window* buffer any data that has not yet been acknowledged by the receiver, and rate control is achieved by varying the size of the congestion window; thus, TCP congestion control is called *window-based*. After a great deal of analytical and empirical study, the long-term average throughput of a TCP connection is well-understood [14, 16]. Let p denote the packet loss probability and t_{RTT} denote the round trip time (RTT) of a TCP session. The TCP throughput R in packets per second is well-approximated by

$$R = \frac{\sqrt{3/2}}{t_{\mathrm{RTT}}\sqrt{p}\,(1+9p(1+32p^2))}. \qquad (1)$$

This simplification of the full equation [16, Eq. (29)] is obtained by making the standard assumptions that the TCP

congestion window size evolves without any preset maximum, the duration of the TCP sender's time out counter is set to $4t_{\mathrm{RTT}}$ and one packet is acknowledged with each ACK transmission, and pessimistically simplifying $\min\{1, 27p/8\}$ to be equal to $27p/8$. The equation

$$R = \frac{\sqrt{3/2}}{t_{\mathrm{RTT}}\sqrt{p}} \tag{2}$$

provides a good approximation of Eq. (1) for all but the largest values of p and we use this simplified equation in some of our analysis. Controlling the transmission rate to match these or any other equation is called *equation-based* rate control.

Equation-based rate control separates the calculation of a target long-term average transmission rate from the mechanics of transmission. This introduces design freedom that can be used in a variety of ways. For example, in streaming media applications, temporary reductions in transmission rate can cause the decoder's playback buffer to underflow. When such transient throughput changes are inevitable, as with TCP, avoiding underflow implies a long playback delay and thus large memory requirements and poor interactivity. With equation-based rate control, the transmission rate can be made to only slowly vary by using long-term averages for the equation parameters (*e.g.*, p and t_{RTT} in Eq. (1)). More generally, the dynamics of a connection can be designed based on the content type and other criteria.

The most mature equation-based rate control protocol is the TCP-Friendly Rate Control (TFRC) of Floyd *et al.* [5, 6]. TFRC is described with respect to a unicast implementation, but the original motivation for the work was to design a multicast congestion control protocol. With TFRC, the sender adjusts its transmission rate to match Eq. (1) using its own estimate of t_{RTT} and an estimate of p fed back by the receiver. The use of Eq. (1) means that, assuming the estimates of t_{RTT} and p approximate the RTT and packet loss probability that a TCP connection would experience under the same circumstances, the TFRC connection has the same long-term throughput as a TCP connection.

3. WEBRC OVERVIEW

WEBRC congestion control is achieved by having the sender send packets within a given session to several different channels. Individual receivers dynamically join and leave these channels according to the network congestion they experience. The channels associated with a session consist of a base channel identified with the channel number $\mathtt{CN} = \mathtt{T}$ and $\mathtt{T}$ wave channels with channel numbers $\mathtt{CN} = 0, 1, \ldots, \mathtt{T}-1$. At the sender, time is partitioned into time slots, each of duration TSD seconds, where the recommended value for TSD is 10. There are T time slot indices associated with a session. As time progresses, the time slot index increases by one modulo T each TSD seconds.

The output of the server, totaled over all of the channels, is at the constant rate of SR_b bps.[1] The server output is split amongst the channels, and the rates on the channels vary over time. For the base channel the variation in rate is mild. Packets are sent to the base channel at a low rate BCR_P at the beginning of a time slot and this rate exponentially decreases to $\mathtt{P} \cdot \mathtt{BCR\_P}$ at the end of the time slot; $\mathtt{P} < 1$ is a constant with a recommended value of 0.75. This pattern for the base channel repeats over each time slot, as shown in Fig. 1.

For each wave channel i, packets are sent starting at rate BCR_P. The rate rises quickly to a high rate and from this point decreases by a fixed fraction P each TSD seconds until the rate BCR_P is reached at the end of time slot i. Then, for a period of Q times slots called the quiescent period, no packets are sent to wave channel i, and thereafter the whole cycle repeats itself, where the duration of the cycle is $\mathtt{T} \cdot \mathtt{TSD}$ seconds. Thus, the wave channels are going through the same cyclic pattern of packet rate transmission spaced out evenly by TSD seconds. The beginning of the wave is designed to ensure that the minimum rate on a wave channel while it is active is BCR_P and the total sending rate across all of the channels is SR_P at any point in time. Equations for the wave channel rate under a fluid model and details on scheduling discrete transmission events using the fluid model are given in [10, 11].

The number of channels depends on the range of reception rates to be supported. For the sum of the rates of the active channels to reach SR_P, there must be

$$\mathtt{N} = \left\lceil \log_{1/\mathtt{P}} \left(1 + \frac{1}{\mathtt{P}} \left(\frac{1}{\mathtt{P}} - 1 \right) \frac{\mathtt{SR\_P}}{\mathtt{BCR\_P}} \right) \right\rceil - 1 \tag{3}$$

channels active at a time. N is also the duration of a wave measured in time slots. The number of wave channels is $\mathtt{T} = \mathtt{N} + \mathtt{Q}$.

Each packet contains congestion control header information consisting of the channel number CN, time slot index TSI, and packet sequence number PSN. The PSN is a numbering of packets within a channel, increasing by 1 modulo 2^{16} with each packet. For wave channels, the last packet before the quiescent period has $\mathtt{PSN} = 2^{16} - 1$ and the first packet of each wave is numbered accordingly. A WEBRC receiver uses CN to identify packets with channels, TSI to identify time slot changes, and PSN to detect packet losses. Having a known value for the last packet of each wave allows packet losses at the ends of waves to be detected.

Before joining a session, a receiver must obtain a session description that includes the session parameters needed to perform WEBRC congestion control. Implementations may hold certain parameters constant or in fixed relations to reduce the length of the session description. We assume that the sender and receiver share knowledge of all of the "Session parameters" in Table 1. Once the session parameters are known, congestion control adjustments are performed at each receiver independently of all other receivers and without any impact on the sender. Most of the complexity of the protocol is in the state variables that each receiver tracks and the receivers' decisions on when to join wave channels.

When a receiver enters a session it first joins the base channel, and it remains joined to the base channel for the duration of its participation in the session. The receiver is normally receiving several waves at once. The first wave channel that the receiver joins is the one that is nearest the end of its active wave. Since the ordering of the rates of the wave channels depends on the time slot index, the receiver must receive at least one base channel packet to obtain the current TSI before increasing its rate by joining wave channels. Subsequently, throughout the session the receiver joins wave channels in sequence—joining the next higher-rate wave—and leaves wave channels when they be-

[1] The suffix _b indicates that the rate is in bps. Rates in packets per second have the suffix _P.

