Digital Fountain Technical Report DF2002-07-001

Wave and Equation Based Rate Control Using Multicast Round Trip Time: Extended Report

Michael Luby and Vivek K Goyal
Digital Fountain, 39141 Civic Center Drive, Suite 300, Fremont, CA 94538
{luby,vivek}@digitalfountain.com

September 4, 2002

Abstract

Wave and Equation Based Rate Control (WEBRC) is a new equation-based, multiple rate congestion control protocol that is naturally suited to multicast but also applicable to unicast. No previous multiple rate congestion control algorithm is equation based. A main impediment to extending equation-based rate control to multiple rate multicast was until now the lack of a suitable analogue to the round trip time (RTT) in unicast. This paper introduces an analogue of unicast RTT, called multicast round trip time (MRTT), that can be measured by receivers without placing any added message processing burden on the server or intermediate network elements.

WEBRC is TCP-friendly, equation-based rate control that uses MRTT in the place of RTT; thus, MRTT and RTT are forced to have analogous effects on throughput. Like RTT, MRTT reflects buffer occupancy. 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.

Like FLID-DL, WEBRC is insensitive to large IGMP leave message processing. Key advantages of WEBRC over RLC and FLID-DL 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, at times nonexistent; and fairness to TCP is very good.

This report subsumes the paper on WEBRC that appears in Proc. ACM SIGCOMM 2002 [15]. The protocol is evolving in response to continuing simulation and to experience using Digital Fountain's WEBRC implementations in both laboratory and wide-area network environments. This document represents the state of the art as of this date and thus includes some adjustments to the protocol since the -02 revision of the WEBRC protocol specification draft [13]. An -03 revision will be submitted and will inevitably include some refinements not described in this document.

Index Terms—**bandwidth utilization, multicast, multiple rate congestion control, network protocols, network simulation, round trip time, multicast round trip time, waves, transmission control protocol, TCP-friendly rate control, unicast.**

Contents

I. Introduction

The paper [15] 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 [6], [8]. At any point in time, a receiver is receiving packets from the base channel and some number of consecutive 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) [26] 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.

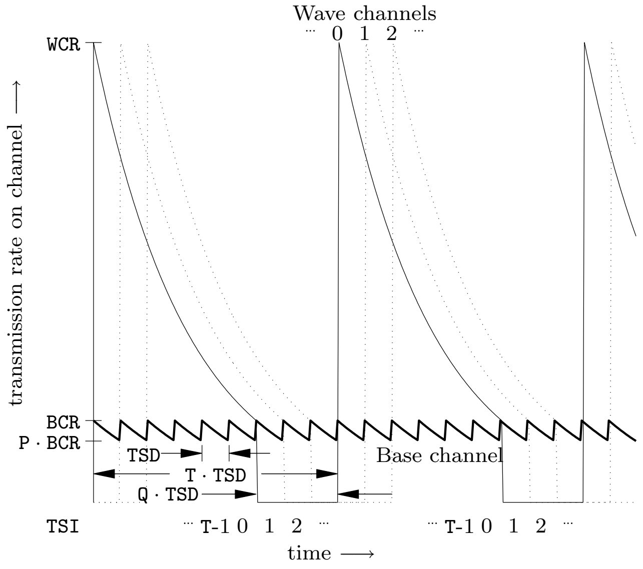


Fig. 1. Rates on WEBRC channels under a fluid transmission model. The base channel rate has small variation over a period of `TSD` seconds. Each of `T` wave channels has much more variation over a period of `T · TSD` seconds. Each wave channel is quiescent for `Q · TSD` seconds in each period. (Three of the `T` wave channels are shown and the repeating wave shape is slightly simplified here. Actual wave shapes are shown in Fig. 14.)

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. WEBRC is more extensively compared to other multicast congestion control protocols in Section V.

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 [13].

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], [26]. Thus, WEBRC is a suitable congestion control protocol to use with the ALC approach to reliable content delivery [10]. ALC combines the LCT building block [11] with the FEC building block [16]. The LCT building block provides higher-level session support, whereas the FEC building block provides coding that enables reliable content delivery as described in [17]. WEBRC can also be used for unicast and for other types of applications, such as streaming.

II. 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 [5]. 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, and some degree of fairness is achieved with the techniques described in [25], [23], [22], [6], [27].[1]

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 [18], [21]. 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 [21, 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

$$R = \frac{\sqrt{3/2}}{t_{\mathrm{RTT}}\sqrt{p}} \qquad (2)$$

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.* [6], [8]. 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. (The details of the estimation of p are reviewed in Section VI-D.3 since very similar techniques are used in WEBRC.)

III. 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 `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.[2]

[1]This paper uses only a non-technical definition of fairness whereby two protocols are fair to each other if they have similar long-term throughput when sharing network links under the same network conditions. Denda [4] provides a detailed exposition of fairness issues in networks. See also [27].

[2]The typewriter font is used for all parameter and variables names that match the draft protocol specification document [13] and the

The output of the server, totaled over all of the channels, is at the constant rate of SR_b bps.[3] 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 some initial rate. 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 $\mathtt{P} \cdot \mathtt{BCR\_P}$ 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 Section VI-C.2.

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

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

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 I. 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.

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	$\mathtt{N} + \mathtt{Q}$
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 I
Summary of terms

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 become quiescent.[4] The receiver knows when channels become quiescent by monitoring the TSIs 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

associated ns simulator code. In addition, it is used for some multi-character variable names that would look awkward in the math font.

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

[4]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.

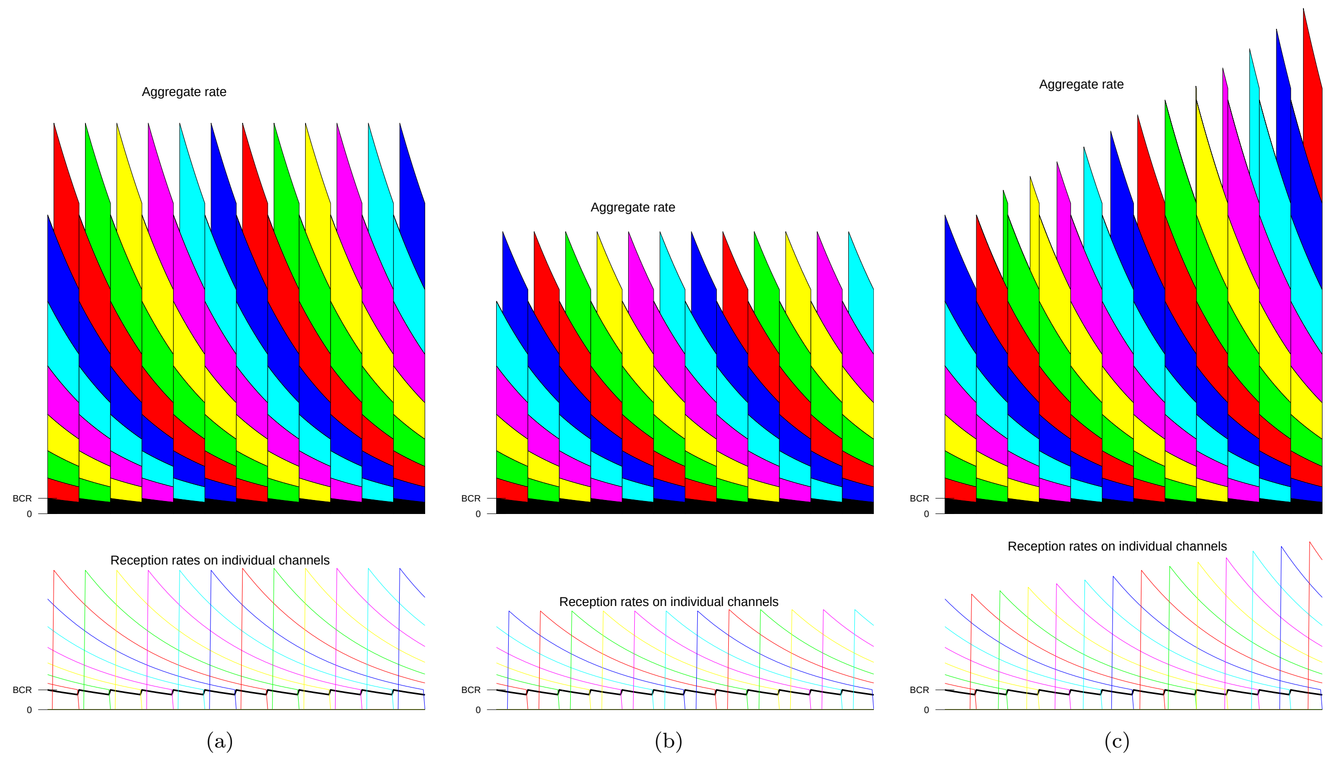


Fig. 2. How a WEBRC receiver adjusts its reception rate as a function of time. The receiver joins the wave channels in succession, catching the end of every wave. The amount of each wave that it catches determines the aggregate rate. (a) Catching the same part of each wave results in constant average 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.

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.

Sections III-A–III-C below introduce the basics of receiver operation; the full details are given in Section VI-D.4. 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.

A. 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 $\mathtt{NWC}$ to $\mathtt{NWC}+1$ increases the reception rate by a multiplicative factor

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

Thus the next wave is joined if $\Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P}$ is at most the target rate, where $\mathtt{ARR\_P}$ is the receiver's estimate of the aggregate rates of the subscribed channels.[5]

The target rate is continually updated based on a set of measured parameters including the average loss probability and the average MRTT. The target rate is given by

$$\mathtt{TRATE\_P} = \min\left\{\max\{\mathtt{SSR\_P},\, \mathtt{REQN\_P}\},\, \mathtt{MRR\_P}\right\} \tag{5}$$

where

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

These two equations capture many of the principles of WEBRC. First, the equation-based rate $\mathtt{REQN\_P}$ is an analogue of Eq. (1) where $\mathtt{ARTT}$ is an average of the receiver's measurements of MRTT and $\mathtt{LOSSP}$ is the receiver's estimate of the loss event rate.[6] Thus, the computed rate is

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

[6]As in TFRC, a loss event is a period of duration $\mathtt{ARTT}$ 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 $\Gamma_{\texttt{NWC}} \cdot \texttt{ARR\_P} \leq \texttt{TRATE\_P}$ 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 II
Essentials of receiver operation

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. 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 the true reception rate TRR_P when packet losses are detected. Details on how measurements are made and state variables are updated are given in Section VI-D.3.

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.

The basic actions of the receiver are summarized in Table II. This excludes initializations, accounting for the finite number of active channels, join timeouts, and the mechanism described in Section III-C. The full details are given in Section VI-D.

B. 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 $\texttt{SSR\_P} = \infty$ and replaces Eq. (5) with

$$\texttt{TRATE\_P} = \min \{4 * \texttt{TRR\_P}, \texttt{MRR\_P}\}. \tag{7}$$

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 $\texttt{P}^{t/\texttt{TSD}}$ 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.[7] 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 TRR_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.

[7]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.

C. 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 $\mathtt{P}^{t/\mathtt{TSD}}$, 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.

IV. 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.

A. 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.

An important concept for a session subject to congestion control is the amount of time after a congestion event at a network element that the element is vulnerable to further congestion due to actions of a receiver or the sender initiated before the receiver or the sender learns of the congestion. To define this, we first need to introduce the idea of non-causal events. Causality of events is defined with respect to the flow of information through the network based on the protocol. For example with TCP, when the receiver sends an acknowledgement packet to the sender, the actions of the sender are only affected after it has received the acknowledgement. Similarly, when a packet is lost at a network element, the actions of the receiver are only affected after the packet flow has had time to reach the receiver. In general, let (T_i, P_i) be the time and place of the occurrence of event E_i. Then, a pair of events (E_1, E_2) are *non-causal* if T_1 plus the time it takes information in the context of the protocol to flow from P_1 to P_2 is greater than T_2 while it is also true that T_2 plus the time it takes information in the context of the protocol to flow from P_2 to P_1 is greater than T_1. For example in TCP, suppose event E_1 is the loss of a packet at a network element and E_2 is any event that occurs at that same network element after E_1 but before the information about event E_1 has time to travel via data packets from the element to the receiver, then via acknowledgement packets from the receiver to the sender, and then via data packets from the sender back to the element. Then, E_1 and E_2 are non-causal.

A congestion event occurs at a network element when a packet is lost at the element.[8] The *congestion aftermath* is the period of time after a congestion event at a network element that relevant, non-causal events affect the flow of packets through the network element. For TCP, the non-causal events that are relevant to the congestion are the arrivals of data packets at the element that are sent prior to the sender receiving information about the loss from the receiver, and thus the end of the congestion aftermath is when the last such data packet arrives to the element. The arrivals of these data packets to the element are relevant to the congestion because their arrival rate is increasing, unmitigated by the loss, and they can cause further congestion. This motivates designing the aggressiveness of the receiver reception rate so that it is inversely governed by the duration of the congestion aftermath, because the longer the congestion aftermath the less responsive the flow is to loss and thus the smaller the reception rate should be to limit the amount of loss when it occurs. It can be easily seen that the duration of the congestion aftermath for a TCP connection is approximately the RTT value at the time the loss occurs. Thus, because the TCP reception rate is inversely related to the RTT, this implies that the TCP

[8]The same definition holds when "packet loss" is replaced with "congestion is detected and marked in the packet using an ECN flag."

reception rate is inversely governed by the duration of the congestion aftermath. This is another justification for the receiver 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 wide-area network it may create a large discrepancy in rates between competing flows.

B. 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 packet flow from that wave is received. 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.[9] 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.

C. 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.

D. MRTT examples

We now give a couple of examples that show some of the properties of the MRTT. In the first example, the multicast tree consists of a line of routers with one receiver hanging off each router, as shown in Fig. 3. Here, all the receivers have the same target reception rate and thus they will all have the same join time. What is interesting about this example is that all of the receivers have the same MRTT value, despite the fact that some of the receivers are much farther from the sender than others. The reason for this is that the closer receivers help to draw the packets through the tree to reduce the MRTT for the farther receivers to the same value. Thus, the target reception rates of the receivers in this tree, which depends on their MRTT values, will all stay the same.

The example in Fig. 4 shows the same multicast tree, but in this case receiver B has a higher target rate and therefore an earlier join time than the other two receivers. This state could have been obtained from the state shown in Fig. 3 by receiver B increasing its reception rate while the other receivers stayed at the same reception rate. In this state, receiver B measures the highest MRTT value, and the other two receivers measure a lower MRTT value. This will tend to reduce the target rate of receiver B and increase the target rates of the other receivers, bringing the target rates of all the receivers closer together.

Fig. 5 shows an example where there is a network element that is far from the sender and each receiver has a short connection to this network element. This example shows the relationship between the join time of each receiver and

[9]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.

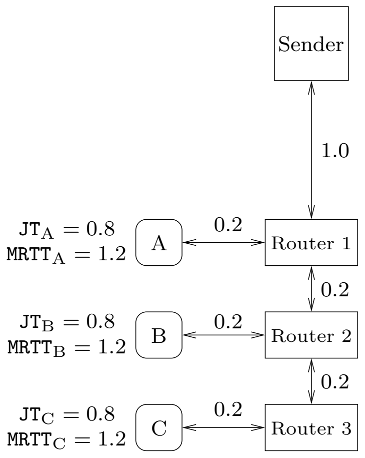

Sender
1.0
$\mathtt{JT}_{\mathrm{A}} = 0.8$
$\mathtt{MRTT}_{\mathrm{A}} = 1.2$
A
0.2
Router 1
0.2
$\mathtt{JT}_{\mathrm{B}} = 0.8$
$\mathtt{MRTT}_{\mathrm{B}} = 1.2$
B
0.2
Router 2
0.2
$\mathtt{JT}_{\mathrm{C}} = 0.8$
$\mathtt{MRTT}_{\mathrm{C}} = 1.2$
C
0.2
Router 3

Fig. 3. A multicast tree with all receivers in a line and all with the same join time. Note that all the MRTT values are the same, despite the fact that some of the receivers are farther from the sender than others.

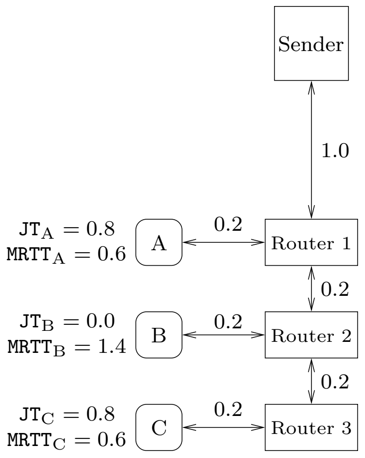

Sender
1.0
$\mathtt{JT}_{\mathrm{A}} = 0.8$
$\mathtt{MRTT}_{\mathrm{A}} = 0.6$
A
0.2
Router 1
0.2
$\mathtt{JT}_{\mathrm{B}} = 0.0$
$\mathtt{MRTT}_{\mathrm{B}} = 1.4$
B
0.2
Router 2
0.2
$\mathtt{JT}_{\mathrm{C}} = 0.8$
$\mathtt{MRTT}_{\mathrm{C}} = 0.6$
C
0.2
Router 3

Fig. 4. A multicast tree with all receivers in a line and Receiver B with an earlier join time than the other two receivers. Note that the MRTT value of Receiver B is largest, and the other two receivers' MRTT values are smaller.

its resulting measured MRTT value. Note that the earlier the join time of a receiver the larger its measured MRTT value is. This effect tends to equalize the target reception rates of the receivers.

E. MRTT global properties

In this subsection the global properties of an MRTT measurement for a wave are described. 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 $\mathtt{RTT}_i$ be the RTT of cycle C_i, *i.e.* $\mathtt{RTT}_i$ 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

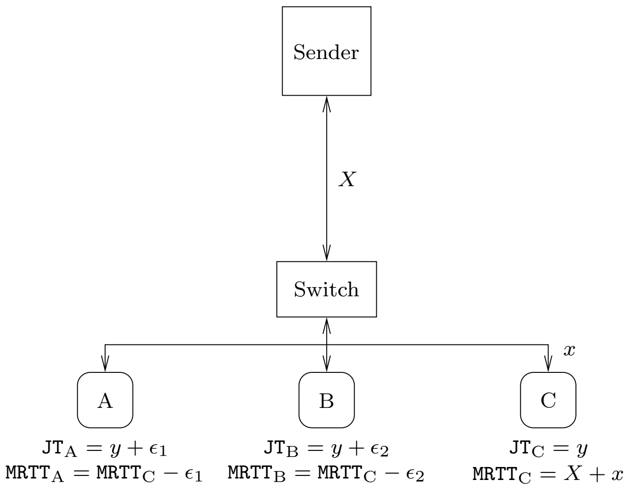

Sender
X
Switch
x
A
B
C
$\mathtt{JT}_{\mathrm{A}} = y + \epsilon_1$
$\mathtt{MRTT}_{\mathrm{A}} = \mathtt{MRTT}_{\mathrm{C}} - \epsilon_1$
$\mathtt{JT}_{\mathrm{B}} = y + \epsilon_2$
$\mathtt{MRTT}_{\mathrm{B}} = \mathtt{MRTT}_{\mathrm{C}} - \epsilon_2$
$\mathtt{JT}_{\mathrm{C}} = y$
$\mathtt{MRTT}_{\mathrm{C}} = X + x$

Fig. 5. A multicast tree with all receivers hanging off a network element at the same distance. Without loss of generality, Receiver C is assumed to have the earliest join time. The MRTT values are then as shown, assuming $\epsilon_1 \leq X$ and $\epsilon_2 \leq X$.

at N_i to travel to receiver i once the forwarding state has been established. Let $\mathtt{JT}_i$ be the join time of receiver i, *i.e.*, $\mathtt{JT}_i$ is the wave time at i when i sends a join to the wave. Let $A_i = \mathtt{JT}_i + \mathtt{RTT}_i$ for receiver i. Note that A_i is the wave time at N_i when the join sent at wave time $\mathtt{JT}_i$ 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 $\mathtt{MRTT}_i = Q_i - \mathtt{JT}_i$.

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 $Q_i = T_i > A_i = \mathtt{JT}_i + \mathtt{RTT}_i$ and thus $\mathtt{MRTT}_i = Q_i - \mathtt{JT}_i > \mathtt{RTT}_i$. 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 $Q_i = A_i = \mathtt{JT}_i + \mathtt{RTT}_i$ and thus $\mathtt{MRTT}_i = Q_i - \mathtt{JT}_i = \mathtt{RTT}_i$. 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.

For example, in Fig. 6 the cycle C_{B} from B to the sender and back is taut and the cycle C_{D} from D to Router 4 and back is taut but the cycle C_{A} from A to Router 1 and back is slack. Each slack cycle is the *child* of a unique taut cycle; conversely, a taut cycle can be the *parent* of several slack cycles. In Fig. 6, C_{B} is the parent of C_{A} and C_{A} is the

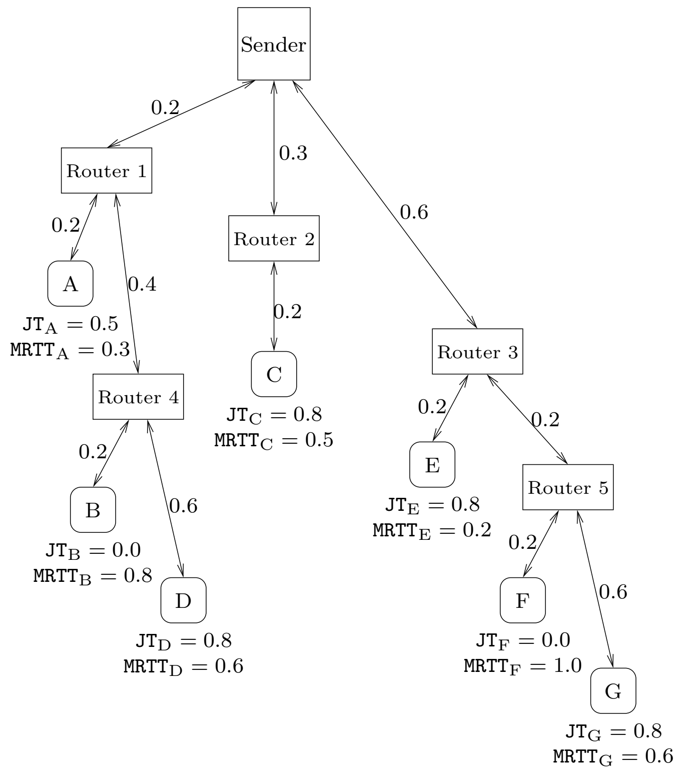


Fig. 6. A multicast tree and the final MRTT values for the receivers.

child of C_B. As another example, in Fig. 3 the cycle C_A between A and the sender is taut and is the parent of the slack cycles associated with other receivers.

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.

F. Buffer filling

In this subsection the effect of buffer filling on the MRTT values of the receivers is described. 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 up to the current value of $\mathtt{MRTT}_i$ then C_i becomes taut. For example, in Fig. 6, if the RTT of C_A increases by 0.1 seconds from its current value of 0.2 seconds to $\mathtt{MRTT}_A = 0.3$ seconds, then C_A is taut and any further increases in its RTT will lead to an increase in $\mathtt{MRTT}_A$. 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 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.

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.

G. WEBRC congestion aftermath

The congestion aftermath for WEBRC is defined as follows. A non-causal event that is relevant to a congestion event for WEBRC is any join by a receiver that occurs after the congestion event has occurred at a network element interface but before the congestion has been detected by the receiver. This is because this is the only non-causal event relative to the congestion that can increase the packet rate flow through the interface. There may be many such relevant non-causal events for a particular congestion event, but only the relevant join that triggers the first data packet from the wave to arrive at the interface will increase the rate of flow through the interface.

The duration of the congestion aftermath can be analyzed as follows. Suppose there is a lost packet at a network element interface at wave time X with respect to the next wave that receivers will join but have not yet joined. Only joins to this wave sent by receivers below the interface at a join time that is less than X that would arrive at the interface at a wave time greater than X with respect to the interface are non-causal and relevant.

If there are no relevant joins then there is no congestion aftermath for that loss event. If there is at least one relevant join then let T be the wave time at the interface when the first data packet arrives at the interface in response to a relevant join. Then $T - X$ is the duration of the congestion aftermath. Note that T satisfies $\mathtt{MRTT}_{\mathrm{Y}} \geq T - \mathtt{JT}_{\mathrm{Y}}$ for all receivers Y below the interface. Since $\mathtt{JT}_{\mathrm{Y}} \leq X$ for a relevant join from any receiver Y, this means that $\mathtt{MRTT}_{\mathrm{Y}} \geq T - X$ for a relevant join from any receiver Y. But, since $T - X$ is the duration of the congestion aftermath, this means that the $\mathtt{MRTT}_{\mathrm{Y}}$ measurement just after the loss for each relevant join from a receiver Y is at least the duration of the congestion aftermath. Thus, because the reception rate of each WEBRC receiver is inversely related to its MRTT, this implies that the reception rate for each WEBRC receiver is inversely governed by the duration of the congestion aftermath. Furthermore, all receivers that send relevant non-causal joins relative to a congestion event receive a measurement of their MRTT based on their join that is at least the duration of the congestion aftermath. Thus, the impact on the MRTT of a receiver due to a non-causal relevant join is almost immediate. This is another reason the definition of the MRTT is natural, and why the reception rate of each WEBRC should be inversely related to its MRTT.

As an example, in Fig. 6 suppose there is a loss at wave time 0.51 with respect to the next wave to be joined on the leftmost interface out of the sender. Note that since $\mathtt{JT}_{\mathrm{D}} = 0.8 > 0.51$, receiver D will detect the loss before its join time and in reaction will probably defer its join to a later time, and in any case this is not a relevant join. Both receivers A and B will not detect the loss before their join time and thus they will send the join before the loss is detected. The join from receiver B will trigger the delivery of a data packet over the sender interface 0.29 seconds after the loss is detected at the interface, and this is the duration of the congestion aftermath. Note that both $\mathtt{MRTT}_{\mathrm{A}} = 0.3$ and $\mathtt{MRTT}_{\mathrm{B}} = 0.8$ are larger than the duration of the congestion aftermath, and that both of these receivers obtain these MRTT measurements due to their sending non-causal relevant joins.

H. Relative scaling

In this subsection the relative scaling properties of WEBRC are described. 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 $1, 2, \ldots, n$ then the measured MRTT values $\mathtt{MRTT}_1, \mathtt{MRTT}_2, \ldots, \mathtt{MRTT}_n$ will be the same with respect to join times $\mathtt{JT}_1, \mathtt{JT}_2, \ldots, \mathtt{JT}_n$ and with respect to the join times $\mathtt{JT}_1 + x, \mathtt{JT}_2 + x, \ldots, \mathtt{JT}_n + x$ for any value of x. Note that the target rate $\mathtt{TRATE}_i$ of receiver i is proportional to $1/\mathtt{P}^{\mathtt{JT}_i}$. This implies that the measured MRTT values $\mathtt{MRTT}_1, \mathtt{MRTT}_2, \ldots, \mathtt{MRTT}_n$ will be the same with respect to the set of target rates $\mathtt{TRATE}_1, \mathtt{TRATE}_2, \ldots, \mathtt{TRATE}_n$ and the set of target rates $c \cdot \mathtt{TRATE}_1, c \cdot \mathtt{TRATE}_2, \ldots, c \cdot \mathtt{TRATE}_n$ 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 $\mathtt{TRATE}_i/\mathtt{TRATE}_j$ is approximated by $(\sqrt{p_j} \cdot \mathtt{MRTT}_j)/(\sqrt{p_i} \cdot \mathtt{MRTT}_i)$, 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.

I. Pairwise coordination

In this subsection we investigate the coordination of the reception rates of two receivers due to the interactions between their MRTT values. The importance of 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. 7 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 Section IV-H, $\mathtt{MRTT}_{\mathrm{A}}$ and $\mathtt{MRTT}_{\mathrm{B}}$ depend only on the difference between $\mathtt{JT}_{\mathrm{A}}$ and $\mathtt{JT}_{\mathrm{B}}$, and not on their absolute values. Define $\Delta\mathtt{JT}_{\mathrm{A,B}} = \mathtt{JT}_{\mathrm{A}} - \mathtt{JT}_{\mathrm{B}}$ and $\Delta\mathtt{MRTT}_{\mathrm{A,B}} = \mathtt{MRTT}_{\mathrm{A}} - \mathtt{MRTT}_{\mathrm{B}}$. Fig. 8 plots $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ as a function of $\Delta\mathtt{JT}_{\mathrm{A,B}}$. In this plot, the value of $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ is a negative fixed value for large enough positive values of $\Delta\mathtt{JT}_{\mathrm{A,B}}$, $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ is a positive fixed value for small enough negative values of $\Delta\mathtt{JT}_{\mathrm{A,B}}$, and $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ transitions linearly between the two when the absolute value of $\Delta\mathtt{JT}_{\mathrm{A,B}}$ is small. The linear transition interval is of length $2X$ in both dimensions, and the transition passes through the origin. The plot will pass through the origin as long as $|Y - Z| \leq X$.

Fig. 8 shows that $\Delta\mathtt{JT}_{\mathrm{A,B}}$ and $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ are anti-correlated, *i.e.*, if $\Delta\mathtt{JT}_{\mathrm{A,B}} > 0$ then $\Delta\mathtt{MRTT}_{\mathrm{A,B}} < 0$ and if $\Delta\mathtt{JT}_{\mathrm{A,B}} < 0$ then $\Delta\mathtt{MRTT}_{\mathrm{A,B}} > 0$.

Let $\mathtt{TRATE}_{\mathrm{A}}$ and $\mathtt{TRATE}_{\mathrm{B}}$ be the respective target reception rates of A and B. Because each wave decreases by a factor of $\mathtt{P}$ each $\mathtt{TSD}$ seconds, $\mathtt{TRATE}_{\mathrm{B}}/\mathtt{TRATE}_{\mathrm{A}} = 1/\mathtt{P}^{\Delta\mathtt{JT}_{\mathrm{A,B}}/\mathtt{TSD}}$. Using Eq. (2), $\mathtt{TRATE}_{\mathrm{B}}/\mathtt{TRATE}_{\mathrm{A}}$ is also approximately equal to $(\sqrt{p_{\mathrm{A}}} \cdot \mathtt{MRTT}_{\mathrm{A}})/(\sqrt{p_{\mathrm{B}}} \cdot \mathtt{MRTT}_{\mathrm{B}})$, where p_{A} and p_{B} are the respective loss rates of A and B.

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. Since the join times for each subsequent wave are inversely related to the MRTT measurements from previous waves, this implies that if $\Delta\mathtt{JT}_{\mathrm{A,B}} > 0$ then the next value of $\Delta\mathtt{JT}_{\mathrm{A,B}}$ will be smaller, and if $\Delta\mathtt{JT}_{\mathrm{A,B}} < 0$ then the next value of $\Delta\mathtt{JT}_{\mathrm{A,B}}$ will be larger. This implies that the only fixed point in this plot is the origin, *i.e.*, at equilibrium $\mathtt{JT}_{\mathrm{A}} = \mathtt{JT}_{\mathrm{B}}$ and $\mathtt{MRTT}_{\mathrm{A}} = \mathtt{MRTT}_{\mathrm{B}}$ when the loss rates are equal, and thus at equilibrium the reception rates of A and B are the same. Furthermore, the closer A and B are to each other relative to their distance from the sender, the more powerful the

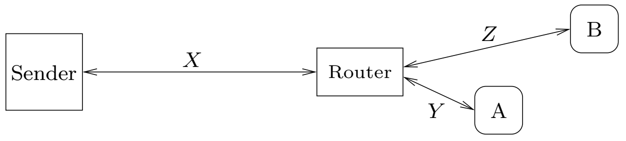


Fig. 7. Two nearby receivers A and B connected to a faraway sender through the same router.

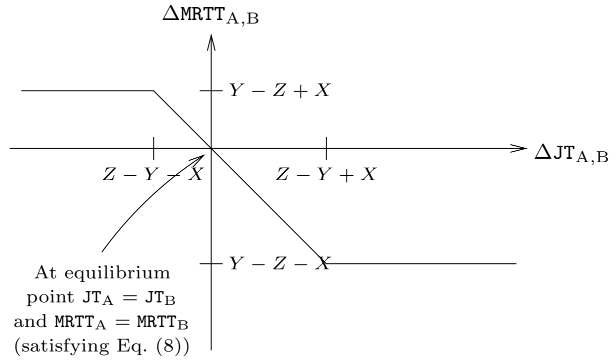


Fig. 8. Plot of $\Delta\texttt{MRTT}_{\mathrm{A,B}}$ as a function of $\Delta\texttt{JT}_{\mathrm{A,B}}$ for Fig. 7.

MRTT influence towards equilibrium.

For general loss rates when p_{A} and p_{B} are not necessarily equal, with respect to Eq. (2) equilibrium is achieved when

$$\Delta\texttt{JT}_{\mathrm{A,B}} = \texttt{TSD} \cdot \log_{1/\texttt{P}} \left(\frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} \cdot \frac{\texttt{MRTT}_{\mathrm{A}}}{\texttt{MRTT}_{\mathrm{B}}} \right). \tag{8}$$

The relationship between loss rates and target rates at equilibrium can be roughly summarized as follows.

$$\begin{array}{rl}
\frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} < \frac{Z}{Y+X}: & \frac{\texttt{TRATE}_{\mathrm{B}}}{\texttt{TRATE}_{\mathrm{A}}} \approx \frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} \cdot \frac{Y+X}{Z} \\
\frac{Z}{Y+X} < \frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} < \frac{Z+X}{Y}: & \frac{\texttt{TRATE}_{\mathrm{B}}}{\texttt{TRATE}_{\mathrm{A}}} \approx 1 \\
\frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} > \frac{Z+X}{Y}: & \frac{\texttt{TRATE}_{\mathrm{B}}}{\texttt{TRATE}_{\mathrm{A}}} \approx \frac{\sqrt{p_{\mathrm{A}}}}{\sqrt{p_{\mathrm{B}}}} \cdot \frac{Y}{Z+X}
\end{array}$$

Thus, the coordination due to the MRTT values tends to equalize the reception rates of the receivers even when loss rates are not equal.