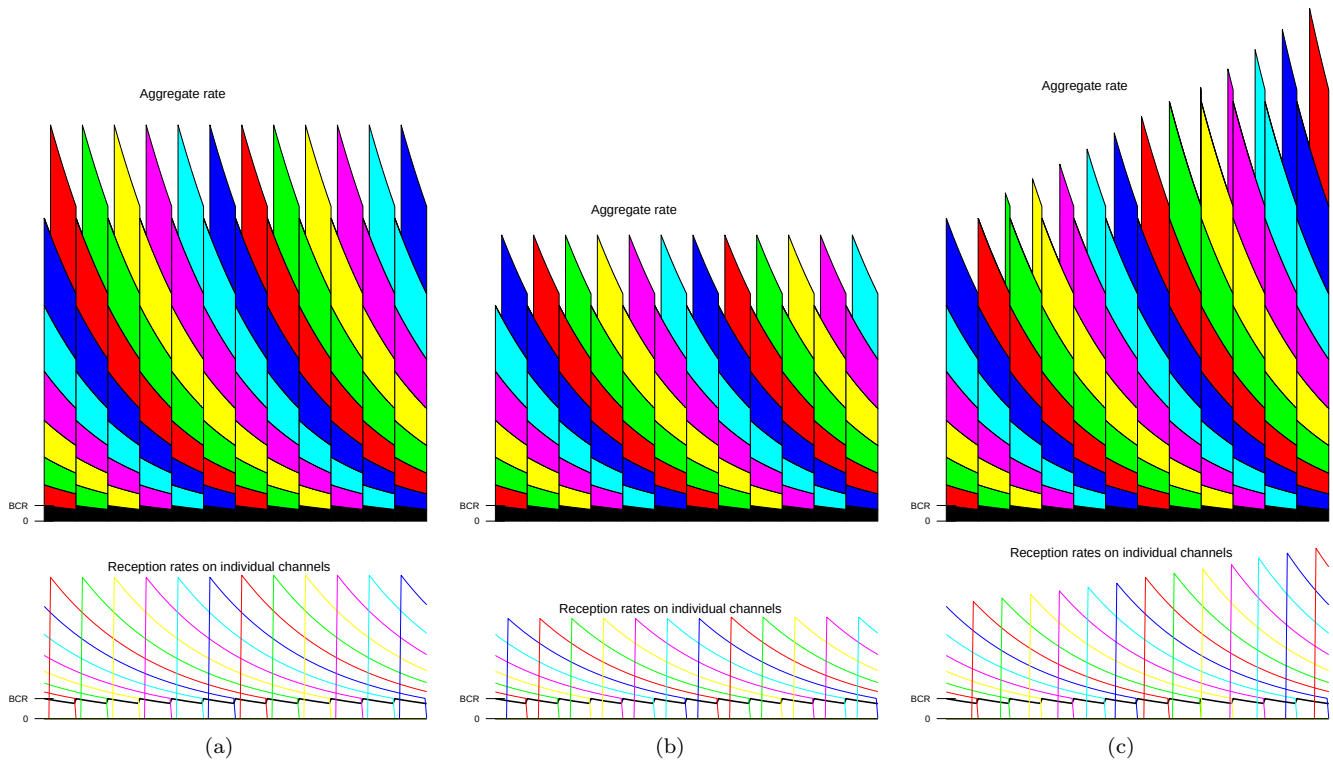


Figure 2: How a WEBRC receiver adjusts its reception rate as a function of time. The receiver joins the wave channels in succession, catching the end of every wave. The amount of each wave that it catches determines the aggregate rate. (a) Catching the same part of each wave results in constant average throughput. (b) Catching slightly less of each wave results in a lower aggregate rate. (c) A slow increase in reception rate corresponds to gradually joining the waves at earlier points in their descents.

come quiescent.[2] The receiver knows when channels become quiescent by monitoring the `TSI`s of incoming packets.

The net effect of these rules for which wave channels are joined is that the receiver "catches" the end of every wave. The reception rate at the receiver is determined by how early each wave channel is joined by the receiver: the earlier the receiver joins a channel with respect to when its wave started, the higher the reception rate. The receiver has fine-grained control over when it joins waves and hence there is fine-grained control over the average reception rate. For example, with the recommended values $\mathtt{P} = 0.75$ and $\mathtt{TSD} = 10$, catching the last 50 seconds of each wave instead of the last 49 seconds increases the rate by about 1.5%.

Fig. 2 shows receiver reception rates on individual channels and the resulting aggregate reception rate. Parts (a) and (b) illustrate that to maintain a steady average reception rate, the receiver joins each successive wave at the same point in its descent; for a lower reception rate the waves are joined later. Part (c) shows that each wave at a slightly earlier relative time than the previous wave slowly increases the average reception rate.

Full details on the receiver logic are given in [10] and, in addition, the reasons for many design choices are described in [11]. Here we are only able to summarize the receiver operation. The normal, equation-based operation is described next. Following that are descriptions of session start-up and a refinement that attempts to stabilize operation based on queue occupancy rather than packet losses.

3.1 Equation-based receiver operation

The way the receiver adjusts its reception rate is inspired by TFRC. The receiver maintains a target reception rate and is allowed to join the next wave channel if after joining, its reception rate would at most equal its target reception rate. The decision to join or not join is made periodically, at the end of each *epoch*. This decision is facilitated by the shapes of the waves. Regardless of the time within a time slot, increasing the number of joined, active waves from `NWC` to $\mathtt{NWC} + 1$ increases the reception rate by a multiplicative factor

$$\Gamma_{\mathtt{NWC}} = \frac{(1/\mathtt{P})^{\mathtt{NWC}+2} - 1}{(1/\mathtt{P})^{\mathtt{NWC}+1} - 1} \approx \frac{1}{\mathtt{P}}. \quad (4)$$

Thus the next wave is joined if $\Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P}$ is at most the target rate, where `ARR_P` is the receiver's estimate of the aggregate rates of the subscribed channels.[3]

The target rate is continually updated based on a set of measured parameters including the average loss probability

[2]The quiescent period exists so that even a large IGMP leave latency will not cause a wave channel to become active again before the leave takes effect.

[3]The receiver is always disallowed from joining if the previous join has seemingly not taken affect, *i.e.*, no packets from been received from the previously joined channel. A timeout mechanism is used to detect failed joins and try again.

Description	Name	Value
Packet header fields:		
Channel number	CN	
Time slot index	TSI	
Packet sequence number	PSN	
Session parameters:		
Session rate	SR_b	
Base channel rate	BCR_P	1.0
Packet payload length (bytes)	LENP_B	1024
Rate decrease factor (per TSD sec)	P	0.75
Time slot duration (sec)	TSD	10
Quiescent time slots	Q	$\lceil 300/\mathtt{TSD} \rceil$
Active time slots	N	Eq. (3)
Time slot indices, wave channels	T	N + Q
Receiver parameters:		
Epoch length (sec)	EL	0.5
Maximum reception rate	MRR_b	
Receiver variables:		
Averaged multicast round trip time	ARTT	
Estimated loss event rate	LOSSP	
Anticipated reception rate (averaged IRR_P adjusted for joins and leaves)	ARR_P	
True reception rate (averaged RR_P)	TRR_P	
Reception rate (current epoch)	RR_P	
Intended reception rate (received and lost packets in current epoch)	IRR_P	
Slow start rate	SSR_P	
Number of subscribed wave channels	NWC	
Rate increase factor from joining	$\Gamma_{\mathtt{NWC}}$	Eq. (4)
Target rate	TRATE_P	Eq. (5)
Equation rate	REQN_P	Eq. (6)

Table 1: Summary of terms

and the average MRTT. The target rate is given by

$$\mathtt{TRATE\_P} = \min\left\{\max\{\mathtt{SSR\_P}, \mathtt{REQN\_P}\}, \Gamma_{\mathtt{NWC}}^{3/2} \cdot \mathtt{TRR\_P}, \mathtt{MRR\_P}\right\} \quad (5)$$

where

$$\mathtt{REQN\_P} = \frac{\sqrt{3/2}}{\mathtt{ARTT}\sqrt{\mathtt{LOSSP}}\,(1 + 9\,\mathtt{LOSSP}(1 + 32\,\mathtt{LOSSP}^2))}. \quad (6)$$

These two equations capture many of the principles of WEBRC. First, the equation-based rate REQN_P is an analogue of Eq. (1) where ARTT is an average of the receiver's measurements of MRTT and LOSSP is the receiver's estimate of the loss event rate.[4] (Computations of these averages are described in the Appendix.) Thus, the computed rate is exactly the target rate suggested in TFRC with MRTT playing the role of RTT. Then, Eq. (5) adjusts this rate so that it is at least as large as a slow start rate SSR_P and not larger than a maximum receiver reception rate MRR_P or a multiple of the current reception rate TRR_P. The slow start rate is used as in TCP as a rate threshold below which fast increase of the reception rate is acceptable. It is set below TRR_P when packet losses are detected. Limiting the target rate with an appropriate multiple of the true reception rate prevents the receiver from attempting to increase its reception rate when buffers are filling without packets being lost.

WEBRC receivers below a common bottleneck coordinate through different means than in RLC and FLID-DL. Both RLC and FLID-DL send explicit increase signals in packets to coordinate when receivers are allowed to increase their reception rates. Instead, WEBRC receivers coordinate their reception rates indirectly through the convergence of their target rates. A receiver with a higher target rate joins a wave channel earlier than a receiver with a lower target rate. Since the rate of the wave is decreasing continually over time, packet loss typically occurs in a short interval of time after the first join to a wave is made, and thus the higher rate receiver will tend to experience more packet loss than the lower rate receiver. Because the target reception rate of each receiver is inversely related to the square root of its loss rate, this will tend to make the target reception rates of the two receivers converge. The worst case for this convergence mechanism is for all the packet losses to be common to both receivers. Even then, the dependence of the target rate on the loss rate tends to drive the reception rates of the receivers together. As we describe later in more detail, a higher rate receiver with an earlier join time will tend to have a larger measured MRTT value than a lower rate receiver with a later join time. Because the target reception rate is inversely related to MRTT, this will also tend to make the target reception rates of the two receivers converge.

[4]As in TFRC, a loss event is a period of duration ARTT starting with the detection of a packet loss.

Trigger	Actions
Epoch boundary	Update ARR_P and TRR_P. Compute LOSSP, REQN_P and TRATE_P. If not in a loss event, not waiting for a join, and $\Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P} \leq \mathtt{TRATE\_P}$ then join the next wave.
– Join	Update ARR_P and increment NWC.
Packet reception	Check for new time slot. Increment counter for RR_P. Check for packet loss. Update LOSSP-related variables. If packet is from new wave, update ARTT
– New time slot	Leave newly quiescent wave channel. Update ARR_P and decrement NWC.
– Packet loss	Increment counter for IRR_P. Initiate loss event if necessary.

Table 2: Essentials of receiver operation

The basic actions of the receiver are summarized in Table 2. This excludes initializations, accounting for the finite number of active channels, join timeouts, and the mechanism described in Section 3.3. These and other details have been omitted to allow a more complete discussion of MRTT and simulation results.