Fig. 9 plots both $\texttt{MRTT}_{\mathrm{A}}$ and $\texttt{MRTT}_{\mathrm{B}}$ as a function of $\Delta\texttt{JT}_{\mathrm{A,B}}$. In the interval where $\texttt{MRTT}_{\mathrm{B}}$ is increasing, the cycle C_{B} between B and the sender and back is slack and the cycle C_{A} between A and the router and back is taut; conversely, in the interval where $\texttt{MRTT}_{\mathrm{A}}$ is decreasing, C_{A} is slack and C_{B} is taut. In the intervals where both $\texttt{MRTT}_{\mathrm{A}}$ and $\texttt{MRTT}_{\mathrm{B}}$ are constant, both C_{A} and C_{B} are taut.

Variations in the difference between the join times of a receiver associated with a slack cycle and a receiver associated with the parent taut cycle affects the MRTT of the receiver associated with the slack cycle, *i.e.*, if C_i is a child of C_j, $\Delta\texttt{JT}_{i,j}$ affects $\texttt{MRTT}_i$. In contrast, recall from Section IV-F that variations in the RTT of a taut cycle affect the MRTT of the taut cycle and the MRTT of its slack cycle children. Thus, in some sense the slack cycles and the taut cycles are playing different signaling roles. The reaction of the MRTTs of taut cycles and their children signal changes in latency due to buffer filling, whereas the reaction of the MRTTs of slack cycles signal changes in the differences between reception rates of receivers.

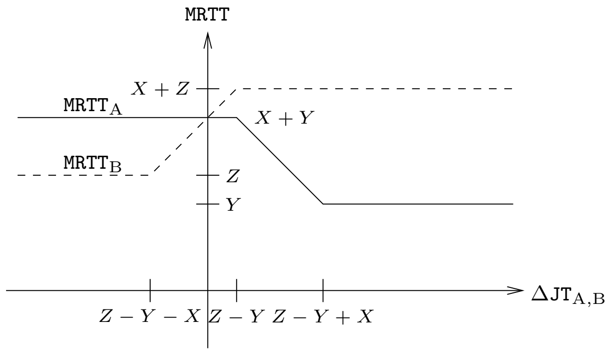


Fig. 9. Plot of $\texttt{MRTT}_{\mathrm{A}}$ and $\texttt{MRTT}_{\mathrm{B}}$ as a function of $\Delta\texttt{JT}_{\mathrm{A,B}}$.

Fig. 10 is similar to Fig. 7 except that the receivers A and B are much farther apart from each other compared to their distance from the sender. Fig. 11 plots $\Delta\texttt{MRTT}_{\mathrm{A,B}}$ as a function of $\Delta\texttt{JT}_{\mathrm{A,B}}$ for this example. In this plot, the value of $\Delta\texttt{MRTT}_{\mathrm{A,B}}$ is a negative fixed value for small enough negative values of $\Delta\texttt{JT}_{\mathrm{A,B}}$ and it is an even smaller negative fixed value for large enough positive values of $\Delta\texttt{JT}_{\mathrm{A,B}}$ and $\Delta\texttt{MRTT}_{\mathrm{A,B}}$ transitions linearly between the two in an interval where $\Delta\texttt{JT}_{\mathrm{A,B}}$ is positive and close to zero. The linear transition interval is of length $2X$ in both dimensions, and the transition does not pass through the origin, because $|Y - Z| > X$.

Similar to Fig. 8, Fig. 11 shows that $\Delta\texttt{JT}_{\mathrm{A,B}}$ and $\Delta\texttt{MRTT}_{\mathrm{A,B}}$ are anti-correlated, and thus the MRTT helps to equalize the reception rates of the receivers. In this case, if the loss rates are equal and if $\Delta\texttt{JT}_{\mathrm{A,B}} > 0$ then the next value of $\Delta\texttt{JT}_{\mathrm{A,B}}$ will be smaller because $\Delta\texttt{MRTT}_{\mathrm{A,B}} < 0$ at every point. This implies that the fixed point in this example when the loss rates of the two receivers are equal is where $\texttt{JT}_{\mathrm{A}} < \texttt{JT}_{\mathrm{B}}$ and $\texttt{MRTT}_{\mathrm{A}} < \texttt{MRTT}_{\mathrm{B}}$ and this occurs when $\texttt{TRATE}_{\mathrm{A}}/\texttt{TRATE}_{\mathrm{B}} = Z/(Y + X)$. Note that if A and B were not coordinated, *i.e.* participating in independent TCP sessions, then the ratio of their reception rates at equilibrium would be $(Z + X)/(Y + X)$. Thus, at the WEBRC equilibrium, the target reception rate of A is higher than it is for B but still their reception rates are tending to equalize compared to their uncoordinated behavior. In general, when p_{A} and p_{B} are not necessarily equal coordination and equalization of the reception rates of A and B occurs and equilibrium is achieved when Eq. (8) is satisfied. Furthermore, the larger the value of X in relationship to $|Y - Z|$, the more powerful the attraction to the fixed point.

J. Global coordination

In this subsection we investigate the coordination of the reception rates of all receivers joined to a session due to the interactions between their MRTT values. It turns out that the properties of the MRTTs and the inverse relationship between the target reception rates and the MRTTs have

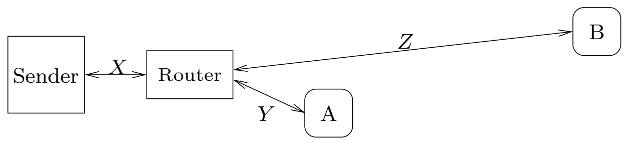


Fig. 10. Two pairwise distant receivers A and B connected to a sender through the same router.

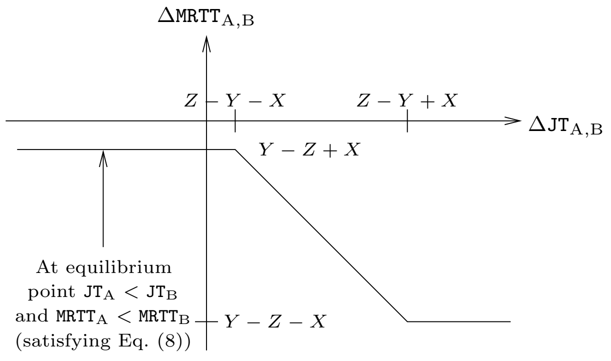


Fig. 11. Plot of $\Delta\mathtt{MRTT}_{\mathrm{A,B}}$ as a function of $\Delta\mathtt{JT}_{\mathrm{A,B}}$ for Fig. 10.

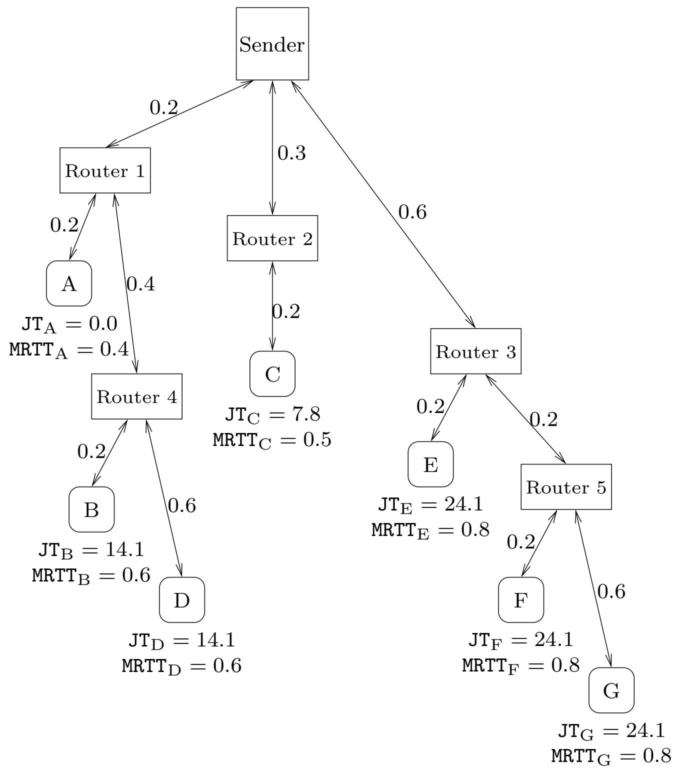

Sender
0.2
0.3
0.6
Router 1
Router 2
Router 3
Router 4
Router 5
0.2
0.4
0.2
0.2
0.2
0.2
0.6
0.2
0.6
A
$\mathtt{JT}_{\mathrm{A}} = 0.0$
$\mathtt{MRTT}_{\mathrm{A}} = 0.4$
B
$\mathtt{JT}_{\mathrm{B}} = 14.1$
$\mathtt{MRTT}_{\mathrm{B}} = 0.6$
C
$\mathtt{JT}_{\mathrm{C}} = 7.8$
$\mathtt{MRTT}_{\mathrm{C}} = 0.5$
D
$\mathtt{JT}_{\mathrm{D}} = 14.1$
$\mathtt{MRTT}_{\mathrm{D}} = 0.6$
E
$\mathtt{JT}_{\mathrm{E}} = 24.1$
$\mathtt{MRTT}_{\mathrm{E}} = 0.8$
F
$\mathtt{JT}_{\mathrm{F}} = 24.1$
$\mathtt{MRTT}_{\mathrm{F}} = 0.8$
G
$\mathtt{JT}_{\mathrm{G}} = 24.1$
$\mathtt{MRTT}_{\mathrm{G}} = 0.8$

Fig. 12. Equilibrium when all loss probabilities are equal.

a beneficial global influence in coordinating the receiver reception rates.

Consider the case when the multicast tree is fixed, all the latencies in the tree are fixed, the set of receivers is fixed and the loss probabilities of all receivers is the same. Although this is a special case, it does illustrate the coordinating effect of the MRTT, and it can occur if the receivers are below a common bottleneck link. Based on the analysis given in the other subsections of this section, it can be shown that there is a unique equilibrium set of MRTT values. This defines a particular cycle structure, with some taut cycles and possibly some slack cycles. Each taut cycle defines an equivalence class consisting of the set of receivers associated with the taut cycle and its slack cycle children, and all receivers in an equivalence class will have the same equilibrium target reception rate. The relative target reception rates of the equivalence classes can be determined by finding the set of join times and MRTT values so that Eq. (8) is satisfied for each pair of equivalence classes.

As an example, Fig. 12 shows the equilibrium target reception rates for a particular multicast tree when all loss probabilities are equal. The equivalence classes for this example, listed in decreasing order of target reception rates, are $\{\mathrm{A}\}$, $\{\mathrm{C}\}$, $\{\mathrm{B,D}\}$ and $\{\mathrm{E,F,G}\}$. Eq. (8) was used to calculate the equilibrium join times.

K. 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.

V. Multicast Congestion Control Comparisons

There has been a significant amount of previous work on multicast congestion control protocols. This section briefly describes previous protocols and compares them to WEBRC.

A. *Single rate congestion control*

With single rate congestion control, all receivers in a multicast session receive data at the same rate. Thus, the sender attempts to determine the appropriate transmission rate for the receiver with the worst network conditions.[10] Examples of single rate multicast congestion control protocols include PGMCC [24] and TFMCC [28].

Several obstacles to designing a single rate congestion control protocol are outlined in [6]. Principal among these is that feedback of the loss rate from each receiver to the sender must be avoided to prevent implosion. For the calculation of the sending rate it is sufficient to hierarchically aggregate feedback or somehow inhibit the least useful feedback [28]. One of the major limitations of a single rate protocol is that all receivers are receiving packets at the same rate, and the rate is that of the worst case receiver in the session, *i.e.*, the rate that is comfortable for the receiver with the worst connection to the sender. Thus, as more and more receivers join the session the reception rate of each individual receiver tends to degrade. Although this is suitable in homogeneous networking environments, or in heterogeneous environments where speedy delivery is not a concern, this is not attractive in a heterogeneous network environment when delivery speed is an issue.

A recent work that builds on TFRC is TCP-friendly multicast congestion control (TFMCC) [28]. This is a single rate congestion control protocol where each receiver calculates a target reception rate based on its own measured loss rate and its own measured unicast RTT to the sender. Then, the receivers continually report these rates back to the sender in a scalable fashion, and the sender adjusts its sending rate based on the lowest reported rate.

B. *Multiple rate congestion control*

Multiple rate congestion control protocols allow the various receivers in a single session to receive data at different rates. The sender sends packets to multiple multicast channels, and receivers adjust their reception rates by joining and leaving these channels. Each receiver determines its own appropriate reception rate and adjusts accordingly, independent of other receivers. Examples of multiple rate congestion control protocols include RLM [19], RLC [26], FLID-DL [1] and now WEBRC.

The potential benefits of multiple rate congestion control compared to single rate congestion control are twofold. The first benefit is that the sender's behavior is independent of and oblivious to the number of receivers participating in the session; in particular, there is no feedback from receivers to the sender. Thus, multiple rate congestion control has the promise of being able to scale to a potentially limitless number of receivers using a single sender. The second benefit is that, because the reception rate of each individual receiver adjusts to the available bandwidth between the sender and that receiver, there is the potential to deliver data to each individual receiver at the fastest possible rate for that receiver, even in a highly heterogeneous network architecture, using a single sender. In stark contrast to single rate congestion control, as more receivers join a multiple rate congestion control session the reception rate of each individual receiver can improve.

WEBRC is related to the Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) protocol [1], which was inspired by RLC [26], which in turn is related to RLM [19]. In the remainder of this section we briefly review these previous protocols.

C. *RLM*

The technique of congestion-controlled cumulative layered multicast was first proposed by McCanne, Jacobson and Vetterli [19] in the context of packet video transmission to large, heterogeneous audiences. Their Receiver-driven Layered Multicast (RLM) protocol achieves scalability by using a receiver-driven methodology, in which the receivers tune their subscription level by joining and leaving layers. They advocate an approach in which receivers drop a layer when they experience packet loss and periodically perform *join experiments* by subscribing to an additional layer.

There are several challenges that this approach introduces. First, one receiver's join experiments can introduce packet loss at other receivers behind the same bottleneck link, producing a potential source of unfairness or inefficiency. Second, standard approaches to cumulative layered multicast have exponentially increasing rates over the layers, which implies that the frequency of join experiments across the layers must be carefully designed to be friendly to TCP traffic and other sessions. Addressing these challenges motivated Vicisano, Rizzo, and Crowcroft to propose their RLC protocol.

D. *RLC*

Receiver-driven Layered Congestion Control (RLC) [26] was designed to provide a TCP-friendly multiple rate congestion control scheme that scales to large audience sizes, requires no modifications to routers or routing protocols, and does not require any coordination amongst receivers. For full scalability, a receiver-driven approach is required, as maintenance of per-receiver state at the source is infeasible and unscalable. But, uncoordinated join experiments by receivers pose substantial problems, as was observed in [19]. The authors of RLC cleverly avoid this problem by synchronizing join experiments. The source places *synchronization points* or *increase signals* into packets, where receivers can now only add a given layer after an appropriate increase signal for that layer. These increase signals are also cumulative, *i.e.*, an increase signal j indicates that all receivers whose maximum subscription level is at most j

[10] To prevent all receivers from suffering when one receiver has a particularly poor connection, rules may be put in place to force such a receiver to unsubscribe.

can join a single additional layer. The use of cumulative increase signals solves the problem of synchronizing receivers behind a shared bottleneck, since when one receiver joins a layer that exceeds the bottleneck bandwidth, all other receivers behind that bottleneck will have also joined a layer. Then, all receivers that experience packet loss will drop back to their original rate prior to the join experiment.

By oversubscribing, a receiver can push the network into a state of congestion. To alleviate the congestion, the receiver must then unsubscribe from a layer by performing an IGMP leave operation. These leave operations often incur substantial latency, leaving the network in a congested state for a prolonged period. Because of this substantial cost of oversubscription, RLC includes a mechanism to prevent some of the joins that would likely lead to oversubscription. The RLC source periodically injects a brief burst of packets on each layer prior to a synchronization point on that layer. The burst on layer i is designed to simulate the rate of layer $i+1$, the idea being that those receivers that lose packets during the burst learn that adding layer $i+1$ is unsafe, without incurring the cost of a join and leave operation. Unfortunately, if a receiver does not lose a packet in the burst, it still has no guarantee that adding the layer is safe, since the burst may be of insufficient length to induce packet loss. Thus a receiver is still prone to oversubscription. The complexity and lingering uncertainty associated with avoiding costly IGMP operations is one of the main problems with RLC.

Another challenge addressed by RLC is the problem of appropriately orchestrating synchronization signals across the layers. The primary goal for RLC is to be fair to other instances of itself as well as to other congestion control protocols such as TCP. As with most proposed layered multicast schemes, RLC requires that the rates on the layers must be exponentially spaced using a doubling scheme, *i.e.*, the rates on the layers follow the pattern 1, 1, 2, 4, 8, ... Dropping a layer with this scheme performs a TCP-like multiplicative decrease. However, adding a layer immediately doubles the rate, and therefore RLC cannot be TCP-like at a fine granularity, since it cannot perform fine-grained additive increase. However, it performs TCP-like additive increase at a coarse scale by placing increase signals on layer i at a frequency of $1/R_i$, where R_i is the cumulative rate through layer i. When used in conjunction with a doubling scheme on the layer rates, the trajectory induced by this distribution of increase signals corresponds to linear increase on average over large time scales. One issue which RLC does not adequately address is the dramatic fluctuations in network bandwidth consumption and the potential for rapid queue buildup that a doubling scheme can induce.

E. FLID-DL

The design of WEBRC is heavily influenced not only by TFRC, but also by the Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) protocol of Byers *et al.* [1]. The "dynamic layering" in FLID-DL refers to the use of dynamic layers that have highly varying rates over time. Each dynamic layer repeats the pattern of starting at a high maximum rate, decreasing in a series of steps to a zero rate, and then remaining at the zero rate for a period of time called the *quiescent period* (see Fig. 13(a)). The rates of all the dynamic layers follow the same pattern but with different time shifts. At any given time, the receiver is subscribed to a number of dynamic layers K and these are the K lowest-rate dynamic layers at that moment. The receiver controls its reception rate by the number of dynamic layers it subscribes to (see Fig. 13(b)). Note that periodic joins are needed to *maintain* any given average transmission rate.

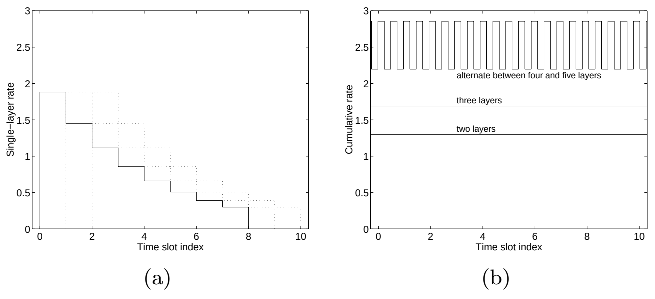


Fig. 13. The dynamic layering of FLID-DL. (a) Transmission rates on each dynamic layer are periodic, decreasing from a high maximum rate to zero in a series of steps and remaining at zero for a quiescent period. (b) Receivers adjust their rates by subscribing to different numbers of dynamic layers.

The decreasing step-like pattern and subsequent quiescent period of each dynamic layer remedies the severe problems associated with slow IGMP leave operations. Rate reductions in RLM or RLC require leaving one or more multicast groups. Large latencies for these leave operations can cause the transmission rate to remain too high for an unacceptably long period, leading to packet loss and disruption of other flows. The declining rates of the dynamic layers creates an implicit rate reduction from the lack of joining.

FLID-DL also introduces methodology for designing a layered scheme where the rate increase step size can be any constant factor. In particular, [1] recommends using a step size factor of 1.3, instead of the factor of 2 used by RLC. This is useful in reducing the large rate increase jumps inherent in RLC, thereby reducing the possible packet loss when new layers are joined. Given a particular step size factor, FLID-DL introduces a scheme for spacing out the increase signals placed into packets in such a way that on average the reception rate increase by receivers without packet loss is linear, *i.e.*, just like in TCP. Like in RLC, FLID-DL uses these increase signals to coordinate the times when receivers will attempt to step up their reception rate.

F. WEBRC versus FLID-DL

Although FLID-DL has been described as the state of the art for multiple rate congestion control in [27], FLID-DL does have some major flaws that WEBRC overcomes, including an excessive rate of join and leave control traffic, an excessive amount of packet loss, worse performance at certain reception rates than at other reception rates, and a lack of sensitivity to RTT.

For both FLID-DL and WEBRC, a receiver sends one IGMP join and leave on average during each time slot. The time slot duration recommended for FLID-DL in [1] is 0.5 seconds. The reason for this small value is primarily because with FLID-DL each rate step increase is by a factor of 1.3, and once a receiver is joined at a particular rate it remains joined at this rate for the duration of the time slot. Thus, if the reception rate before the rate step increase is just below the available bandwidth, the reception rate after the rate step exceeds the available bandwidth by a fixed fraction, which quickly leads to massive packet loss unless the rate is quickly decreased when packet loss occurs. Since a FLID-DL receiver can only decrease its reception rate at the end of a time slot, time slot durations must be short. In addition, if the available bandwidth to a FLID-DL receiver is in between two possible reception rates then the reception rate of the FLID-DL receiver can oscillate between under-utilization of the available bandwidth (causing slower than full-rate delivery of the data) and over-utilization of the available bandwidth (causing high packet loss at network elements).

The recommended time slot duration in WEBRC is 10 seconds. The reason that the time slot duration is so much longer for WEBRC than for FLID-DL is primarily because the WEBRC receiver can fine-tune its reception rate to any value and thus avoid drastic overshooting of the available bandwidth and the resulting massive packet losses. In particular, the WEBRC receiver can join a wave during its smooth descent at the appropriate time to achieve any particular reception rate. To increase its maximum reception rate slightly, a WEBRC receiver simply joins the next wave at a slightly earlier time then the time it joined the previous wave. Similarly, to decrease its maximum reception rate slightly, a WEBRC receiver joins the next wave at a slightly later time then the time it joined the previous wave. Thus, a WEBRC receiver can finely adjust its reception rate to the available bandwidth, providing higher overall link utilization and less packet loss than with FLID-DL.

For RLM and RLC, at low reception rates the amount of control traffic is similar to that for FLID-DL, although at higher reception rates the frequency of control traffic drops off.

Another advantage of the wave-shaped transmission rate is that, similar to a FLID-DL receiver, a WEBRC receiver can decrease its reception rate simply by not joining additional channels (and leaving channels when the sending rate of the wave goes to zero). This is useful in overcoming problems with networks that take a long time to process IGMP leave messages.

Perhaps one of the reasons that there has been no equation-based approach to multiple rate congestion control previous to WEBRC is that, previous to WEBRC, there was no concept of a scalable, receiver-measured round trip time for multiple rate congestion control protocols. Thus, RLM, RLC and FLID-DL receivers do not adjust their reception rates in response to fluctuations in the round trip time. Instead, the RTT value used to determine the aggressiveness of the flow is set to a fixed constant. For example, for a FLID-DL receiver it could be 200 ms. This means that a FLID-DL receiver would compete fairly at a coarse level with a 200 ms RTT TCP session, but would be over-aggressive competing with a TCP session with a larger RTT and under-aggressive competing with a TCP session with a smaller RTT. WEBRC provides a solution to these issues through its use of the MRTT.

Finally, one of the more subtle advantages of WEBRC over RLC and FLID-DL is that the packet format is simpler, requiring no special "increase signal" that signals to the receivers when to attempt to join an additional layer for RLC and FLID-DL. The simplified packet format allows refined versions of WEBRC receivers to be deployed as more sophisticated receiver protocols are developed without having to upgrade already deployed WEBRC receivers and WEBRC senders.

VI. WEBRC Details

Expanding greatly on Sections I and III, this section provides details on WEBRC omitted from [15]. Some of these details are inconsistent with the current protocol specification draft [13] because they are improvements inspired by continuing simulations and by experience with Digital Fountain's implementations of WEBRC.

A WEBRC session is comprised of a set of channels originating from a single sender. At the network layer, a channel can be uniquely identified by a (sender IP address, local channel ID) pair. For multicast, the local channel ID is a multicast group address. For unicast, the local channel ID is an identifier assigned by the sender so that no two local channels associated with the sender are the same. The packets within a WEBRC session carry channel numbers that are logically numbered consecutively from 0 to $\mathtt{T}$. Channels 0, 1, ..., $\mathtt{T}-1$ are called *wave channels* and channel $\mathtt{T}$ is called the *base channel*.

The transmission rates on the base and wave channels are periodic and depend only on time as measured by the sender, not on the number of receivers in the session or on the properties or conditions of the receivers. The transmission rate on the base channel has period $\mathtt{TSD}$ seconds; it starts at its maximum value $\mathtt{BCR}$ at the beginning of each *time slot* and decays exponentially to a fraction $\mathtt{P}$ of its peak value over each time slot as shown by the bold curve in Fig. 1.

Each of the wave channels also has a periodic rate, but both the period and the variation of the rate are much greater than for the base channel. The rate on a wave channel has period $\mathtt{T}\cdot\mathtt{TSD}$ and this period can be divided into $\mathtt{T}$ time slots with *time slot indices* ($\mathtt{TSI}$s) 0, 1, ..., $\mathtt{T}-1$. For $\mathtt{Q}$ consecutive time slots, the channel is *quiescent*, meaning that the rate is zero. A wave channel is called *active* in any time slot in which it is not quiescent. For the $\mathtt{N}=\mathtt{T}-\mathtt{Q}$ active time slots the rate increases quickly to reach a high wave crest rate $\mathtt{WCR}$ and then decreases exponentially with a decay of $\mathtt{P}$ per time slot, ending at a rate of BCR. To a first approximation, the wave channel rates are as shown in Fig. 1. More precisely, the beginning of each wave is modified so that the total sender output across all channels is constant at rate SR and the rate of each wave channel is at least $\mathtt{P} \cdot \mathtt{BCR}$ during all active time slots, subject to some other considerations and exceptions detailed in Section VI-C.2 and [7]. The wave channels and time slots are numbered such that wave channel i becomes quiescent at the end of time slot i.

A WEBRC receiver joins the base channel immediately upon entering a session and remains subscribed to that channel for the duration of the session. The receiver's reception rate varies coarsely depending on the number of wave channels that it is subscribed to, and it varies finely depending on the timing of the join to the highest-rate wave channel it subscribes to. To increase its reception rate by a given amount, the receiver appropriately times its join to the channel currently at the lowest rate amongst the unsubscribed channels. To decrease its reception rate, the receiver does nothing; the decay on every wave channel automatically lowers the rate. (The receiver unsubscribes to wave channels as they become quiescent. The $\mathtt{Q} \cdot \mathtt{TSD}$ second quiescent period is designed to give a large margin for error in completing this operation.) Because of these rules for increasing and decreasing, the subscribed channels are at any time the lowest rate active channels. The common exponential decay factor for all of the channels—including the base channel—makes joining the next wave channel increase the reception rate by a factor that is independent of the time within the time slot.

The remainder of this section details the operations of WEBRC senders and receivers. We begin with a listing of the parameters fixed within a single session, some of which have already been described, in Section VI-A. Then Section VI-B gives the packet header format, the sender is described in Section VI-C, and finally the receiver—where the vast majority of the complexity lies—is described in Section VI-D.

In the opening paragraphs of this section and in Fig. 1, we sometimes neglected to include units on the rates because it only mattered that the units were consistent. Where units are significant, suffixes _b, _P, and _B are used for bits per second, packets per second, and bytes, respectively. For example, BCR_b is the base channel rate in *bits* per second while MRR_P is the maximum receiver reception rate in *packets* per second. Most of the terminology and notation is consistent with the most recent (though slightly out of date) IETF standards-track protocol specification document for WEBRC [13].

A. Session parameters

In WEBRC, the sender has many fewer parameters than the receiver. Thus we first give the sender parameters (most of which apply also to the receiver) and then the receiver parameters.

A.1 Sender inputs and parameters

The primary input to the sender for a session is SR_b. SR_b is the sender transmission rate in bits per second at any point in time in aggregate to all channels. This is also the maximum rate in bits per second at which any receiver could receive data from the session.

The secondary inputs to the sender are listed below. These are secondary because in general their values will be fixed to default values that will not change or because they are set based on non-WEBRC considerations.

- BCR_P is the maximum rate of the base channel in packets per second. The default value is 1.
- LENP_B is the length of packets in bytes. The value of LENP_B depends on the complete protocol, but in general this should be set to as high a value as possible without exceeding the MTU size for the network that would cause fragmentation. The default value is 1024.
- P is the drop in the rate of any channel over the duration of a time slot. The default value is 0.75.
- QD is the minimum quiescent period duration measured in seconds. The default value is 300. The setting for this default is based on the settings of query interval timers within router and switches. The default settings for these timers ensure that in the worst case packets for a multicast group will be forwarded over an interface at a switch or router for at most 150 seconds after the last receiver below that interface has left the group, even if all IGMP and PIM control messages are lost. Note that 300 seconds is twice this worst case time.
- TSD is the time slot duration measured in seconds. The default value is 10.

From these inputs the following sender parameters can be derived:

- $\mathtt{BCR\_b} = 8 \cdot \mathtt{LENP\_B} \cdot \mathtt{BCR\_P}$ is the rate of the base channel in bits per second at the beginning of a time slot. The default values for BCR_P and LENP_B make the base channel rate 8192 bits per second. Note that this is only one-third of the slowest typical dialup connection speed available today.
- $\mathtt{SR\_P} = \mathtt{SR\_b}/(8 \cdot \mathtt{LENP\_B})$ is the sender transmission rate in packets per second.
- $\mathtt{N} = \lceil \log_{1/\mathtt{P}}(1 + (1/\mathtt{P})(1/\mathtt{P} - 1)(\mathtt{SR\_P}/\mathtt{BCR\_P})) \rceil - 1$ is the number of active time slots for a wave channel. N is also the number of wave channels active in every time slot.
- $\mathtt{Q} = \lceil \mathtt{QD}/\mathtt{TSD} \rceil$ is the number of quiescent time slots for a wave channel. (Each wave channel is quiescent for at least $\mathtt{Q} \cdot \mathtt{TSD}$ seconds, which may exceed QD.) The default values for QD and TSD make Q equal 30.
- $\mathtt{T} = \mathtt{N} + \mathtt{Q}$ is the total number of time slots in a cycle. T is also the total number of wave channels.

A.2 Receiver inputs and parameters

Before or upon joining a session, the receiver should have the values of BCR_P, LENP_B, P, TSD, N, Q and T. Some of these values may be fixed or obtained out of band and others may be immediately deduced. For example, the receiver can obtain LENP_B and T from the first packet received from the base channel, and the receiver could measure BCR_P once it is joined to the base channel. The values of P, Q and TSD may be fixed to default values built into the receiver if they do not change from session to session, and the value of N can be computed as $\mathtt{T}-\mathtt{Q}$. For multicast, the receiver must also know the mapping between channel numbers and multicast group addresses; for unicast, the channel identities assigned by the sender must be known.

The receiver has additional parameters which are generally fixed for the duration of a session. Parameters for which recommended values can be derived from other WEBRC parameters are described as their roles arise in Section VI-D. The following parameter affects receiver behavior and has no relevance to the sender:

- MRR_b is the maximum receiver reception rate in bits per second. The receiver will not join a wave channel if it increases the anticipated reception rate above MRR_b. Thus, an approximate upper bound to the reception rate in a session is the minimum of SR_b and MRR_b. A recommended value for MRR_b is the bandwidth capacity of the last link to the receiver. MRR_P is the maximum receiver reception rate in packets per second, *i.e.*, $\mathtt{MRR\_P} = \mathtt{MRR\_b}/(8\cdot\mathtt{LENP\_B})$.

B. Packet header format

A WEBRC session may use either "short format" or "long format" congestion control information. The short format is described below; each field is allotted twice as many bits in the long format.

Every packet sent to a WEBRC session contains three fields of congestion control information:

- Time slot index (TSI) — The time slot index is an 8-bit number that indicates the index in $\{0, 1, \ldots, \mathtt{T}-1\}$ of the current time slot. This allows the receiver to very coarsely synchronize time within a precision of TSD seconds.
- Channel number (CN) — CN is the 8-bit index of the channel that the packet belongs to. For the base channel $\mathtt{CN} = \mathtt{T}$, and for the wave channels $\mathtt{CN} \in \{0, 1, \ldots, \mathtt{T}-1\}$. The sender knows which channels are assigned to the session and the mapping between the channels and the CNs.
- Packet sequence number (PSN) — Packets within a channel are numbered consecutively, modulo 65536, with 16-bit packet sequence numbers. For wave channels, the last packet before the quiescent period begins has $\mathtt{PSN} = 65535$; the PSN of the first packet of each wave thus depends on SR_P, BCR_P, TSD and P. A PSN of 65535 on the base channel has no special significance.

The choice of 8 bits for the TSI and CN fields limits WEBRC to 256 total channels (a base channel and 255 wave channels). Using the default value of 10 seconds for TSD and 255 time slot indices, a time slot index is reused after approximately 40 minutes. Using the default values of 300 for QD, 1 for BCR_P, 1024 for LENP_B and 0.75 for P, the maximum aggregate rate for a session is limited to $8192\,(4/3)^{255-30}$ bps, which is approximately 10^{23} Gbps, *i.e.*, much more than can be imagined with any future technology. The length of the PSN field does not limit the number of packets in a wave because PSN is allowed to wrap around within a wave. Sequence numbers will wrap around within one second if the rate of the wave is more than 500 Mbps, which means that the aggregate rate of the session is more than 2 Gbps with the default value of 0.75 for P. Long format congestion control information is recommended for sessions with extremely high aggregate rates (much greater than 2 Gbps).