3.2 Receiver operation for session start-up

When it first enters a session, a receiver does not have enough information to produce meaningful values of ARTT and LOSSP. Furthermore, it is desirable for the receiver to quickly determine and utilize the available bandwidth between the sender and the receiver. Therefore the receiver starts with $\mathtt{SSR\_P} = \infty$ and uses a modified version of Eq. (5) where the exponent for TRR_P is 3. With no further modifications, the target rate is very likely to be large enough to justify joining an additional wave channel at each epoch boundary and the reception rate would increase exponentially, approximately by a factor 1/P per epoch (4/3 per 0.5 seconds, with the recommended parameter values).

Because WEBRC is not very reactive in reducing its rate—relying solely on the $\mathtt{P}^{t/\mathtt{TSD}}$ decay of wave channel transmission rates with time—it is more important for WEBRC to avoid overshooting the available bandwidth than for more

reactive protocols such as TCP.[5] A WEBRC receiver uses two mechanisms to attempt to identify that it has subscribed to too many wave channels, *i.e.*, that the sum of the rates of subscribed channels exceeds the available bandwidth, *before the first packet loss is experienced*. If either of these detects that the available bandwidth has been reached or if a packet loss is detected, `SSR_P` is set in proportion to the current reception rate `TRR_P` and the `LOSSP` tracking is initialized in such a way that `REQN_P` equals `TRR_P`, *i.e.*, Eq. (6) is solved for `LOSSP`. With `SSR_P` no longer infinite, the regular equation-based rules apply for the remainder of the session.

The first mechanism is to compare successive MRTT measurements. Neglecting competing traffic, changes in multicast trees, and equipment variation, a large increase in MRTT upon joining channel $i+1$ reveals that buffer occupancies have increased since channel i was joined, so the rate increase from joining channel i is not supported by the available bandwidth. A certain MRTT increase may be due to the interpacket spacing on the wave channels; a threshold based on the interpacket spacing determines when an increase is considered "large."

The second mechanism has the potential to detect that joining channel i has increased the subscribed rate above the available bandwidth before channel $i+1$ is joined. Once packets start arriving with $\texttt{CN} = i$, the reception rate should approximately agree with the receiver's estimate of the subscribed rate, `ARR_P`. If the reception rate lags behind `ARR_P`, it is an indication that the available bandwidth has been reached and packets from the session are accumulating in buffers. While starting up a session, the receiver thus measures the reception rate `RR_P` on a full epoch after a join has taken effect and checks that `RR_P` is close to `ARR_P` before joining another wave channel. This slows the rate at which wave channels are joined to at most once per two epochs.

3.3 Avoiding buffer overflows

The use of waves in WEBRC provides an opportunity to sometimes stabilize the reception rate without packet losses or explicit congestion notification. Since the transmission rates on the wave channels are decreasing as $\texttt{P}^{t/\texttt{TSD}}$, a receiver can infer from a constant reception rate that the available bandwidth is being used and that the reception rate reflects the rate at which buffers are emptying—not the actual transmission rate. Joining another wave channel would then not increase the reception rate but rather overload the path from sender to receiver. To prevent this, receivers are disallowed from joining when `RR_P` is not below (by some margin) the maximum `RR_P` since the last join. If the receiver would have otherwise joined but this rule is invoked, `LOSSP` is reset in proportion to the current target rate.

This rule is most likely to come into effect when a WEBRC session is the only flow over a bottleneck link and when the amount of available buffering is relatively large. When this rule comes into effect, the tendency is to wait for empty queues before joining and thus avoid buffer overflows. This is in contrast to the usual stabilizing pressure that comes from packet losses, which implies that the queue occupancy is high. In particular, this enhancement to the equation-based approach may make WEBRC sessions less of a hindrance to short-lived TCP flows. It also should be noted that the manipulation of `LOSSP` makes it less closely reflect the actual packet loss fraction of the session. However, there is no inherent value in the accuracy of `LOSSP`; rather, it is desirable for the target rate to match the available bandwidth.

[5]Note that the problem of large IGMP leave latencies that motivates the quiescent periods on the wave channels also prevents multicast leave operations from being a robust way to reduce reception rate.

4. MULTICAST ROUND TRIP TIME

One of the main impediments to multiple rate congestion control has been the lack of a counterpart to the TCP notion of the RTT. Although the RTT is not explicitly measured in TCP except for the purpose of setting the time for some exceptional timeout values, the observed behavior of TCP can in many respects be modeled using the RTT. The most important influence of the RTT is that the sending rate of a TCP session is inversely proportional to the RTT. This is important for several reasons, as outlined below.

In this section a natural multicast analogue of the unicast RTT is introduced, hereafter called the multicast round trip time (MRTT). Through the use of Eqs. (5) and (6), the target reception rate of each WEBRC receiver is inversely related to its MRTT. The sensitivity of WEBRC to MRTT provides the same loss-reducing benefits as RTT sensitivity provides to TCP. The use of the MRTT tends also to equalize receiver reception rates.

4.1 The role of RTT in TCP

When a buffer along the path from a TCP sender to the TCP receiver starts filling up, the RTT of the session increases commensurately. Due to the influence of the RTT on the sending rate, this increase in the RTT causes the rate of increase in the TCP sending rate, and possibly the sending rate itself, to slow down. This in turn both delays the buffer overflow and causes it to be less severe when it occurs. This provides rationale for the reception rate to be inversely related to the RTT.

When there is a bottleneck link that is shared by several sessions, the question of what share of the bandwidth over the link is *fair* to each session is an interesting one. One way to answer this question is to say that the bandwidth allocation is fair if it is shared by each session in proportion to its *utility*, where the *utility* of a session is the ratio of the *benefit* to the *cost* of the session. For TCP, the cost can be defined as the total time spent by network elements to process a packet, and thus the cost is the RTT. Since each packet is delivered to one receiver, the benefit can be defined to be one. Thus, the utility of a TCP session is inversely proportional to the RTT. Because the rate of a TCP session is inversely proportional to its RTT, competing TCP sessions do share a bottleneck link fairly with respect to this definition of utility. This is another justification for the receiver reception rate to be inversely related to the RTT.

Other definitions of utility are possible, and in particular although the definition given above is natural, it does have the weakness that when TCP is deployed in a WAN environment the discrepancy in rates between competing flows with differing RTTs is very high.

4.2 MRTT measurement

A WEBRC receiver makes a measurement of its MRTT each time it joins a wave. The MRTT is measured as the time between when a join is sent for a wave and when the first packet is received from that wave. These raw MRTT measurements are averaged to obtain a value used in Eq. (6).

For a session with one receiver, the MRTT of the receiver is the RTT between the receiver and the sender, where the RTT is defined to be the time for the join to propagate up to the sender plus the time for the data packet to propagate back down to the receiver.[6] For a session with more than one receiver, the MRTT is a generalization of the unicast RTT that depends on the tree structure and the join times of the other receivers in the session. The more intriguing properties of the MRTT are explored in the following subsections.

4.3 Wave time and join time

To describe the properties of the MRTT, it is useful to introduce the notion of *wave time*. Wave time is a local measure of time at each location in the multicast tree that is relative to the flow of packets from a wave. It is wave time t at a location when packets sent by the sender t seconds after the beginning of the wave would arrive at that location. Thus, for example, if it takes 0.5 seconds for a packet to flow from the sender to a receiver then it is wave time t at the receiver $t + 0.5$ seconds after the wave starts at the sender.

Define the *join time* of a receiver to a wave as the wave time at that receiver when the join is sent for the wave. Thus, a receiver with join time zero joins the wave when the beginning of the wave would arrive at the receiver. Note that all receivers with the same target reception rate will have the same join time to each wave, *i.e.*, they will all join the wave when the wave would reach them at the same point in its descent. In general, receivers that have higher target reception rates will have earlier join times and receivers that have lower target reception rates will have later join times. Thus, the join time is a reflection of the target reception rate of a receiver before the join, and the MRTT value measured as a result of the join impacts the target reception rate after the join.

In the following subsections the relationship between the join times of receivers and the resulting measured MRTT values is described. It should be noted that receivers do not explicitly compute their join times in the WEBRC protocol. Instead, each receiver continually computes its target reception rate and based on this and other measured parameters decides when to join the next wave in its descent. However, it is convenient to analyze the WEBRC protocol via the join time abstraction, and the join times fully capture the relevant information about the actions of the receivers within the context of describing MRTT measurements.