C. Making waves — The WEBRC sender

Since WEBRC is completely receiver driven, the sender has no responsibilities beyond building correct packet headers and sending at the appropriate rates on each channel. First, the sender must determine the number of wave channels to produce so that the packet transmission rate—aggregate over all the channels—equals SR_P. Controlling the transmission rate is then not difficult, but requires some technique to convert the piecewise continuous rates derived from a fluid model (approximated, for example, in Fig. 1) to packet transmission schedules or sequences of interpacket intervals. Attaching the correct headers is then just a matter of keeping a counter for TSI and one PSN counter for each channel.

C.1 Number and duration of waves

N is both the number of active waves at any time and the duration of a wave in time slots. With k active waves following the exponential form shown in Fig. 1, the total rate at the end of a time slot is

$$\left(\mathtt{P}+1+\frac{1}{\mathtt{P}}+\cdots+\frac{1}{\mathtt{P}^{k-1}}\right)\mathtt{BCR\_P} = \mathtt{P}\cdot\frac{(1/\mathtt{P})^{k-1}-1}{(1/\mathtt{P})-1}\cdot\mathtt{BCR\_P}.$$

The number of active waves N should be the smallest k such that this rate is at least SR_P. This yields

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

C.2 Precise wave shapes

Using N active wave channels with the exponential form shown in Fig. 1 gives a total sender output that is periodic with period TSD and varies by a multiplicative factor P over each period. For a variety of reasons, it is desirable to have constant rate for the aggregate sender output across all channels. In particular, this reduces buffering requirements between the sender and the network and within the sender for the module that provides payload data to the transmitter.

To have constant-rate sender output, the rate at the beginning of each wave is altered to be the difference between the desired constant rate and the sum of the remaining channels. In addition, we would like for waves to start at as high a rate as possible while ensuring a positive minimum rate—which turns out to be $\mathtt{P}\cdot\mathtt{BCR}$—for the entire duration of any time slot in which the wave is active. Maximizing the rate at the beginning of the wave prevents a receiver from attempting to join a wave before the first packet is sent on the wave and minimizes the effect of interpacket spacing on the MRTT measurements made at high rates. (The minimum rate requirement is abandoned for $\mathtt{SR\_P}/\mathtt{BCR\_P} < (2-\mathtt{P}^2)/(1-\mathtt{P})$.)

The value of $\mathtt{SR\_P}/\mathtt{BCR\_P}$ determines which of four possible wave shape equations applies. The cases break down as follows:

Case 1: $\mathtt{SR\_P}/\mathtt{BCR\_P} \geq (2-\mathtt{P}^2)/(1-\mathtt{P})$
– (a) $\mathtt{SR\_P}/\mathtt{BCR\_P} \geq \mu + \mathtt{P}^2(\mathtt{P}^{-\mathtt{N}}-1)/(1-\mathtt{P})$
– (b) $\mathtt{SR\_P}/\mathtt{BCR\_P} < \mu + \mathtt{P}^2(\mathtt{P}^{-\mathtt{N}}-1)/(1-\mathtt{P})$

Case 2: $\mathtt{SR\_P}/\mathtt{BCR\_P} < (2-\mathtt{P}^2)/(1-\mathtt{P})$
– (a) $\mathtt{SR\_P}/\mathtt{BCR\_P} \geq \mathtt{P}(\mathtt{P}^{-\mathtt{N}}-1)/(1-\mathtt{P})$
– (b) $\mathtt{SR\_P}/\mathtt{BCR\_P} < \mathtt{P}(\mathtt{P}^{-\mathtt{N}}-1)/(1-\mathtt{P})$

The threshold $(2-\mathtt{P}^2)/(1-\mathtt{P})$ and the distinctions between the (a) and (b) subcases are far from self-evident; see [7] for details. Here we will mostly restrict our attention to Case 1(a).

In Case 1, the wave is designed to start at rate $\mu \cdot \mathtt{BCR\_P}$ where

$$\mu = \frac{1}{2}\left((1-\mathtt{P})\cdot\frac{\mathtt{SR\_P}}{\mathtt{BCR\_P}} + \mathtt{P}^2\right). \tag{9}$$

Each wave reaches its crest, defined as the point after which the wave is a simple exponential decay, in its second or third active time slot. Subcase (a) is when the crest is in the second active time slot.

In Case 1(a), the wave crests at rate $\mathtt{WCR}$ while there are $\mathtt{N}-2$ waves at rates $\mathtt{P}\cdot\mathtt{WCR}$, $\mathtt{P}^2\cdot\mathtt{WCR}$, …, $\mathtt{P}^{\mathtt{N}-2}\cdot\mathtt{WCR}$, the base channel at rate $\mathtt{P}^{\mathtt{N}-1}\cdot\mathtt{WCR}$, and one wave (earlier than its crest) at rate $\mu\cdot\mathtt{BCR}$. Thus,

$$\mathtt{WCR}\left(1+\mathtt{P}+\mathtt{P}^2+\cdots+\mathtt{P}^{\mathtt{N}-1}\right)+\mu\cdot\mathtt{BCR} = \mathtt{SR}$$

or

$$\mathtt{WCR} = \frac{1-\mathtt{P}}{1-\mathtt{P}^{\mathtt{N}}}(\mathtt{SR}-\mu\cdot\mathtt{BCR}). \tag{10}$$

Let t_0 be the time from the beginning of the wave to the crest. Noting that $\mathtt{N}\cdot\mathtt{TSD}-t_0$ is the time for the rate on a channel to decay from $\mathtt{WCR}$ to $\mathtt{BCR}$,

$$t_0 = \mathtt{TSD}\left(\mathtt{N}-\log_{1/\mathtt{P}}\frac{\mathtt{WCR}}{\mathtt{BCR}}\right). \tag{11}$$

This always yields $t_0 \geq \mathtt{TSD}$, and with the condition that separates Subcase (a) from Subcase (b), $t_0 \leq 2\,\mathtt{TSD}$. Now for convenience consider a wave that starts at time zero. The rate $R(t)$ on the wave channel at time $t \in [0, \mathtt{N}\cdot\mathtt{TSD}]$ is given by

$$\frac{R(t)}{\mathtt{BCR}} = \begin{cases} \mu, & 0 \leq t < t_0 - \mathtt{TSD}; \\ \mathtt{SR} - \left(1+\dfrac{\mathtt{P}^{-(\mathtt{N}-1)}-1}{1-\mathtt{P}}\right)\mathtt{P}^{t/\mathtt{TSD}}, & \\ & t_0-\mathtt{TSD} \leq t < \mathtt{TSD}; \\ \mathtt{SR} - \left(\mu+\dfrac{\mathtt{P}^{-(\mathtt{N}-1)}-1}{1-\mathtt{P}}\cdot\mathtt{P}^{t/\mathtt{TSD}}\right), & \\ & \mathtt{TSD} \leq t < t_0; \\ \mathtt{P}^{t/\mathtt{TSD}-\mathtt{N}}, & t_0 \leq t \leq \mathtt{N}\cdot\mathtt{TSD}. \end{cases} \tag{12}$$

The value of μ has been chosen to balance two competing goals: to make waves start at a high rate and to make the waves follow a simple exponential decay as much as possible. The chosen μ makes the wave an exponential decay for at least the last $\mathtt{N}-3$ time slots. The rate for $t \in [t_0-\mathtt{TSD}, t_0)$ has an awkward form and no rationale beyond making the rates of all the channels add to the constant $\mathtt{SR}$. Fig. 14 shows some examples of waves generated with Eq. (12), and Fig. 15 shows how stacking the waves makes the cumulative rate constant.

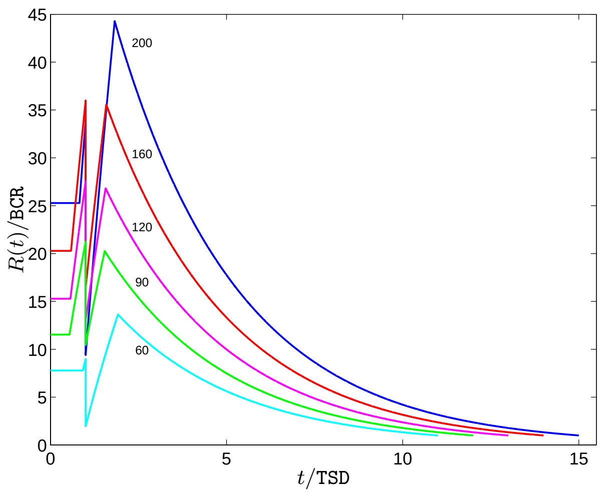


Fig. 14. Waves designed with Eq. (12). The curve labels are values of $\mathtt{SR}/\mathtt{BCR}$, and $\mathtt{P} = 0.75$.

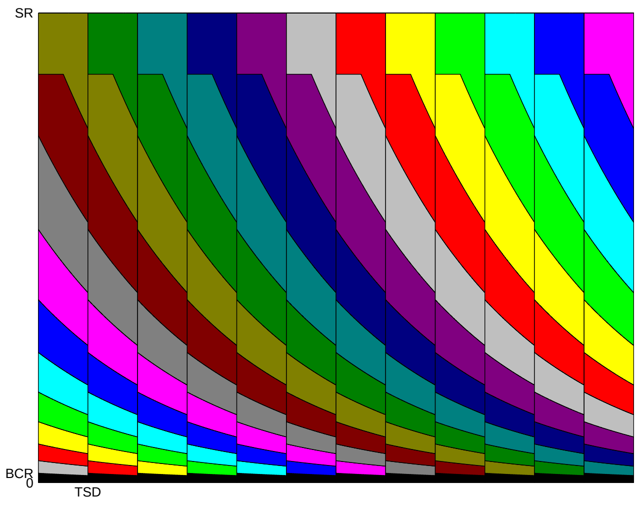


Fig. 15. Server output totalled over all channels is constant. Here $\mathtt{P} = 0.75$ and $\mathtt{SR}/\mathtt{BCR} = 50$ so $\mathtt{N} = 10$. Black represents the base channel and the $\mathtt{T} = 14$ other colors represent wave channels. (For this example, $\mathtt{Q} = \mathtt{T}-\mathtt{N} = 4$; $\mathtt{Q}$ is usually larger in practice.)

C.3 Packet scheduling

Since a WEBRC sender has constant total rate, it sends packets evenly spaced by $1/\mathtt{SR\_P}$ seconds. The challenge is to assign channel numbers to these packets so that the rates on the base channel and wave channels approximately follow the desired fluid-model equations. The suggested implementation—given in the accompanying *ns* code—is to compute a sequence of channel numbers to cover one time slot and to then use this sequence for all time slots with an appropriate cyclic shifting of channel numbers for the wave channels. The sequence is computed by first determining idealized packet transmission times based on the

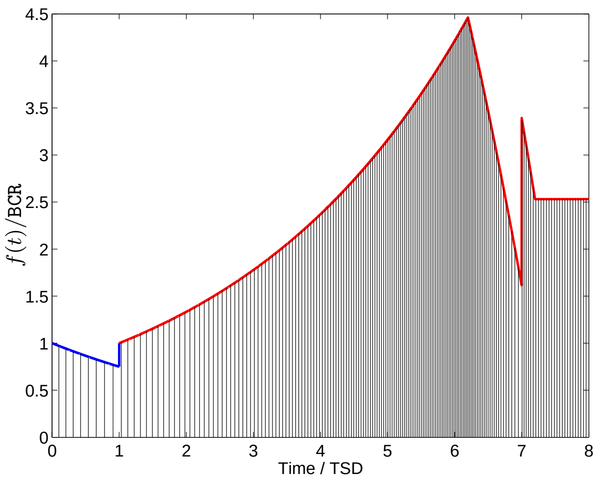


Fig. 16. Graphical depiction of the computation of packet transmission times from fluid-model rates. The packet transmission times divide the area under $f(t)$ into $\texttt{SR\_P} \cdot \texttt{TSD}$ unit-area regions. In this example $\texttt{P} = 0.75$, $\texttt{BCR\_P} = 1$ and $\texttt{SR\_P} = 18$.

fluid model and then sorting (channel number, transmission time) pairs by the transmission times modulo TSD.

Packet transmissions are discrete events, so there is some ambiguity in converting a transmission rate with temporal variation into a sequence of transmission times. To be faithful to a transmission rate of $R(t)$ packets per second, the transmission times should be selected so that the number of transmissions in an interval $[a, b)$ is approximately $\int_a^b R(\tau)\, d\tau$. Of course, this cannot hold for all intervals, but the relative error should decrease as $b - a$ increases.

The suggested mechanism for converting fluid-model rates to transmission times is based on the observation that the number $\texttt{SR\_P} \cdot \texttt{TSD}$ of packets sent in one time slot is also the number of packets in one wave plus one period of the base channel. Form a function $f(t)$, $t \in [0, (\texttt{N}+1)\texttt{TSD}]$, by taking one period of the base channel followed by one time-reversed wave. This function is strictly positive on $[0, (\texttt{N}+1)\texttt{TSD}]$ and integrates over this interval to $\texttt{SR\_P} \cdot \texttt{TSD}$. Choosing times at which to transmit packets is then equivalent to dividing the area under $f(t)$ into $\texttt{SR\_P} \cdot \texttt{TSD}$ regions of unit area. The left edge of each of these regions is taken as a packet transmission time. Thus, the first packet sent in a time slot is a base channel packet and the first packet of any wave is sent $(\mu \cdot \texttt{BCR\_P})^{-1}$ seconds after the beginning of its first active time slot. An example of this graphical construction is given in Fig. 16. The algebraic equivalent is to solve

$$\int_0^{\tau_k} f(t)\, dt = k, \qquad k = 0,\, 1,\, 2,\, \ldots,\, \texttt{SR\_P} \cdot \texttt{TSD} - 1.$$

Details are given in [7], along with MATLAB code that covers all four wave shape cases. The transmissions on all the channels for a set of consecutive time slots is depicted in Fig. 17.

C.4 Normal sender operation

When the sender is initialized, it determines a sequence of channel numbers $\{c_k\}_{k=1}^{\texttt{SR\_P} \cdot \texttt{TSD}}$ to use in time slot 0. In the process of determining this sequence, the sender also determines the PSNs for the wave channel packets. The sender operation is then very simple.

The sender maintains counters for the time slot index TSI and for the PSN on the base channel. Let i denote the current time slot index. Every packet has the time slot index in the header as described in Section VI-B. When c_k arises from the list of channel numbers, it can correspond to the base channel T or to any of the wave channels. If $c_k = \texttt{T}$, the sender sends a packet on the base channel with PSN value determined by the PSN counter and then increments the PSN counter for the base channel. Otherwise, the sender sends a packet on channel $(c_k + i)_{\texttt{T}}$ with the PSN value taken from the precomputed list.[11] The sender goes through its list of $\texttt{SR\_P} \cdot \texttt{TSD}$ c_ks once for each time slot. At the end of each time slot, it increments the time slot index modulo T.

D. Catching waves — The WEBRC receiver

The bulk of the complexity in WEBRC is in the receiver operation. The receiver is responsible for determining an appropriate target reception rate and for timing its joins of wave channels to make the actual reception rate match the target.

In normal operation, the target reception rate TRATE_P is calculated from the receiver's own measurements of network conditions. The measurements of loss event rate and MRTT are called LOSSP and ARTT, respectively, and play the roles of p and t_{RTT} in Eq. (1) to give TRATE_P. A different computation for TRATE_P holds during a start-up period at the beginning of session when the receiver does not yet have meaningful measurements of network conditions.

Computations of TRATE_P occur at the boundaries of *epochs*, which each have duration EL seconds. The epoch length is a small fraction of the time slot duration so that there is fine granularity in where waves may be joined. A standard setting for $\texttt{TSD} = 10$ is $\texttt{EL} = 0.5$. This choice allows fine-grained control of the reception rate up to a precision of approximately 1% with the default setting of 0.75 for P.

Upon determining TRATE_P, the receiver has two choices: to join the next wave channel right away or not. Joining an $(i + 1)$st wave channel when already subscribed to i wave channels increases the reception rate by a multiplicative factor of

$$\Gamma_i = \frac{(1/\texttt{P})^{i+2} - 1}{(1/\texttt{P})^{i+1} - 1}. \tag{13}$$

Notice that there is no dependence on the time within the time slot. The receiver maintains an estimate of its anticipated reception rate ARR_P and joins if ARR_P increased by this multiplicative factor is less than or equal to TRATE_P.

Fig. 18(a) illustrates how the exponentially decaying transmission rates and the rule for joining combine to give

[11]The notation $(n)_m$ denotes the integer in $\{0,\, 1,\, \ldots,\, m-1\}$ that is congruent to n modulo m.

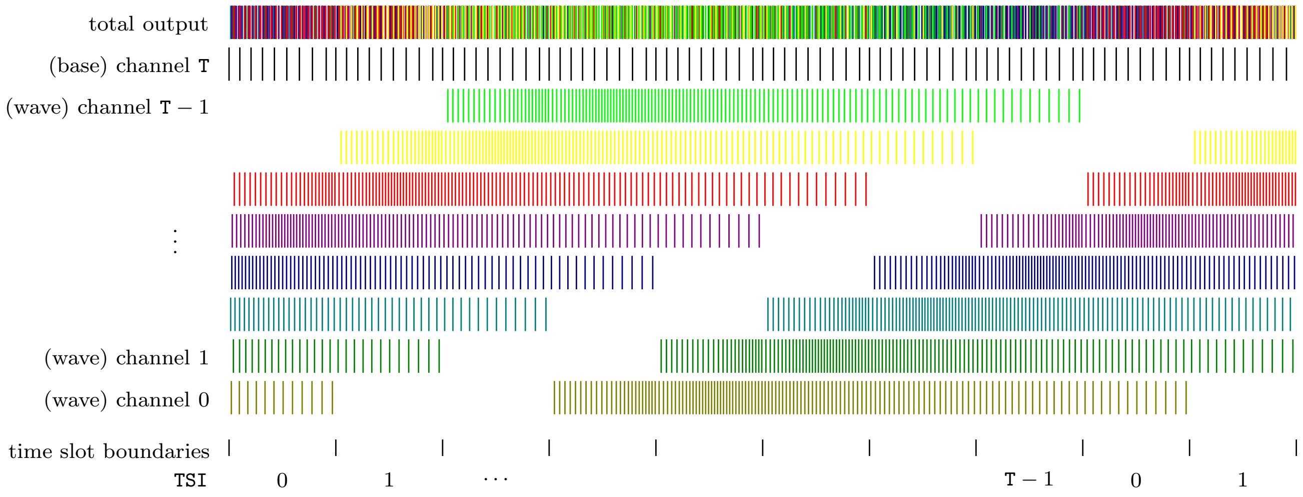


Fig. 17. Idealized packet transmission times for a session with BCR_P = 1, SR_P = 14, P = 0.75, Q = 2, and TSD = 10. (Larger ratios SR_P/BCR_P are of more interest, and in practice Q is much larger.) Notice that waves start at time slot boundaries and reach their crests in their second active time slot. (In Case 1(b), the crest is in the third active time slot.) Taking all the channels together, the sender transmits at a constant rate.

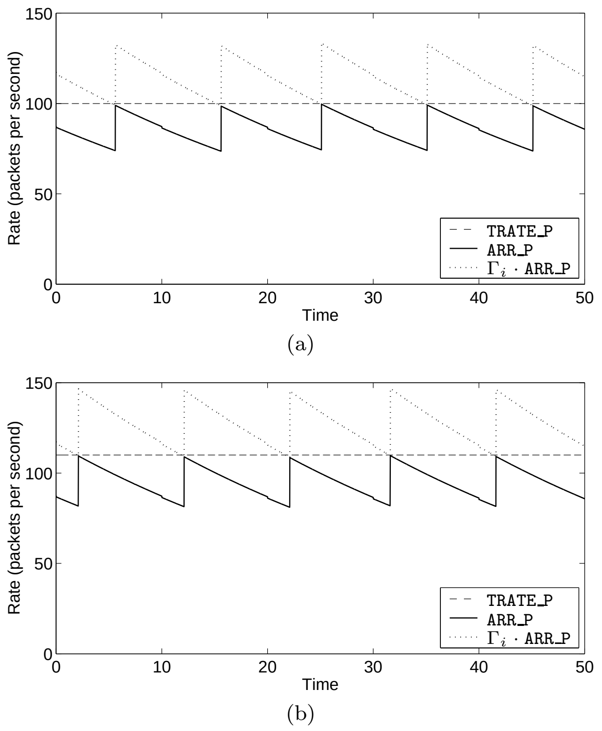


Fig. 18. Depictions of target rate and anticipated reception rate with TSD = 10 and P = 0.75. (a) Joins are issued when the increased anticipated reception rate is at most equal to the target rate. (b) The timing of joins determines the average throughput.

Description	Name	Value
Multicast round trip time:		
Time of join on channel i	$\mathtt{JOIN}_i$	
Time of first packet on channel i	$\mathtt{FIRST}_i$	
Single MRTT measurement	MRTT	Eq. (14)
Running $(\mathtt{MRTT})^2$ average	V	
Number of MRTT measurements	K	
Averaging weight	α	$\in [0.1, 0.25]$
EWMA-like weight for ARTT and V	ω	Eq. (15)
Normalized version of ω	ω'	Eq. (16)
Loss event rate:		
Packets since last loss	W	
Packets in short-term history	X	
Loss events in short-term history	Y	
Long-term reciprocal loss event rate	Z	
1/(loss event rate)	$\mathtt{Z}_1$	Eq. (20)
1/(loss event rate) (next packet lost)	$\mathtt{Z}_2$	Eq. (21)
EWMA weight for loss intervals	δ	0.2
Short- to long-term transfer per TSD	ν	0.3
Rate averages:		
EWMA weight for TRR_P (normal)	ζ_{normal}	Eq. (26)
EWMA weight for TRR_P (start-up)	$\zeta_{\text{start-up}}$	Eq. (27)
EWMA weight for ARR_P (normal)	β_{normal}	Eq. (28)
EWMA weight for ARR_P (start-up)	$\beta_{\text{start-up}}$	0
Start-up:		
Max MRTT increase to stay in start-up		Eq. (30)
Min TRR_P to stay in start-up		Eq. (32)

TABLE III
RECEIVER DETAILS (SUPPLEMENT TO TABLE I)

a reception rate that is approximately periodic with period TSD. For simplicity, TRATE_P is shown as a constant and ARR_P is idealized. A receiver that joins the wave channels with the same frequency (once per TSD) but joins waves earlier has a higher reception rate, as shown in Fig. 18(b).

This brief discussion omits many details that are specified later in this section. Since the intuitive meanings of LOSSP, ARTT, and ARR_P are sufficient to understand the receiver operation, their full descriptions are deferred to Section VI-D.3. First, the receiver state and the various actions and decisions by the receiver are described. It may be useful to refer to Tables I and III to keep track of notation.

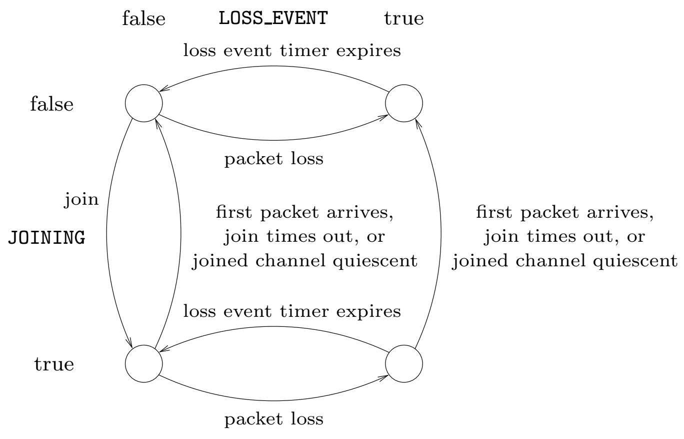

false
LOSS_EVENT
true
loss event timer expires
false
packet loss
join
JOINING
first packet arrives, join times out, or joined channel quiescent
first packet arrives, join times out, or joined channel quiescent
loss event timer expires
true
packet loss

Fig. 19. State diagram for WEBRC receiver. Wave channels are joined only when JOINING and LOSS_EVENT are false. Loss events are initiated by packet losses and ended when the loss event timer (with duration ARTT) expires.

D.1 Main portions of receiver state

The receiver state includes two Boolean variables. LOSS_EVENT is set when a lost packet is revealed by a missing PSN and is then cleared after ARTT seconds. This duration of a loss event is consistent with TFRC. JOINING is set when a wave channel is joined and is then cleared when the first packet on that wave arrives or the join timer (set when the wave channel is joined) expires. The receiver does not join wave channels when either LOSS_EVENT or JOINING is true, regardless of the target and measured reception rates. Fig. 19 gives a state diagram.

The receiver updates several variables as packets arrive or at epoch boundaries. These include a loss event rate LOSSP, average multicast round trip time ARTT, slow start rate SSR_P, and a few variants of the reception rate. Within each epoch, received packets are counted to give the reception rate RR_P; received and lost packets, as indicated by missing PSNs, are counted to give the "ideal" reception rate IRR_P. These raw measurements are averaged over time to give the "true" reception rate TRR_P and "anticipated" reception rate ARR_P, respectively.

Along with JOINING, a few other variables are associated with joining wave channels. When wave channel i is joined, the time of the join $\mathtt{JOIN}_i$ is recorded. This is used in updating ARTT when the first packet arrives on channel i. NWC is the number of active wave channels that the receiver is currently subscribed to. Because the subscribed channels are numbered consecutively and the time slot indices are coordinated with the wave channel numbers, NWC and TSI suffice to determine the full set of subscribed wave channels.

To detect packet losses, the receiver tracks as $\mathtt{SEQNO}_i$ the PSN of the last packet received on each wave channel i. When wave channel i is left, $\mathtt{SEQNO}_i$ is set to a "false" value; thus $\mathtt{SEQNO}_i$ reveals whether the first packet on wave channel i has arrived. Other state information is described in conjunction with the LOSSP calculations in Section VI-D.3.

D.2 Receiver actions and events

There are various receiver events, some of which are triggered by the passing of time on the receiver and others by the reception of packets.

Entering a session When a receiver first joins a session, it immediately joins the base channel, clears LOSS_EVENT, sets JOINING, and assigns $\mathtt{NWC} = 0$. At the same time, it starts a timer that indicates an epoch boundary every EL seconds. Most of the computations made by the receiver occur at epoch boundaries, although a few occur with every received packet.

The TSI field in the first base channel packet allows the receiver to synchronize itself, as much as necessary, with the session: the TSI field indicates the time slot, and the time within the time slot is not tracked on an ongoing basis. However, the time between the first packet arrival and the first time slot boundary is used to refine the anticipated reception rate ARR_P (see Section VI-D.3). The CN field of a base channel packet equals T, so the first packet also indicates the number of wave channels.

Packet reception The header fields of every received packet are examined. The TSI field is used to determine if the time slot has changed. A TSI change from i to $(i+1)_\mathtt{T}$ indicates that wave channel i has just become quiescent. The receiver thus decrements NWC by one, leaves wave channel i, and sets $\mathtt{SEQNO}_i$ to "false." If the first packet on this wave channel had not arrived, JOINING is cleared. The time slot change also prompts adjustments to ARR_P described in the next section.

The (CN, PSN) pair is used to determine if there has been packet loss. Without packet loss, PSN should be one greater (modulo 65536) than $\mathtt{SEQNO}_\mathtt{CN}$. When this condition does not hold, at least one packet has been lost; if LOSS_EVENT is false, it is the beginning of a new loss event. At the beginning of a loss event, LOSS_EVENT is set to true, a loss event timer is set for ARTT seconds, and the slow start rate is updated to $\mathtt{SSR\_P} = \max\{\mathtt{P} \cdot \mathtt{TRR\_P}, \mathtt{SSMINR\_P}\}$. SSMINR_P is the minimum allowed slow start threshold rate and is set to $\Gamma_0 \cdot \Gamma_1 \cdot \mathtt{BCR\_P}$. Whether or not a packet has been lost, $\mathtt{SEQNO}_\mathtt{CN}$ is updated with each received packet, and the reception rate counters and variables associated with LOSSP are incremented as described in Section VI-D.3.

The first packet of a wave prompts some additional actions. A first packet—indicated by $\mathtt{SEQNO}_\mathtt{CN}$ = "false"—requires $\mathtt{FIRST}_\mathtt{CN}$ to be set to the current time, JOINING to be cleared, and the join timer to be cancelled. Also, ARTT is updated as described in Section VI-D.3.

Epoch boundary Most computations happen at epoch boundaries and it is at epoch boundaries that wave channels may be joined. The receiver first updates the reception rate variables and calculates LOSSP as described in Section VI-D.3. Then, if LOSS_EVENT and JOINING are false, the receiver calculates TRATE_P and possibly joins the next wave channel.

For fairness with TCP, in the spirit of TFRC, the target rate is fundamentally given by an analogue of Eq. (1):

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

The actual target rate uses a few adjustments to REQN. First, the receiver uses the slow start concept to adjust the target rate upward to SSR_P, if applicable. The idea behind this adjustment is that if the receiver had a true reception rate R at the beginning of the most recent loss event, it should be allowed to subscribe up to rate $\mathtt{P}\cdot R$ even if REQN is not this high. (Note that ARR_P has to drop to $\mathtt{P}\cdot R/\Gamma_{\mathtt{NWC}} \approx \mathtt{P}^2 \cdot R$ for a join to be allowed.) The target rate is further adjusted to not exceed the maximum reception rate MRR_P. The target rate is thus normally (*i.e.*, when the receiver is not in start-up mode) given by

$$\mathtt{TRATE\_P} = \min\left\{\max\{\mathtt{SSR\_P},\, \mathtt{REQN\_P}\},\, \mathtt{MRR\_P}\right\}. \quad (5)$$

Whether or not the target rate justifies joining the next wave channel depends on the current reception rate ARR_P and (mildly) on the number of subscribed wave channels. When subscribed to NWC wave channels, joining the next wave channel increases the reception rate by a factor $\Gamma_{\mathtt{NWC}}$. The next wave channel is thus joined if $\Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P} \leq \mathtt{TRATE\_P}$.

At the beginning of session, the receiver does not have enough information to compute a meaningful value of REQN. This is called the start-up period and is indicated by SSR_P equally its initial value of infinity. In start-up mode, the target rate is

$$\mathtt{TRATE\_P} = \min\left\{4 \cdot \mathtt{TRR\_P},\, \mathtt{MRR\_P}\right\}. \quad (7)$$

Some additional rules for joining and for updating SSR_P apply in start-up mode. These are detailed in Section VI-D.4.

Joining If the decision based on the target rate is to join, the receiver joins the next wave channel, which is channel $i = (\mathtt{TSI} + \mathtt{NWC})_{\mathtt{T}}$. In addition to sending the join message, the receiver sets JOINING, increases ARR_P multiplicatively by $\Gamma_{\mathtt{NWC}}$, increments NWC, and records the current time in $\mathtt{JOIN}_i$ for subsequent ARTT calculations.

Since joins can only happen when JOINING is cleared, the receiver joins wave channels at most once every MRTT, and in fact slower because joins are at epoch boundaries. (In start-up mode, joins are restricted further.) The rate increase factor is $\Gamma_i \approx 1/\mathtt{P}$, and $\mathtt{P} > 1/2$ is recommended, so WEBRC slow start behaves more conservatively than TCP slow start. As noted in [28], this is desirable for a multicast protocol.

For robustness to lost join messages, the receiver uses a *join timer* that is set when the join message is sent. The purpose of the join timer is to fix a time at which to give up on receiving packets from the last wave channel and assume the join message was lost. A large timer setting makes the receiver sluggish in its response to lost joins. On the other hand, a small value risks canceling joins that otherwise would have been successful, thus lowering the reception rate and creating extra IGMP and PIM SM traffic.

An appropriate value for the timer seems hard to determine from *ns* simulations alone and our implementation experience with multicast is currently quite limited. In particular, we have little evidence of the likelihood of lost join messages, and the ideal timer duration depends on the variation of MRTT in a session. Assuming no changes to MRTT and no jitter, the first packet should arrive after at most

$$\mathtt{ARTT} + \frac{1}{\mathtt{P}\cdot\mathtt{BCR\_P}} \text{ seconds},$$

where the second term arises from a bound on the interpacket time on any wave channel that is conservative and would be closely approached only for certain values of $(\mathtt{SR\_P}, \mathtt{BCR\_P}, \mathtt{TSD}, \mathtt{P})$ [7]. Our recommendation for the timer setting is

$$10\,\mathtt{ARTT} + 2\,\frac{1}{\mathtt{BCR\_P}}.$$

The factor of 10 seems suitably conservative since a lost join should inhibit the generation of more joins.

Join timeout If the join timer expires, the receiver leaves the pending channel, decrements NWC, reduces ARR_P by dividing it by $\Gamma_{\mathtt{NWC}}$, and clears JOINING. In addition, there should be some mechanism to retard the target rate and use a larger timer value for the next join. (Recall that the next join will be for the same wave channel since the receiver always subscribes to consecutive wave channels.) We have not tested any such mechanism because the *ns* implementation of multicast never loses join messages. One possibility would be to increase ARTT, for example by a factor of 2. This would generally halve the target rate and would approximately double the duration of the join timer. It is possible that the drastic effect on the reception rate would be too prolonged; further study with multicast implementations is needed.