4.4 MRTT global properties

Consider a fixed multicast tree with fixed latencies for all links, a fixed set of receivers and a fixed set of join times for these receivers to the wave. For each receiver i, let N_i be the network element closest to the sender with the property that the join from receiver i is the first to arrive at N_i. Let C_i be the cycle that goes up the tree from i to N_i and then back down the tree to i. Let $\texttt{RTT}_i$ be the RTT of cycle C_i, *i.e.* $\texttt{RTT}_i$ is the time it takes for a join sent from receiver i to propagate up to and establish forwarding state for the wave at N_i, plus the time it takes for a data packet received at N_i to travel to receiver i once the forwarding state has been established. Let $\texttt{JT}_i$ be the join time of receiver i, *i.e.*, $\texttt{JT}_i$ is the wave time at i when i sends a join to the wave. Let $A_i = \texttt{JT}_i + \texttt{RTT}_i$ for receiver i. Note that A_i is the wave time at N_i when the join sent at wave time $\texttt{JT}_i$ from receiver i establishes forwarding state for the wave at N_i. Let T_i be the wave time when a first packet from the wave arrives at N_i, and let Q_i be the maximum of T_i and A_i. Then, Q_i is the wave time at N_i when N_i forwards a first packet from the wave towards receiver i, and thus Q_i is also the wave time when a first packet from the wave passes through every point along the path from N_i to receiver i and arrives at i. The measured MRTT value at receiver i is $\texttt{MRTT}_i = Q_i - \texttt{JT}_i$.

Cycle C_i between N_i and i is called *slack* if $T_i > A_i$, *i.e.*, if the join from i establishes forwarding state at N_i at wave time A_i before a first packet from the wave arrives at N_i at wave time T_i, and thus the first packet from the wave arrives at i at time T_i. Note that in this case $Q_i = T_i > A_i = \texttt{JT}_i + \texttt{RTT}_i$ and thus $\texttt{MRTT}_i = Q_i - \texttt{JT}_i > \texttt{RTT}_i$. Cycle C_i is called *taut* if $T_i \leq A_i$, *i.e.*, if the join from i establishes forwarding state at N_i at wave time A_i after a first packet from the wave arrives at N_i at time T_i, and thus the first packet from the wave arrives at i at time A_i. Note that in this case $Q_i = A_i = \texttt{JT}_i + \texttt{RTT}_i$ and thus $\texttt{MRTT}_i = Q_i - \texttt{JT}_i = \texttt{RTT}_i$. Since each forward and reverse link in the multicast tree appears in exactly one cycle in each MRTT measurement, this implies that the sum of the MRTT values of all receivers in each measurement is at least the sum of the delays of all links in the multicast tree, and equality is achieved only if all cycles are taut.

It should be noted that even small discrepancies in join times of receivers will tend to make cycles taut. For example, typical RTTs in a well-tuned network should be small, *e.g.*, tens or at most a few hundreds of milliseconds. On the other hand, with the default WEBRC settings, small differences in target reception rates imply significant differences in join times, *e.g.*, a 1.45% difference in target reception rates implies a 500 ms difference in join times. Thus, often cycles will be taut, not slack.

4.5 Buffer filling

When a buffer associated with an interface starts to fill, the latency through the interface starts to grow. If the interface is within a taut cycle during an MRTT measurement and the interface latency grows, then the MRTTs of the receivers associated with the cycle and all of its children slack cycles will grow by the same amount. If the interface is within a slack cycle and the interface latency grows then the MRTTs of all receivers stays the same. However, the RTT of a slack cycle can only change within certain bounds without becoming taut. In particular, if C_i is a slack cycle associated with receiver i and if the RTT of C_i increases to the current value of $\texttt{MRTT}_i$ then C_i becomes taut. As mentioned in the previous subsection, cycles are typically taut.

Changes in the relative join times of receivers can completely change the cycle structure between consecutive MRTT measurements. Generally, receivers that have higher target reception rates relative to other receivers will have earlier join times and thus are more likely to form taut cycles that extend farther towards the sender when an MRTT measurement is made, and thus the measured MRTT for these higher reception rate receivers tends to be higher. This

[6] With current versions of multicast, joins actually only propagate back to the router closest to the sender and not to the sender. To avoid unnecessary complexity in the descriptions, the sender and the closest router to the sender are viewed as being coincident, *i.e.*, the latency between the sender and the closest router is zero.

in turn tends to reverse the roles of the receivers in the next measurement, *i.e.*, lower reception rate receivers for the previous wave will tend to be higher reception rate receivers at the next wave and thus will tend to form a taut cycle at the next measurement. Thus, from one wave to the next the cycle structure may be quite different due to changes in join times of receivers, and increases in interface buffer latencies will tend to be averaged into this process. Overall, whenever a buffer starts filling up the MRTT values for all the downstream receivers eventually increase, and the higher reception rate downstream receivers feel the effect most immediately, causing the maximum reception rate through the bottleneck link to be reduced by the filling buffer. A detailed explanation of the impact of buffer filling on reception rates due to changes in the MRTT is beyond the scope of this paper.

The overall reaction to increases in interface latencies for WEBRC is similar to, but more subtle than, the corresponding reaction for TCP. As a special case, when there is one WEBRC receiver in the session then there is one taut cycle between that receiver and the sender and any increase in latency between the receiver and the sender will increase the MRTT of the receiver. This is exactly analogous to TCP.

4.6 Relative scaling

Given a fixed multicast tree with a fixed set of latencies and a fixed set of receivers, the measured MRTT values for the receivers depends only on the differences between the join times of the receivers, and not on the absolute join times. Thus, if there are n receivers 1, 2, ..., n then the measured MRTT values $\texttt{MRTT}_1$, $\texttt{MRTT}_2$, ..., $\texttt{MRTT}_n$ will be the same with respect to join times $\texttt{JT}_1$, $\texttt{JT}_2$, ..., $\texttt{JT}_n$ and with respect to the join times $\texttt{JT}_1 + x$, $\texttt{JT}_2 + x$, ..., $\texttt{JT}_n + x$ for any value of x. Note that the target rate $\texttt{TRATE}_i$ of receiver i is proportional to $1/\texttt{P}^{\texttt{JT}_i}$. This implies that the measured MRTT values $\texttt{MRTT}_1$, $\texttt{MRTT}_2$, ..., $\texttt{MRTT}_n$ will be the same with respect to the set of target rates $\texttt{TRATE}_1$, $\texttt{TRATE}_2$, ..., $\texttt{TRATE}_n$ and the set of target rates $c \cdot \texttt{TRATE}_1$, $c \cdot \texttt{TRATE}_2$, ..., $c \cdot \texttt{TRATE}_n$ for any positive value c. The implication is that the measured MRTT values only depend on the relative target rates of the receivers and not on their actual target rates.

Note also from Eq. (2) that for any pair of receivers i and j the ratio of their target rates $\texttt{TRATE}_i/\texttt{TRATE}_j$ is approximated by $(\sqrt{p_j} \cdot \texttt{MRTT}_j)/(\sqrt{p_i} \cdot \texttt{MRTT}_i)$, where p_i and p_j are the loss rates respectively of i and j. Furthermore, the MRTT values depend on the link latencies of the multicast tree. This implies that the interactions between the target rates of receivers depends only on the ratio of the link latencies of the multicast tree and on the ratio of the loss rates of the receivers, and not on the absolute values.

4.7 Receiver coordination

The importance of receiver reception rate coordination and equalization is that it maximizes the utility of packets that pass through each link in the network. Ideally, each packet that passes through a link is used by all receivers that are downstream of that link.

Fig. 3 shows two receivers A and B both connected to the same router which in turn is connected to the sender. The RTT of the cycle between the router and the sender is X, the RTT of the cycle between A and the router is Y and the RTT of the cycle between B and the router is Z. As described in Subsection 4.6, $\texttt{MRTT}_\text{A}$ and $\texttt{MRTT}_\text{B}$ depend only

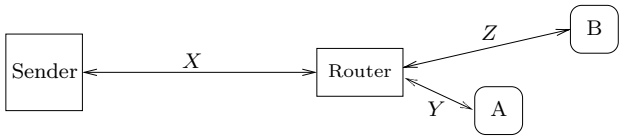


Figure 3: Two nearby receivers A and B connected to a faraway sender through the same router.

on the difference between $\texttt{JT}_\text{A}$ and $\texttt{JT}_\text{B}$, and not on their absolute values. Consider the special case when packets are lost independently and randomly between the sender and the router and there are no losses on the other links, and thus the loss rates p_A and p_B are equal. If $|Y - Z| < X$ then at equilibrium $\texttt{MRTT}_\text{A} = \texttt{MRTT}_\text{B} \geq (X + Y + Z)/2$. Furthermore, when there are only two receivers below the bottleneck link, typically both cycles will be taut in the measurements, in which case $\texttt{MRTT}_\text{A} = \texttt{MRTT}_\text{B} = (X + Y + Z)/2$.

Note that if only receiver A is in the session then $\texttt{MRTT}_\text{A} = X + Y$ whereas if only receiver B is in the session then $\texttt{MRTT}_\text{B} = X + Z$. Not only do the two receivers equalize their reception rates, but also the MRTT values of the receivers when both are participating in the session is smaller than their MRTT values when each receiver is individually participating in the session. Thus, the reception rates of both receivers is higher when both participate in the session than when they individually participate in the session, and their reception rates are equal.

In general, when there are multiple proximate receivers downstream of a bottleneck link their reception rates tend to coordinate and equalize due to the measured MRTT values, and as more and more receivers join the session the MRTT values of the receivers already in the session tend to decrease. Thus, the reception rates of the receivers tend to coordinate and increase as more and more receivers join the session. A detailed analysis of receiver coordination is beyond the scope of this paper.

4.8 Fair sharing

As described previously, when there is a bottleneck link that is shared by several sessions, the share of the bandwidth for each session that is fair is proportional to its utility, where utility is the ratio of benefit to cost. Recall that the sum of the MRTT measurements of all receivers for a wave is at least the sum of the delays of all links in the multicast tree, and equality is achieved when all cycles associated with receivers are taut. Recall also that one definition of the cost of the session is the total amount of time spent by the network to deliver packets, which in the multicast case is closely approximated by the sum of the delays of all links in the multicast tree. This implies the sum of the measured MRTTs is at least the cost of the session, and when there are slack cycles the sum of the measured MRTTs can be greater than the cost of the session. As observed in the previous subsection, the equilibrium average measured MRTT values of proximate receivers will be approximately the same. Since each receiver uses the inverse of its average measured MRTT value to determine its target reception rate, this implies in equilibrium the target reception rate of each receiver will be proportional to the number of receivers divided by the sum of their measured MRTT values, *i.e.*, at most proportional to the utility of the session. There will be more and more

slackness in the cycles of the receivers below a bottleneck in each MRTT measurement as the number of receivers below the bottleneck link grows, implying that the sum of their MRTT measurements will be larger than the cost of the session, and that consequently their proportional equilibrium target reception rates will be below the utility of the session, *i.e.*, they will act more conservatively.

The description of the more general case, when the receivers below the bottleneck link are further apart from one another and therefore their reception rates do not equalize, is also quite interesting. It can be shown that the receivers naturally tend to partition into sets of receivers that all have the same equilibrium reception rate, and the reception rate of each set of receivers is at most proportional to a natural extension of the notion of utility. A detailed description and analysis of fair sharing is beyond the scope of this paper.

5. SIMULATION RESULTS

We have simulated the WEBRC protocol extensively with *ns* [15]. These simulations are sampled and summarized here; more results are reported in [11]. Our implementation of WEBRC and some simulation scripts are available online so that the reader can verify our results and experiment further. The parameters given in Table 1 were used.

5.1 Sessions without dynamic competition

5.1.1 Convergence to desired rate

In the first set of simulations a single WEBRC sender is connected to a single WEBRC receiver with two intervening routers. The link between the two routers has random packet losses and the speeds of all links are high enough that router buffer sizes are irrelevant. These experiments are designed to verify that a WEBRC receiver obtains estimates `LOSSP` and `ARTT` that accurately reflect the packet loss probability and multicast RTT and that the reception rate can be predicted with Eq. (1). In these experiments the sender is started at time zero and the receiver is started at a uniformly random time in $[0, \texttt{TSD}]$ to eliminate any phase synchronization between sender and receiver. Each simulation runs for 500 seconds.

Fig. 4 shows the first 250 seconds of two such experiments in which the packet loss probability is 1% and the RTT is 0.2 sec. The three panels give `ARTT`, `LOSSP`, and the throughput.[7] In both experiments the receiver enters the session at about time 9. The first MRTT measurement is biased upward by the interpacket interval on the base channel so it is an overestimate and has large variance. Subsequent measurements lead to convergence of `ARTT` to the appropriate value of 0.2 sec. `LOSSP` is initialized when the first packet loss occurs to a value that corresponds through Eq. (6) to the current reception rate. In one of the simulations the first loss occurs later, when the reception rate is higher, so the value of `LOSSP` is lower. In both trials `LOSSP` quickly settles to close to the appropriate value of 0.01. The early behavior of the throughput depends on the timing of the first packet loss. This transient behavior seems to have no effect on steady-state throughput. At steady state the throughput is approximately periodic with period `TSD`.

[7]Throughput is aggregated at 1 second intervals while all changes of `ARTT` and `LOSSP` are shown. In other experimental results packet loss is also aggregated at 1 second intervals and all changes of internal variables are shown.

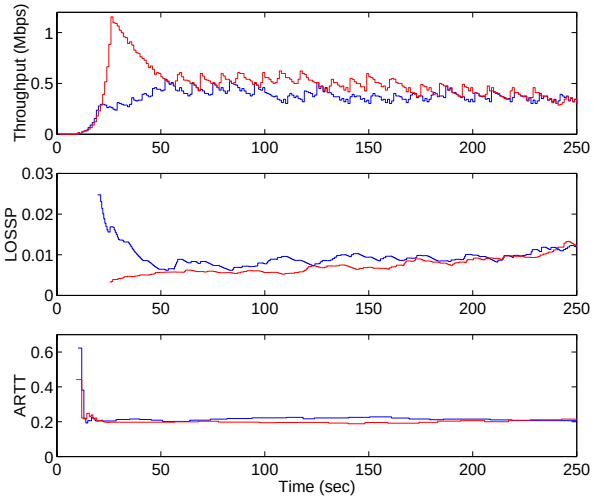


Figure 4: Two trials of a single-receiver WEBRC session on a link with 1% random packet loss and 0.2 sec RTT. `LOSSP` and `ARTT` converge to the correct values, with a small amount of steady-state variation. The throughput varies around the expected value, approximately periodically with period `TSD`.

As a crude approximation, the average steady-state throughput of a WEBRC session should be obtained by evaluating Eq. (1) with p replaced by the packet loss probability and t_{RTT} replaced by the MRTT. However, two adjustments are warranted. First, the rate increases in WEBRC are conservative, or polite, so that the *peak* rate matches the TFRC equation; it is inevitable for the rate to decrease between joins, making the average throughput lower than predicted by Eq. (1). Second, like TFRC, WEBRC uses the loss event rate rather than the packet loss fraction for p in Eq. (1). The loss event rate is lower than the packet loss fraction, especially when the packet loss fraction is large.

Suppose in steady-state the receiver joins each `TSD` seconds to reach a constant target rate R with each join. By integrating the aggregate rate while considering all placements of the join within the time slot, it can be shown that the average throughput $\bar{R}$ satisfies

$$\frac{1-\texttt{P}}{\ln(1/\texttt{P})}(R-\texttt{P}) \le \bar{R} \le \frac{1-\texttt{P}}{\ln(1/\texttt{P})}R. \tag{7}$$

Thus, the difference between peak and average reception rates reduces the throughput of WEBRC approximately by the factor $(1-\texttt{P})/\ln(1/\texttt{P})$. For the default values of $\texttt{P} = 0.75$ and $\texttt{TSD} = 10$ this evaluates to 0.869.

The difference between the packet loss probability p and the loss event rate is due to ignoring losses for a duration of `ARTT` from the beginning of a loss event. This has only a mild affect on throughput, except when p is large. By relating `ARTT` to the reception rate through Eq. 2, it can be shown that the expected loss event rate, as estimated by the receiver, is given approximately by

$$\texttt{LOSSP} = p \Big/ \left(1 + \sqrt{\frac{3p}{2}}\right). \tag{8}$$

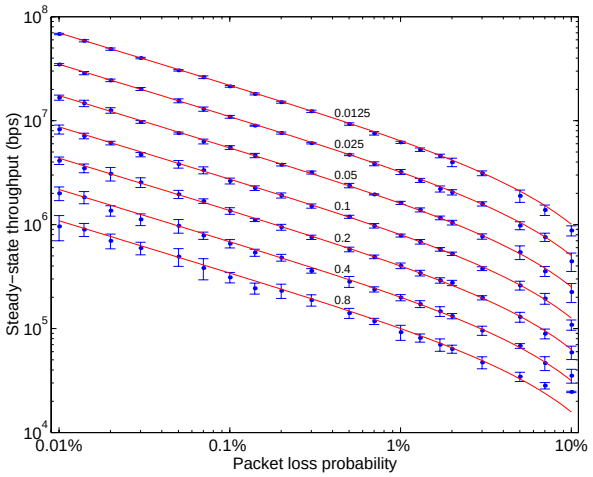


Figure 5: Average steady-state throughputs from simulations of single-receiver WEBRC sessions. The simulation results (·'s with error bars indicating ± one standard deviation) are compared to the desired throughput from Eq. (1), adjusted according to the upper bound of Eq. (7). Curves are labeled with the MRTT in seconds.