Catastrophic failure The receiver should have mechanisms to detect particularly aberrant network behavior. This could include no packets arriving for a long time and repeated failures to join wave channels. This requires further investigation.

D.3 Receiver measurements

The receiver makes measurements of loss event rate LOSSP and average multicast RTT ARTT for use in the TFRC equation (6). It also tracks reception rates for facilitating decisions on joining.

Average multicast round trip time The receiver makes a measurement of MRTT when the base channel is joined and each time a wave channel is joined and at least one packet arrives. The simplest raw MRTT measurement is the time elapsed between the issuance of the join request and the arrival of the first packet of the wave. This measurement inflates the MRTT value because it includes the time between the successful completion of the join and the next packet transmission time on the wave. Also, independent of this bias, raw MRTT measurements may vary greatly even in a static network with a single receiver in a single WEBRC session. The receiver thus computes a value MRTT with reduced bias and uses a variance-adjusted form of an exponentially weighted moving average (EWMA), subject to a maximum decline based on a single measurement, to compute $\mathtt{ARTT}$.

$\mathtt{MRTT}$ is computed when the first packet of a wave arrives. When wave channel i is joined, the calculation uses the time when the join was sent $\mathtt{JOIN}_i$ and the time when the first packet arrived $\mathtt{FIRST}_i$. If the transmission rate on the wave channel were constant with interpacket interval t, the inflation of MRTT discussed above could be treated as a random variable uniformly distributed on $[0, t]$. To eliminate the bias, we could use

$$\mathtt{MRTT} = (\mathtt{FIRST}_i - \mathtt{JOIN}_i) - \frac{1}{2}t.$$

(Note that this value can be negative.)

In earlier versions of WEBRC [14], [12], [13], t was replaced by the interval between the first and second wave packet arrivals, adjusted for lost packets if necessary. This was attractive for being an actual measurement of interpacket spacing as seen with current network conditions. The main disadvantage was that the $\mathtt{MRTT}$ measurement was not available until the second packet arrived. We now advocate using the average interpacket spacing given the current $\mathtt{NWC}$, based on the fluid model underlying the wave packet transmission times, since this can be computed immediately when the first wave packet arrives. This yields

$$\mathtt{MRTT} = (\mathtt{FIRST}_i - \mathtt{JOIN}_i) - \frac{\log(1/\mathtt{P})}{2(1-\mathtt{P})} \cdot \mathtt{P}^{\mathtt{NWC}} \cdot \frac{1}{\mathtt{BCR\_P}}. \quad (14)$$

Recall from Section IV that the $\mathtt{MRTT}$ measurement made by one receiver depends not only on queueing delays but also on the join times of other receivers in the session. For this reason, even without queueing variations there can be very high variance in successive MRTT measurements and the variance depends on the number of receivers in the session. Since the number of receivers is unknown to each receiver, and furthermore dynamic and unknown to the sender, the averaging of $\mathtt{MRTT}$ measurements is based on online variance estimation. The averaging of $\mathtt{MRTT}$ measurements can be regarded as EWMA with a variance-adjusted weight on the current measurement. This is discussed in further generality in the Appendix.

The first $\mathtt{MRTT}$ is obtained when the base channel is joined. An exception to the use of Eq. (14) is made for this case in that

$$\mathtt{MRTT} = \mathtt{FIRST_T} - \mathtt{JOIN_T}$$

is calculated. Neglecting the adjustment for interpacket spacing on the base channel ensures that $\mathtt{MRTT}$ is positive. The resulting positive bias in $\mathtt{ARTT}$ would lower the target rate, but $\mathtt{REQN}$ is not relevant in start-up mode and the effect of this bias is generally small by the time the receiver leaves start-up mode. At this time, $\mathtt{ARTT}$ is initialized to $\mathtt{MRTT}$ and an auxiliary variable $\mathtt{V}$, representing the average of $(\mathtt{MRTT})^2$, is initialized to $(\mathtt{MRTT})^2$. Also, $\mathtt{K}$ is set to 1.

With each subsequent $\mathtt{MRTT}$ measurement, $\mathtt{ARTT}$, $\mathtt{V}$ and $\mathtt{K}$ are updated as follows:

$$\begin{aligned} \omega &= \min\left\{\alpha \cdot \frac{(\mathtt{ARTT})^2}{\mathtt{V}},\, 1\right\} && (15) \\ \mathtt{K} &= \mathtt{K} + 1 && \\ \omega' &= \frac{\omega}{1-(1-\omega)^{\mathtt{K}}} && (16) \\ \mathtt{V} &= (1-\omega') \cdot \mathtt{V} + \omega' \cdot (\mathtt{MRTT})^2 && (17) \\ \mathtt{MRTT_{ave}} &= (1-\omega') \cdot \mathtt{ARTT} + \omega' \cdot \mathtt{MRTT} && (18) \\ \mathtt{ARTT} &= \max\{\mathtt{MRTT_{ave}},\, \mathtt{P} \cdot \mathtt{ARTT}\} && (19) \end{aligned}$$

In this calculation, ω is like an EWMA weight for both $\mathtt{V}$ and $\mathtt{ARTT}$.[12] Note that higher variance of $\mathtt{MRTT}$ leads to smaller average values of ω. The adjustment to ω made in Eq. (16) is a renormalization of EWMA that prevents excessive weight on the initial condition; this adjustment and the state variable $\mathtt{K}$ can perhaps be removed. Eqs. (17) and (18) are EWMA computations with the weight ω'. Eq. (19) is used to prevent $\mathtt{ARTT}$ from dropping by more than a factor of $\mathtt{P}$ with a single join. Large drops of $\mathtt{ARTT}$ create large increases in the target rate and hence may lead to oversubscription. A drop by a factor of $\mathtt{P}$ increases the target rate by a factor $1/\mathtt{P}$, which is approximately enough to allow the receiver to join another wave. Thus, roughly speaking, if the true network MRTT has dropped precipitously, the receiver will be allowed to join one wave per MRTT (rounded up to the nearest multiple of $\mathtt{EL}$).

The choice of α trades off the smoothness of the target rate against the reactivity to network changes that are reflected in MRTT measurements. The recommended range is $\alpha \in [0.1, 0.25]$.

<u>*Loss event rate*</u> WEBRC uses the concepts of loss event and loss event rate from TFRC [6] with little modification. In TFRC, *loss events* are nonoverlapping periods of duration equal to the round trip time that contain all of the packet losses; a loss event is triggered by a packet loss and then lasts for a period of RTT regardless of additional losses in that time. The only difference in WEBRC is that the duration of a loss event is the current $\mathtt{ARTT}$. The number of packets between the beginnings of loss events is called the *loss interval* and the *loss event rate* is the reciprocal of the average loss interval.

In TFRC, the receiver uses the last 8 loss intervals to compute a local loss event rate. The loss intervals $\{s_i\}_{i=1}^{8}$, with s_1 being the most recent, are averaged as

$$\hat{s} = \frac{\sum_{i=1}^{8} w_i^{\mathrm{TFRC}} s_i}{\sum_{i=1}^{8} w_i^{\mathrm{TFRC}}}$$

with weight vector $w^{\mathrm{TFRC}} = (1, 1, 1, 1, \frac{4}{5}, \frac{3}{5}, \frac{2}{5}, \frac{1}{5})$. (The "history discounting" mechanism is omitted here.) The average $\hat{s}$ is influenced only by packets received up to the last loss indication. A large number s_0 of packet receptions since the last loss should decrease the loss event rate estimate. Thus,

$$\hat{s}' = \frac{\sum_{i=1}^{8} w_i^{\mathrm{TFRC}} s_{i-1}}{\sum_{i=1}^{8} w_i^{\mathrm{TFRC}}}$$

[12]It is not really exponential weighting because ω is not constant.

is also computed and the loss event rate is

$$\hat{p} = \frac{1}{\max\{\hat{s},\, \hat{s}'\}}.$$

Roughly, if losing the next packet would increase the average loss interval, we act as if that packet is lost.

An important feature of using a fixed number of loss intervals is that the amount of data collected (packets lost or received) is automatically inversely proportional to the loss probability and hence appropriate for the estimation of the loss probability. In WEBRC the TSD-periodicity of rates tends to make losses somewhat bursty and TSD-periodic. Thus it is important for the loss averaging to use an amount of data that, in addition to being a number of packets inversely proportional to the loss probability, extends in time to a few time slots. To give more weight to recent loss events than to those further in the past, WEBRC uses an EWMA computation on loss intervals. Also, to make sure that LOSSP is not too local in time and not overly affected by the burstiness of losses, WEBRC artificially spreads the losses in its short-term history evenly over the number of packets in its short-term history. The short-term history is slowly averaged into the long-term history at each epoch boundary.

The LOSSP computation uses two weights: δ is an EWMA weight for the loss events and ν is the fraction of the short-term history that is averaged into the long-term history per time slot. The short-term history is a pair $(\mathtt{X}, \mathtt{Y})$ which represents a number of packets and a number of loss events, respectively. The long-term history is a number $\mathtt{Z}$ that represents an average loss interval. The short- and long-term history are combined to give an average loss interval by imagining that $\mathtt{Y}$ loss intervals of length $\mathtt{X}/\mathtt{Y}$ are observed starting from an initial average loss interval of $\mathtt{Z}$. Writing out the expressions for EWMA filtering, for $\mathtt{Y} \in \mathbb{Z}^+$ we obtain

$$\begin{aligned} \mathtt{Z}_1 &= (1-\delta)^{\mathtt{Y}}\mathtt{Z} + \sum_{i=0}^{y-1}\left[\delta(1-\delta)^i \frac{\mathtt{X}}{\mathtt{Y}}\right] \\ &= (1-\delta)^{\mathtt{Y}}\mathtt{Z} + \left(1-(1-\delta)^{\mathtt{Y}}\right)\frac{\mathtt{X}}{\mathtt{Y}} \qquad (20) \end{aligned}$$

as an average loss interval combining short- and long-term history. The number of packets received since the beginning of the last loss event is denoted by W. Just as s_0 in TFRC influences $\hat{p}$ only when s_0 is large, WEBRC defines

$$\mathtt{Z}_2 = (1-\delta)^{\mathtt{Y}+1}\mathtt{Z} + \left(1-(1-\delta)^{\mathtt{Y}+1}\right)\frac{\mathtt{X}+\mathtt{W}+1}{\mathtt{Y}+1} \qquad (21)$$

and then finally defines

$$\mathtt{LOSSP} = \frac{1}{\max\{\mathtt{Z}_1,\, \mathtt{Z}_2,\, 1\}} \qquad (22)$$

so that W influences LOSSP only when W is large.

The short- and long-term loss history are updated as follows. The short-term history (X,Y) is an active count of packets and loss events; at the beginning of each loss event, X is incremented by the number of packets since the beginning of the previous loss event and Y is incremented by one. The long-term history absorbs fraction $\nu \cdot \mathtt{EL}/\mathtt{TSD}$ of the short-term history at each epoch boundary. This means that Z is updated consistent with $(\nu \cdot \mathtt{EL}/\mathtt{TSD}) \cdot \mathtt{Y}$ loss intervals of length $(\nu \cdot \mathtt{EL}/\mathtt{TSD}) \cdot \mathtt{X}$ to give

$$\mathtt{Z} = (1-\delta)^{(\nu\cdot\mathtt{EL}/\mathtt{TSD})\cdot\mathtt{Y}}\mathtt{Z} + \left(1-(1-\delta)^{(\nu\cdot\mathtt{EL}/\mathtt{TSD})\cdot\mathtt{Y}}\right)\frac{\mathtt{X}}{\mathtt{Y}}. \qquad (23)$$

The portion of the loss history averaged into Z then has to be removed from the short-term history:

$$\begin{aligned} \mathtt{X} &= \left(1-\nu\cdot\frac{\mathtt{EL}}{\mathtt{TSD}}\right)\mathtt{X} \qquad (24) \\ \mathtt{Y} &= \left(1-\nu\cdot\frac{\mathtt{EL}}{\mathtt{TSD}}\right)\mathtt{Y} \qquad (25) \end{aligned}$$

Note that although X and Y are conceptually counts of events, they are not limited to integer values. Furthermore, though the derivation of Eq. (20) requires Y to be a positive integer, the use of Eqs. (20) and (21) is extended to all positive real values of Y.

In summary, maintaining and using the loss event history requires the following actions.
• Initialization: To start at $\mathtt{LOSSP} = p$, set $\mathtt{W} = 0$, $\mathtt{X} = 0$, $\mathtt{Y} = 0$, and $\mathtt{Z} = 1/p$. This initialization happens when the receiver leaves start-up mode and possibly at other times, as described in Section VI-D.4.
• With each packet reception: Increment W by one. If the PSN of the packet indicates one or more packet losses, increment W for the lost packets also.
• At the beginning of each loss event: Set $\mathtt{X} = \mathtt{X} + \mathtt{W}$, $\mathtt{Y} = \mathtt{Y} + 1$, and $\mathtt{W} = 0$.
• At each epoch boundary (except in start-up mode): Evaluate Eqs. (23)–(25).
• To compute LOSSP: Evaluate Eqs. (20)–(22).
The recommended values for δ and ν are 0.2 and 0.3, respectively.

The reception rates There are two time-averaged reception rates maintained by the receiver: the "true" reception rate TRR_P and the "anticipated" reception rate ARR_P. They are used for different purposes and thus are based on different raw data and different averaging.

TRR_P is used to determine the target rate in start-up mode (see Eq. (7)) and to detect if start-up mode should end (see Section VI-D.4). In normal operation, the slow start rate is reset in proportion to TRR_P at the beginning of each loss event. For all of these uses, it is desirable for TRR_P to be a fairly smooth version of the reception rate RR_P and an EWMA with weight $\zeta \in (0, 1)$ is used to compute it:

$$\mathtt{TRR\_P} = (1-\zeta)\cdot\mathtt{TRR\_P} + \zeta\cdot\mathtt{RR\_P}.$$

Clearly, smaller values of ζ give smoother estimates TRR_P. For normal operation, the (heuristic) suggested value for ζ is

$$\zeta_{\mathrm{normal}} = \frac{2\cdot\mathtt{EL}}{4+\mathtt{TSD}}. \qquad (26)$$

Being proportional to $\mathtt{EL/TSD}$ is consistent with the fact that $\mathtt{RR\_P}$, as a discrete-time sequence, has a strong periodic component with period $\mathtt{TSD/EL}$. The constant 4 in the denominator is most significant when $\mathtt{TSD}$ and, presumably, $\mathtt{EL}$ are small. It is included to counteract the volatility in $\mathtt{RR\_P}$ when $\mathtt{EL}$ is small that is due to $\mathtt{EL}\cdot\mathtt{RR\_P}$ being an integer.

The value ζ_{normal} is too small for start-up mode. In start-up mode, $\mathtt{RR\_P}$ should be increasing quickly and EWMA with a small value of ζ will cause $\mathtt{TRR\_P}$ to lag significantly behind $\mathtt{RR\_P}$. For the presumably most likely case where the MRTT is less than $\mathtt{EL}$, the receiver will generally join waves at alternate epoch boundaries during start-up. Thus, $\mathtt{RR\_P}[k] \approx \mathtt{P}^{-k/2}\cdot\mathtt{BCR\_P}$ is the rate at the kth epoch boundary after the first base channel packet is received. EWMA filtering of $\mathtt{RR\_P}[k]$ yields

$$\begin{aligned}
\mathtt{TRR\_P}[k] &= \zeta\sum_{i=0}^{k}(1-\zeta)^{k-i}\cdot\mathtt{RR\_P}[i]\\
&\approx \zeta\sum_{i=0}^{k}(1-\zeta)^{k-i}\cdot\mathtt{P}^{-i/2}\cdot\mathtt{BCR\_P}\\
&= \zeta(1-\zeta)^{k}\cdot\mathtt{BCR\_P}\cdot\sum_{i=0}^{k}\left[(1-\zeta)\sqrt{\mathtt{P}}\right]^{-i}\\
&\approx \zeta(1-\zeta)^{k}\cdot\mathtt{BCR\_P}\cdot\frac{\left[(1-\zeta)\sqrt{\mathtt{P}}\right]^{-(k+1)}}{\left[(1-\zeta)\sqrt{\mathtt{P}}\right]^{-1}-1}\\
&= \frac{\zeta(1-\zeta)^{-1}\mathtt{P}^{-1/2}}{\left[(1-\zeta)\sqrt{\mathtt{P}}\right]^{-1}-1}\cdot\mathtt{P}^{-k/2}\cdot\mathtt{BCR\_P}
\end{aligned}$$

To maintain $\mathtt{TRR\_P/RR\_P} > \sqrt{p}$ requires $\zeta > \sqrt{\mathtt{P}}/(1+\sqrt{\mathtt{P}})$ and so

$$\zeta_{\text{start-up}} = \frac{\sqrt{\mathtt{P}}}{1+\sqrt{\mathtt{P}}} \tag{27}$$

is the recommended value for ζ in start-up mode. $\mathtt{TRR\_P}$ is initialized to $(\mathtt{P}-1)/(\log\mathtt{P})\cdot\mathtt{BCR\_P}$ when the first base channel packet arrives because this is the average rate of the base channel.