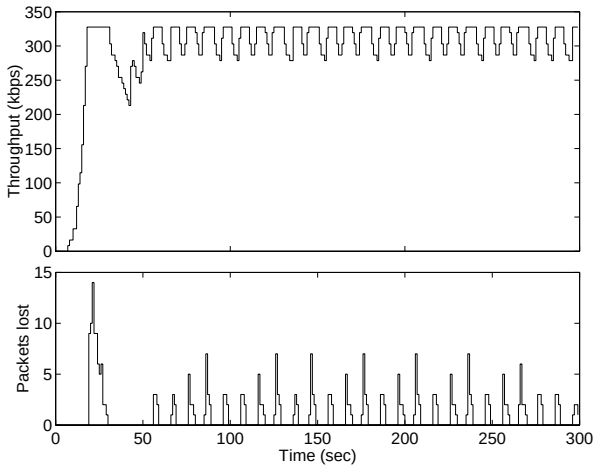


Figure 6: Throughput and packet loss of a WEBRC session where the bottleneck bandwidth is 320 kbps, RTT is 0.1 sec, and buffer length is 4 packets.

LOSSP is thus expected to be slightly smaller than p, and the ratio between the two approaches 1 as p approaches zero.

The results shown in Fig. 4 are consistent with our analysis. Evaluating Eq. (8) with $p = 0.01$ gives 0.0089, and using this along with $t_{\mathrm{RTT}} = 0.2$ in Eq. (1) gives throughput $R = 60.1$ packets per second or, with 1024-byte packets, 492 kbps. Using the upper bound from Eq. (7) then reduces the predicted throughput to 428 kbps. This is close to the average steady-state throughput of 403 kbps observed in eight such trials used in producing Fig. 5.[8]

To assess whether WEBRC achieves the desired throughput more generally, eight trials were conducted for each RTT in {0.0125 sec, 0.025 sec, ..., 0.8 sec} and several loss probabilities between 0.01% and 10%. The average steady-state throughputs are compared to the throughputs from Eq. (1), adjusted according to the upper bound of Eq. (7), in Fig. 5. The fit is very good except for the two lowest-rate points. WEBRC throughput does not match the equation well for rates below about three times the base channel rate.

5.1.2 Bandwidth utilization

The second set of experiments explores the ability of a WEBRC session to utilize a large fraction of available bandwidth. The network topology of the previous subsection is used, but this time there are no random packet losses; rather, the link between the two routers is the bottleneck link and all packet losses are due to buffer overflows in the router closest to the sender.

Fig. 6 shows the throughput and packet loss from a single experiment in which the bottleneck link has 320 kbps bandwidth, the routers can buffer 4 packets, and the RTT, without accounting for queuing delays, is 0.1 sec. Observe that the throughput increases exponentially at the beginning of the session as described in Section 3.2. After the first packet loss, the target rate is governed by the estimates of packet loss probability and MRTT. The burst of losses at the end of slow start leads to a large value of LOSSP and inhibits the receiver somewhat, but a steady state is reached quickly. At steady state, wave channels are joined approximately every TSD seconds; the timing of the joins within the time slot makes the packet loss rate, MRTT, and throughput approximately satisfy Eq. (1). The steady-state throughput is 95% of the bottleneck bandwidth. We refer to this percentage as the *bandwidth utilization* of the bottleneck link.

As it does for TCP, the bandwidth utilization of WEBRC depends on the amount of buffering in the network. The utilization generally increases with greater buffering because the buffers can absorb packets from a portion of a wave at a rate greater than the bottleneck bandwidth. When there is enough buffering, the joining rule described in Section 3.3 can allow a WEBRC session to get nearly 100% bandwidth utilization. Fig. 7 shows the throughput of a WEBRC session when the bottleneck bandwidth is 3.2 Mbps, the routers can buffer 160 packets, and the RTT is 0.1 sec. In this case the steady-state bandwidth utilization is 99.5% and there are no packet losses—not even at session start-up. The receiver leaves the session start-up mode because it observes that RR_P is lagging behind ARR_P.

5.1.3 Responsiveness to network changes

The results of the two previous sections indicate that a WEBRC session with a single receiver operating in isolation in a static network behaves as desired. We now consider a slightly more complicated situation in which the amount of available bandwidth changes during a session. The WEBRC session shares a bottleneck link with a UDP flow that begins 150 seconds after the WEBRC sender and receiver have started. When the UDP flow commences, it uses half of the 1 Mbps bottleneck; after 150 seconds it drops its injection rate to 250 kbps. The WEBRC session has 0.05 sec MRTT and the routers can buffer 6 packets.

[8]Here and in subsequent experiments we define the "steady-state throughput" as the average throughput over the last half of the simulation.

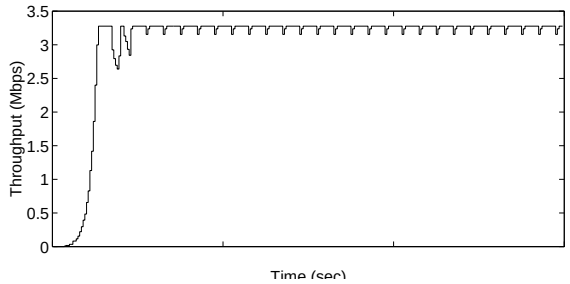


Figure 7: Throughput of a WEBRC session where the bandwidth is 3.2 Mbps, RTT is 0.1 sec, and buffer length is 160 packets. There is no packet loss.

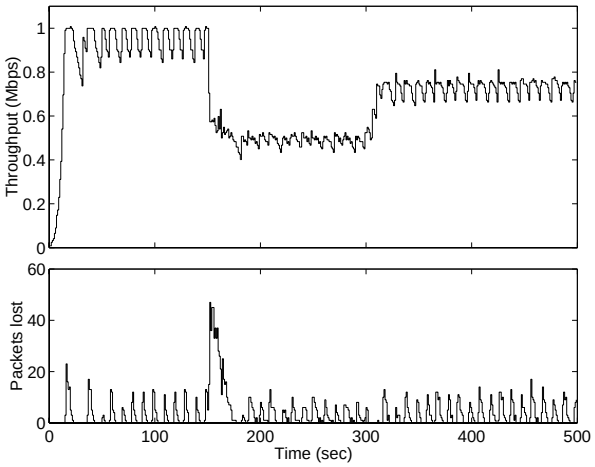
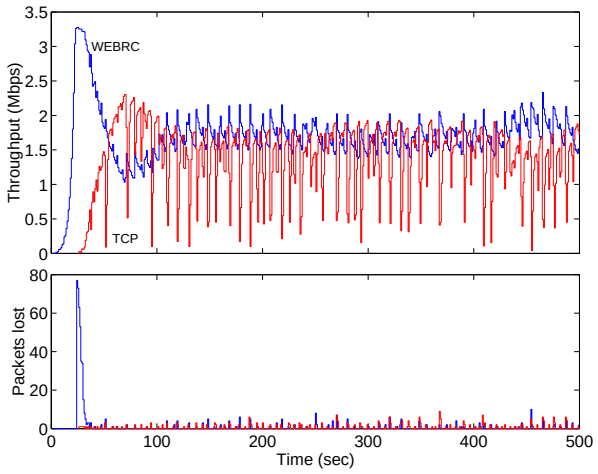
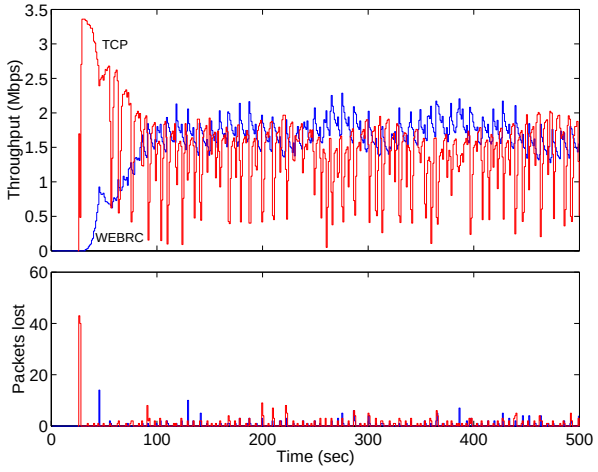


Figure 8: A single-receiver WEBRC session responding to changes in available bandwidth. Initially, 1 Mbps is available. After 150 seconds a 500 kbps constant bit rate flow starts to compete, and after another 150 seconds it reduces its rate to 250 kbps.



(a) WEBRC session starts first.



(b) TCP session starts first.

Figure 9: A WEBRC session and a TCP session share a 3.2 Mbps link. Each has 0.1 sec RTT.

Fig. 8 shows the throughput and packet loss of the WEBRC session. Notice that the initial behavior is no different than in Fig. 6 as the receiver quickly increases its reception rate until the bottleneck bandwidth is reached. The throughput stabilizes until the competing traffic at time 150 causes a burst of packet losses. This increases `LOSSP` and hence decreases the target rate; the receiver allows several time slots to pass without a join, so the packet loss rate decreases. With very little undershoot in the reception rate, `LOSSP` drops enough to allow the receiver to start joining wave channels and again reach a steady state. When the competing traffic drops its rate, the WEBRC receiver quickly recognizes a drop in the packet loss rate and hence increases its reception rate.

5.2 Sharing with TCP

Having similar behavior as TCP in isolation, as demonstrated in Section 5.1, is only the first step toward fairness to TCP flows. The dynamic behavior of WEBRC should allow it to share common links with TCP flows fairly, regardless of which connection started first or got ahead for any reason.

It is important to note that "sharing fairly" does not always mean getting equal throughput. By this strict standard, TCP is not fair to itself because flows competing over a common bottleneck have throughputs inversely proportional to their RTTs [7].

Because of space limitations, only experiments in which one WEBRC session competes with one TCP session, with the MRTT of the former equal to the RTT of the latter, are presented. Under these conditions, we would like for the WEBRC and TCP sessions to have roughly equal average throughput. (More precisely, our expectations could be adjusted according to Eq. (7).)

Fig. 9 shows two simulations with one WEBRC session and one TCP session. In both cases the bottleneck bandwidth is 3.2 Mbps, the RTT is 0.1 sec, and the routers can buffer 40 packets. Whichever session starts first quickly increases its rate to the full bandwidth. Observe that the TCP session is able to start more quickly because it doubles it rate each RTT (0.1 sec), while the WEBRC session increases its rate approximately by a factor $1/\mathtt{P} = 4/3$ every two epochs (1.0 sec). When the second, competing session

starts, it increases its rate to its share of the bottleneck; this increase is much slower than that of the first session. In steady state, the two sessions have similar average throughput, though the WEBRC session has much less variation in its throughput. The numbers of packet losses are also similar. In these particular experiments, the WEBRC session gets 55% and 56% of the total throughput. The WEBRC and TCP sessions together have bandwidth utilizations of 95% and 94%.

A large set of simulations with a slightly older version of the protocol demonstrated very good fairness on average, but showed a great deal of variation between experiments. When the experiment is repeated with two competing TCP sessions there is significant—though less—variation in the throughput ratios. A slight trend of TCP fairing better as the buffer sizes increase was observed and warrants study.

5.3 Receiver interactions

This final section of simulations demonstrates the coordination and sharing properties of WEBRC that stem from properties of MRTT discussed in Sections 4.7 and 4.8.

5.3.1 *Coordination of two receivers*

Consider two receivers in the same WEBRC session as illustrated in Fig. 3. The link between the sender and the router has 0.2 sec RTT and 1% random packet loss. The links to Receivers A and B have 0.05 and 0.1 sec RTT.

Fig. 10 shows the throughputs and average MRTTs in a simulation in which Receiver A joins the session at time 8 and Receiver B joins the session at time 200. Without Receiver B in the session, the MRTT of Receiver A is 0.25 sec, and the receiver's estimate indeed converges to this value. After Receiver B enters the session, the MRTTs of both receivers converge to 0.175 sec. Since the packets lost by the two receivers are the same, the convergence of MRTTs implies also a convergence of reception rates. This is desirable because the bottleneck link is fully utilized by both receivers, with most packets going to both receivers. In fact, in steady-state 96.8% of the packets at the shared router are forwarded to both receivers. The same type of coordination occurs if Receiver B is the first to enter the session.

With the same network topology, if Receivers A and B are in different sessions then their MRTTs are 0.25 sec and 0.3 sec, respectively. Their throughputs are thus lower by 30% and 42%.

5.3.2 *Effect of receiver density*

To verify that a multi-receiver WEBRC session shares fairly as described in Section 4.8, consider a WEBRC session with n receivers that are close to each other and far from the sender. We model this in *ns* as a sender connected to a first router, this first router connected to a second router using a high bandwidth link with 0.198 sec RTT and 2% random packet loss, and n receivers connected to the second router using lossless links with 0.002 sec RTT.

As described in Section 4.4, for any one wave, the sum of the MRTT measurements by the receivers will be at least equal to the sum of the RTTs of the links, which is $0.198 + 0.002n$ sec. The average over the receivers of the MRTT measurements is thus at least $0.198/n + 0.002$ sec. If all cycles are taut at each measurement, the time-average computed by any one receiver will reach an approximate steady-state that is also at least $0.198/n + 0.002$ sec. Since the target reception rate is inversely proportional to the average MRTT, the analysis predicts that having n receivers will multiply the throughput by at most $n/(0.99 + 0.01n)$. This is less than proportional to n and hence less usage of the bottleneck link than one TCP connection for each receiver. Furthermore, slack cycles make the MRTTs larger and thus make the WEBRC session less aggressive.

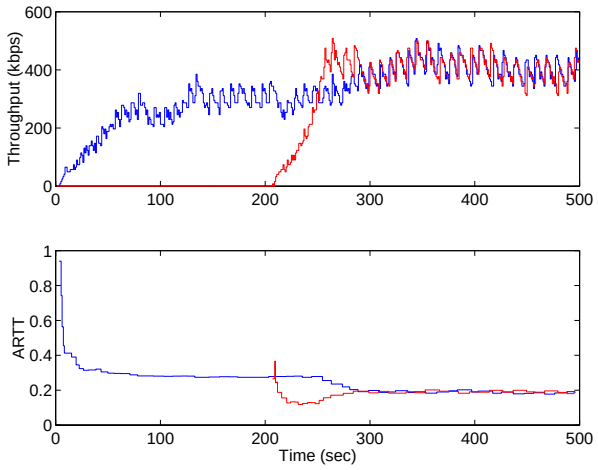


Figure 10: The coordination of two receivers in the scenario depicted in Fig. 3. With reference to Fig. 3, $X = 0.2$ sec, $Y = 0.05$ sec, and $Z = 0.1$ sec. The shared link has 1% random packet loss.

Fig. 11 shows experimental confirmation of these conclusions. Each point in the bottom panel gives the steady-state throughput of one of the n receivers as a function of n. (A small dither is added to the horizontal components to make the points more discernable.) The MRTT analysis with taut cycles, combined with Eqs. (6)–(8), gives the sublinear marked curve; the actual bandwidth usage of the WEBRC session grows even more slowly.

The bottleneck link is used very efficiently in that most packets that traverse the bottleneck link are sent to all the receivers. The bottom panel of Fig. 11 shows the fraction of packets that cross the bottleneck link that are forwarded to each receiver, averaged over 20 waves in steady-state. The average over the receivers is about 95%. The MRTT-based coordination is very good, without any explicit communication between receivers.

6. LIMITATIONS, FURTHER WORK AND CONCLUSIONS

A concern with respect to the deployment of any multicast congestion control protocol is the potential impact on the network by misbehaving senders, receivers, and interlopers. These concerns are discussed in some detail in [10]. In summary, WEBRC does not have any essential concerns with respect to its impact on the network beyond the similar concerns for multicast and TCP flows in general and those detailed below.

Other important factors for the deployment of WEBRC are its performance with respect to the networking infrastructure in which it will be deployed. For example, it is important to understand the propagation delay times of IGMP

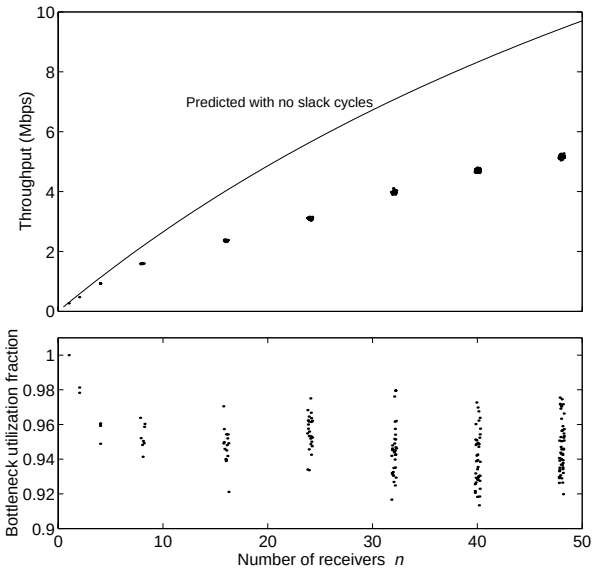


Figure 11: The effect of the number of receivers sharing a common bottleneck on session throughput and efficiency of use of the bottleneck link. A shared link with 2% random packet loss and 0.198 sec RTT connects the sender to a router very close (0.002 sec RTT) to all of the receivers.

and PIM SM join messages as they travel up the multicast tree from a receiver towards the sender. These delays affect the MRTT measurements, which in turn affect the target reception rates of the receivers. In general the reception rate of WEBRC receivers will be lower when these propagation times are higher. Another very important factor is the capabilities of network elements to handle the IGMP and PIM SM control traffic loads that come from WEBRC receivers. For example, if there are 1,000 WEBRC receivers directly connected to a layer three switch, then with the default setting of 10 seconds for `TSD`, the switch will have to be able to handle 100 IGMP join and leave messages per second on average. The combination of these factors is also important to understand, *e.g.* the effect on IGMP join latency of handling 100 IGMP messages per second while forwarding data packets at a high rate. It would be useful to gather these types of performance statistics from a variety of network equipment providers in order to verify that WEBRC is viable.

Finally, WEBRC is affected by the multicast service model on which it is deployed. Today, there are two primary models to consider: ASM and SSM. ASM is more of a concern because of the impact of MSDP and the Rendezvous Point (RP) on the WEBRC data traffic, but this impact will be minimized if the RP is near the WEBRC sender. Of more concern with ASM is that generally the number of available multicast groups may be limited, and each WEBRC session uses many multicast groups. This is not a large concern if the ASM session is administratively scoped, as generally in this case there is an abundant supply of multicast groups. These issues are generally not a concern with SSM.

In conclusion, if the outcomes of further performance measurements are positive, WEBRC has the promise of providing a multiple rate multicast congestion control protocol that is massively scalable, fair to competing flows and network friendly, and that provides high throughput to all receivers in a heterogeneous network environment.

7. ACKNOWLEDGMENTS

Several refinements to WEBRC have been inspired by feedback from Robert Chapman, Armin Haken, and Diane Hernek. We also thank John Byers and the anonymous reviewers for their extensive and helpful comments.

8. REFERENCES

[1] J. Byers *et al.* FLID-DL: Congestion control for layered multicast. In *Proc. 2nd Int. Workshop Netw. Group Comm.*, pages 71–81, Stanford, CA, Nov. 2000.

[2] J. W. Byers *et al.* A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM*, pages 56–67, Vancouver, Sept. 1998.

[3] P. Dagum *et al.* An optimal algorithm for Monte Carlo estimation. *SIAM J. Comput.*, 29(5):1484–1496, Apr. 2000.

[4] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Networking*, 7(4):458–472, Aug. 1999.

[5] S. Floyd *et al.* Equation-based congestion control for unicast applications: the extended version. Technical Report TR-00-003, Int. Comp. Sci. Instit., Berkeley, CA, Mar. 2000.

[6] M. Handley *et al.* TCP friendly rate control (TFRC): Protocol specification. IETF Transport Area Working Group Internet-Draft, Apr. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-tfrc-04.txt`. Work in progress. Expires Oct. 2002.

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

[8] M. Luby *et al.* Asynchronous layered coding protocol instantiation. IETF Reliable Multicast Transport Working Group Internet-Draft, Apr. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-pi-alc-08.txt`. Work in progress. Expires Oct. 2002.

[9] M. Luby *et al.* Layered coding transport building block. IETF Reliable Multicast Transport Working Group Internet-Draft, Feb. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-lct-04.txt`. Work in progress. Expires Aug. 2002.

[10] M. Luby and V. K. Goyal. Wave and equation based rate control building block. IETF Reliable Multicast Transport Working Group Internet-Draft, June 2002. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-webrc-02.txt`. Work in progress. Expires Dec. 2002.

[11] M. Luby and V. K. Goyal. Wave and equation based rate control using multicast round trip time: Extended report. Technical Report DF2002-07-001, Digital Fountain, July 2002. Available on-line at `http://www.digitalfountain.com/technology/`.

[12] M. Luby *et al.* Forward error correction building block. IETF Reliable Multicast Transport Working Group Internet-Draft, Feb. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-fec-06.txt`. Work in progress. Expires Aug. 2002.

[13] M. Luby *et al.* The use of forward error correction in reliable multicast. IETF Reliable Multicast Transport Working Group Internet-Draft, Feb. 2002. `http://www.ietf.org/internet-drafts/draft-ietf-rmt-info-fec-02.txt`. Work in progress. Expires Aug. 2002.

[14] M. Mathis *et al.* The macroscopic behavior of the TCP congestion avoidance algorithm. *Comp. Comm. Rev.*, 27(3), July 1997.

[15] The network simulator – *ns-2*. Available at `http://www.isi.edu/nsnam/ns`.

[16] J. Padhye *et al.* Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM*, pages 303–314, Vancouver, Sept. 1998.

[17] A. Perrig *et al.* Efficient and secure source authentication for multicast. In *Proc. Netw. & Distrib. Syst. Sec. Symp.*, pages 35–46, San Diego, Feb. 2001.

[18] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proc. IEEE INFOCOM*, volume 3, pages 996–1003, San Francisco, CA, Mar.–Apr. 1998.

[19] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network*, 15(3):28–37, May–June 2001.

APPENDIX

Several variables in a WEBRC receiver are moving averages of measurements based on discrete events. For `ARR_P` and `TRR_P` it is sufficient to filter the measurements (in these cases packet counts) at regular intervals using an exponentially weighted moving average (EWMA) with a fixed averaging fraction. However, estimation of the average loss probability `LOSSP` and of the average MRTT `ARTT` both require a more general and sophisticated filtering approach.

The design of TFRC [5, 6] reflects that, because the average packet loss probability can vary by orders of magnitude, any estimate of the average loss probability based on either a fixed number of packets or on a fixed period of time using a fixed averaging fraction will be poor. In TFRC the average is estimated from the numbers of packets between beginnings of loss events, and the number of loss events used is fixed. This particular example of estimating a moving average based on a measurement interval that depends on the measurements hints at a general approach to producing meaningful estimates of moving averages.

The paper [3] introduces a protocol that uses a minimal number of measurements of a random variable with an *a priori* unknown average μ and unknown variance σ^2 to produce a provably good estimate of μ. The protocol uses an on-line stopping rule that depends on ongoing estimates of μ and σ^2 as the measurements are made to decide when there are enough measurements to produce a good estimate of μ. The number of measurements the protocol uses is provably proportional to σ^2/μ^2. This protocol is the inspiration for the estimators introduced below.

The following is a general estimator for a series of measurements $m_1, m_2, \ldots$ of a variable $X \in [0, 1]$. Let $\alpha \in (0, 1)$ be a smoothing constant. Set $a_0 = 1$ and $v_0 = 1$. Do the following to produce the i^{th} estimate a_i. Let

$$c_{i-1} = \alpha \cdot a_{i-1}^2 / v_{i-1}.$$

Then,

$$a_i = (1 - c_{i-1}) \cdot a_{i-1} + c_{i-1} \cdot m_i$$

and

$$v_i = (1 - c_{i-1}) \cdot v_{i-1} + c_{i-1} \cdot m_i^2.$$

This estimator is an EWMA with an averaging fraction that self-adjusts depending on the previous estimates of the averages of X and X^2.

The WEBRC receiver uses this estimator with a small, fixed smoothing constant α to estimate its average MRTT value. The variability in the MRTT measurements of WEBRC receivers can depend on the unknown number of receivers below a bottleneck link, and this estimator appropriately takes into account this variability without *a priori* information on the number of receivers.

For the special case when X is a $\{0, 1\}$-valued variable, the above estimator can be translated into the following form to estimate the inverse of the average of X. Let $\delta \in (0, 1)$ be a smoothing constant. Set $F_0 = 0$ and do the following to produce the i^{th} estimate F_i. If $m_i = 0$, then $F_i = F_{i-1} + \delta$; else $m_i = 1$ and $F_i = (1-\delta) \cdot (F_{i-1} + \delta)$. A simple analysis shows that if X is a random variable with $\Pr[X = 1] = p$ then the expected value of the estimator is eventually $(1 - \delta \cdot p)/p$, *i.e.*, essentially the reciprocal of p.

This estimator can be directly applied to estimate the inverse of the average loss probability, or equivalently the interval between loss events, where each m_i corresponds to a packet and $m_i = 0$ if the packet is received and $m_i = 1$ if the packet is lost. Note:

- This estimate is essentially the same as the EWMA estimate of the interval between loss events suggested in TFRC ([5] states that EWMA works reasonably well, but also offers an estimator with a different weighting strategy that is found to be superior).
- In the simple TCP equation, the reception rate is proportional to the inverse of the square root of the loss probability. A loose translation of the above estimate into a direct estimate of the square root of the inverse of the loss probability yields the following. Let $\delta' = \delta/2$. Set $G_0 = 0$. Do the following to produce the i^{th} estimate G_i. If $m_i = 0$, then $G_i = G_{i-1} + \delta'/G_{i-1}$; else $m_i = 1$ and $G_i = (1 - \delta') \cdot (G_{i-1} + \delta'/G_{i-1})$. Interesting, this is very similar to the TCP sliding window update rule.

The WEBRC receiver estimates the inverse of the average loss probability by applying two filters consecutively to the packet reception measurements. The reason for the first filter is that the loss events in WEBRC are bursty; they typically occur just after a new wave has been joined. To smooth out this burstiness, a simple EWMA filter with a fixed averaging fraction is applied to the packet reception measurements at the end of each epoch to smooth out the bursty loss events over a few time slot durations. Then, the above estimator F_i is applied to this time-smoothed average to produce the estimate of the inverse of the average loss probability, similar to TFRC.

The above description provides the essence of the loss event averaging and the MRTT averaging used in WEBRC. There are many details and nuances that are omitted from this description in order to provide a concise introduction to some of the essential ideas.