In contrast to the actually received rate measured by $\mathtt{TRR\_P}$, $\mathtt{ARR\_P}$ is the receiver's best estimate of the *subscribed* rate. Thus, it uses a raw reception rate that includes lost packets, and it is adjusted to reflect the time-varying rates on the channels and the increases due to new subscriptions. As described in Section VI-D.2, $\mathtt{ARR\_P}$ is used to determine whether joining the next wave channel will cause the receiver to exceed the target rate. It must be adjusted to include the effects of joins immediately because measured reception rates will take some time to show increases.

Like $\mathtt{TRR\_P}$, $\mathtt{ARR\_P}$ is initialized to $(\mathtt{P}-1)/(\log\mathtt{P})\cdot\mathtt{BCR\_P}$ when the first base channel packet arrives. Subsequently at each epoch boundary, $\mathtt{IRR\_P}$ is calculated as the number of packets received and lost (as indicated by missing $\mathtt{PSN}$s) in the previous epoch divided by $\mathtt{EL}$. To account for the decay in transmission rates along with EWMA filtering of $\mathtt{IRR\_P}$, $\mathtt{ARR\_P}$ is updated to $(1-\beta)\cdot\mathtt{P}^{\mathtt{EL/TSD}}\cdot\mathtt{ARR\_P}+\beta\cdot\mathtt{IRR\_P}$. When a wave channel is joined to increase $\mathtt{NWC}$ from i to $i+1$, $\mathtt{ARR\_P}$ is updated to $\Gamma_i\cdot\mathtt{ARR\_P}$. Finally, at every time slot boundary, $\mathtt{ARR\_P}$ is increased by $(1-\mathtt{P})\cdot\mathtt{BCR\_P}$ because of the jump in the base channel rate and decreased by $\mathtt{BCR\_P}$ if the receiver was subscribed to the wave channel that has just gone quiescent.[13]

The update rules for $\mathtt{ARR\_P}$ and the join rule combine to form a nonlinear, discrete-time dynamical system with a discontinuous next-state map that is complicated further by noise in $\mathtt{IRR\_P}$. If $\mathtt{ARR\_P}$ is exactly correct, *i.e.*, equal to the sum of the rates of the subscribed channels, and $\beta = 0$, the rules above will cause $\mathtt{ARR\_P}$ to remain correct.[14] The contribution from $\mathtt{IRR\_P}$ with $\beta > 0$ is intended to drive any error in $\mathtt{ARR\_P}$ to zero. It turns out that it is most difficult to stabilize $\mathtt{ARR\_P}$, by which we mean to drive error in $\mathtt{ARR\_P}$ to zero, when $\mathtt{NWC}$ varies between 0 and 1. With a noise-less fluid model for $\mathtt{IRR\_P}$, it can be shown that

$$\beta \ge 1-\left(\frac{\mathtt{P}}{1+\mathtt{P}}\right)^{\mathtt{EL/TSD}}$$

insures the stability of $\mathtt{ARR\_P}$. Since small β is desirable to reject the noise in $\mathtt{IRR\_P}$, the recommended value of β for normal operation is

$$\beta_{\text{normal}} = 1-\left(\frac{\mathtt{P}}{1+\mathtt{P}}\right)^{\mathtt{EL/TSD}}. \tag{28}$$

Also, a smaller value of β is desirable for start-up mode since $\mathtt{IRR\_P}$ may lag significantly behind the subscribed rate.

The analysis that inspires Eq. (28) neglects network-induced variations in $\mathtt{IRR\_P}$, the fact that $\mathtt{EL}\cdot\mathtt{IRR\_P}$ is an integer, and the vagaries of sender operation. These factors can cause $\mathtt{ARR\_P}$ to drift to too large of a value when $\mathtt{NWC}$ varies between 0 and 1. Nevertheless, we believe that β_{normal} is well-chosen. To stabilize $\mathtt{ARR\_P}$, one additional step is added to the update of $\mathtt{ARR\_P}$ at each epoch boundary: After the EWMA computation, $\mathtt{ARR\_P}$ is clipped through

$$\mathtt{ARR\_P} = \min\left\{\mathtt{ARR\_P},\ \frac{1}{\mathtt{TSD}}\left\lceil\mathtt{TSD}\cdot\frac{(1/\mathtt{P})^{\mathtt{NWC}+1}-1}{(1/\mathtt{P})-1}\cdot\mathtt{BCR\_P}\right\rceil\right\}. \tag{29}$$

The second term in the min operation is the highest rate consistent with the given $\mathtt{NWC}$. The fluid model would give

$$\left(\prod_{i=0}^{\mathtt{NWC}-1}\Gamma_i\right)\cdot\mathtt{BCR\_P} = \frac{(1/\mathtt{P})^{\mathtt{NWC}+1}-1}{(1/\mathtt{P})-1}\cdot\mathtt{BCR\_P},$$

and the term used is adjusted for the packet scheduling technique illustrated in Fig. 16.

Finally, the accuracy of $\mathtt{ARR\_P}$ is important for leaving start-up mode at the right rate (see Section VI-D.4). Since

[13]Because the receiver is subscribed to wave channels consecutively from the one with lowest rate, the receiver will be subscribed to the wave channel that has just gone quiescent unless it is subscribed only to the base channel.

[14]This statement should be qualified further. It does not apply when the non-exponential beginnings of waves, as shown in Fig. 14, are encountered.

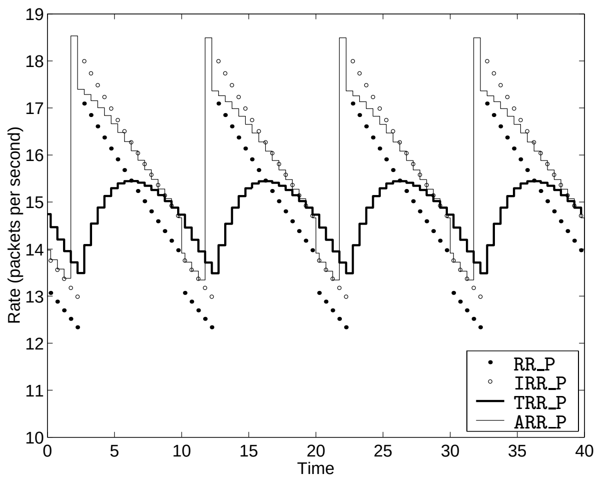


Fig. 20. The variants of the reception rate. RR_P and IRR_P are shown as discrete time series to emphasize that they are used at end of each epoch and then discarded. TRR_P and ARR_P are persistent state variables. ARR_P changes not only at epoch boundaries, but also at time slot boundaries.

buffer build-up may cause IRR_P to not reflect the subscribed rate at all—and this is something we attempt to detect to leave start-up mode—a value of $\beta_{\mathrm{start-up}} = 0$ is used. To improve the accuracy of ARR_P, an adjustment is made at the first time slot boundary detected by the receiver. With t being the time from the first packet arrival (when ARR_P was initialized to $(\mathtt{P}-1)/(\log \mathtt{P}) \cdot \mathtt{BCR\_P}$) to the first time slot boundary, ARR_P is updated to

$$\mathtt{ARR\_P} = \frac{\mathtt{P}^{(\mathtt{TSD}-t)/\mathtt{TSD}}}{(\mathtt{P}-1)/(\log \mathtt{P})} \cdot \mathtt{ARR\_P}.$$

This calculation adjusts ARR_P consistent with the time the receiver entered the session relative to the nearest time slot boundary.

Fig. 20 shows the two raw and two averaged reception rates in an idealized situation. For this demonstration, $\mathtt{TSD} = 10$, $\mathtt{EL} = 0.5$, and the epoch boundaries are offset from the time slot boundaries by 0.25 seconds. IRR_P corresponds to a receiver that periodically varies between $\mathtt{NWC} = 5$ and $\mathtt{NWC} = 6$ and the RR_P shown is consistent with exactly 5% packet loss in each epoch. TRR_P is updated at every epoch boundary and, as expected, is an averaged version of RR_P with about 40% of the peak-to-peak variation of RR_P. ARR_P is updated at epoch boundaries and time slot boundaries and tracks IRR_P closely. In making this plot, it was assumed that two epochs pass while each join message is processed; this lag contributes to the sharp peaks in ARR_P.

D.4 Enhancements beyond equation-based operation

Following an equation-based approach to congestion control is primarily motivated by achieving long-term fairness between sessions. One challenge in equation-based congestion control is to choose a rate before there are valid or reasonable estimates of network conditions. The approach to session start-up in WEBRC was outlined in Section III-B and is detailed below. Another departure from the equation-based approach is motivated by keeping queue occupancies low. This was outlined in Section III-C and is also detailed below.

Start-up mode A receiver is said to be in *start-up mode* when it first enters a session. Before a single packet has been lost, there seems to be no reasonable way to estimate a loss event rate. Also, there is relatively little information from which to estimate an average MRTT.[15] Therefore the decisions on whether to join waves are based on considerations other than maintaining a TCP-friendly rate.

The goal in start-up mode is to quickly reach a number of subscribed channels that matches the available bandwidth between the sender and the receiver and then let the equation-based operation take over. An aggressive way to achieve this is to join one wave channels at each epoch boundary until a loss is detected. This is perfectly reasonable if there is negligible network buffering and negligible propagation and message processing delay because the first join that puts the subscribed rate over capacity between the sender and the receiver will immediately cause a packet loss. In this case, the receiver overshoots by at most one wave channel. Eventually, with appropriate evolution of all the state variables, the receiver will time its joins so that its loss event rate, MRTT, and peak reception rate approximately satisfy Eq. (6).

Of course, round trip times and buffering are not negligible, so this strategy may cause the subscribed rate to greatly exceed the available bandwidth. This must be avoided 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. The receiver thus joins waves less aggressively than at every epoch boundary and leaves start-up mode when a loss is detected or when any of three other conditions that imply full usage of available bandwidth is satisfied. The early detection of full bandwidth utilization, before any packets are lost, can greatly reduce overshoot of the available bandwidth.

Instead of introducing a Boolean variable to indicate start-up mode, WEBRC uses $\mathtt{SSR\_P} = \infty$ to indicate start-up mode. With $\mathtt{SSR\_P} = \infty$, Eq. (5) yields MRR_P independent of all receiver state. To prevent the subscribed rate from greatly exceeding the true reception rate, Eq. (7) is used. This limit on the target rate is mild, and $\mathtt{TRATE\_P} \geq \Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P}$ is usually satisfied throughout start-up. Thus the normal condition to determine whether to join is satisfied at every epoch boundary. To prevent the reception rate from increasing too quickly, the receiver is barred from joining channel $(i+1)_{\mathtt{T}}$ until a full epoch has elapsed after the reception of the first packet on channel i. This requirement is explained by the reception rate-based mechanism for leaving start-up described below.

Leaving start-up: Packet loss A WEBRC receiver must leave start-up mode if it registers a packet loss. Just as in normal operation, the slow start rate is set to

[15]Furthermore, the MRTT measurements have higher variance at low rates because the interpacket spacings on the waves are larger at low rates, so the information is noisy.

$\texttt{SSR\_P} = \max\{\texttt{P} \cdot \texttt{TRR\_P}, \texttt{SSMINR\_P}\}$. (This finite value of $\texttt{SSR\_P}$ marks the end of start-up mode.) The reception rate $\texttt{TRR\_P}$ at the time of the first loss is used to initialize the state variables that determine the loss event rate, as described on page 30.

Leaving start-up: Increase in MRTT Packet losses are generally due to overflows of drop-tail queues or AQM actions that reflect high queue occupancy. In either case, packet losses can be avoided or reduced if reception rate increases are halted when increasing queue occupancy (or queueing delay) is detected.

In WEBRC, this philosophy is applied by leaving start-up mode if there is a sharp increase in MRTT. The obvious difficulty is to have a proper threshold for a "sharp" increase. We use as a threshold the maximum increase in MRTT that can be explained entirely by the interpacket spacing in the fluid transmission model. It may be appropriate to increase the threshold to account for limited timing granularity of the receiver or known burstiness of the sender.

Suppose the last two joined channels are i and $j = (i+1)_{\texttt{T}}$ and that the first packet from channel j has arrived. If there has been no increase in queuing delays and the receiver is grafted to the multicast tree at the same router for both joins, then there is a constant underlying MRTT value. Define $t_i = \texttt{FIRST}_i - \texttt{JOIN}_i$ and $t_j = \texttt{FIRST}_j - \texttt{JOIN}_j$. Then t_i and t_j should reflect exactly the same MRTT, which we denote by t, but with potentially different positive biases due to interpacket spacing. Define $t_{\text{increase}} = t_j - t_i$. The largest value of t_{increase} that is consistent with the hypothesis of no change in the underlying MRTT is obtained if $t_i = t$ and t_j is t plus the full amount of the interpacket spacing on channel j. If the total subscribed rate in packets per second is R, the fluid-model rate on channel j is given by

$$R_j = \frac{1 - \texttt{P}}{1 - \texttt{P}^{\texttt{NWC}+1}} \cdot R.$$

The initialization and evolution of $\texttt{ARR\_P}$ in start-up mode imply

$$\texttt{ARR\_P} \leq \frac{\texttt{P} - 1}{\texttt{P} \log \texttt{P}} \cdot R.$$

Thus,

$$R_j \geq \frac{1 - \texttt{P}}{1 - \texttt{P}^{\texttt{NWC}+1}} \cdot \frac{\texttt{P} \log \texttt{P}}{\texttt{P} - 1} \cdot \texttt{ARR\_P} = \frac{-\texttt{P} \log \texttt{P}}{1 - \texttt{P}^{\texttt{NWC}+1}} \cdot \texttt{ARR\_P}$$

is a bound on the rate of channel j. We should thus have

$$t_{\text{increase}} \leq \frac{1 - \texttt{P}^{\texttt{NWC}+1}}{-\texttt{P} \log \texttt{P}} \cdot \frac{1}{\texttt{ARR\_P}} \tag{30}$$

when the underlying MRTT has not increased. Violations of this inequality cause an end to start-up mode. The slow start rate is set to $\texttt{SSR\_P} = \max\{\texttt{P} \cdot \texttt{TRR\_P}, \texttt{SSMINR\_P}\}$ and the loss event rate variables are initialized consistent with the current value of $\texttt{TRR\_P}$.

Note that the criterion (30) will have no "false positives," *i.e.*, it will not be violated without an increase in queueing delay or change in multicast topology—if the sender behavior is completely consistent with the fluid model. One could use the following alternative strategy to determine a threshold for an MRTT increase: estimate the distribution of $t_i - t$, which will be uniform on an interval $[0, c_i]$; estimate the distribution of $t_j - t$, which will be uniform on an interval $[0, c_j]$, $c_j \neq c_i$; compute the distribution of $t_i - t_j$ which via convolution will be trapezoidal; determine a threshold based on an acceptable probability of false positives. We do not follow this approach because we consider false positives to be worse than false negatives since there are other mechanisms to leave start-up mode. Furthermore, discrepancies between the fluid model and the actual behavior of the sender already create a small, positive probability of false positives.

In a situation where other receivers are not affecting MRTT measurements, an increase in MRTT will probably not be detected before the receiver leaves start-up mode due to a lagging reception rate (described below). This MRTT-based rule is particularly important when there are multiple receivers in the same WEBRC session below a common bottleneck. Suppose a new receiver joins a session in which the other receivers below a shared bottleneck are operating in a steady state. The MRTT measurements of the receiver will initially be low because the other receivers have earlier join times. If the rapid rate increase in start-up mode causes the join time of the new receiver to be significantly earlier than that of its nearby neighbors, a large increase in MRTT is likely. Thus the new receiver will leave start-up mode before causing too much packet loss for other receivers in the session. Of course, it is also possible that any one of the other receivers will cause a loss on the bottleneck link that will force the receiver in question to leave start-up mode.

Leaving start-up: Lagging reception rate Since MRTT is measured as soon as a join takes effect, an increase in MRTT due to queueing means that the *previous* join increased the reception rate sufficiently to create a queueing delay. We would like to be able to detect such queueing delay before joining an additional wave. To do this, we attempt to determine whether the reception rate is consistent with the cumulative subscribed rate or, alternatively, is below the subscribed rate and hence suggests a bottleneck bandwidth has been reached.

Deciding whether a reception rate is consistent with an expected rate is straightforward if one averages over a long period of time. Of course, in opposition to this is the goal of reaching an appropriate reception rate as quickly as possible. In WEBRC this decision is relevant at epoch boundaries since all joins are made at epoch boundaries. It is deemed that a decision can be made as soon as one full epoch has passed since packets started arriving on the last joined wave. The receiver is barred from joining until this full-epoch condition is satisfied.

The decision on whether a bottleneck bandwidth is indicated is made by comparing $\texttt{TRR\_P}$ to the smallest value that is consistent with no queue buildup. Suppose the most recently joined wave channel is i and one full epoch has passed since the first packet was received on channel i. (Algebraically, this means the current time t satisfies $t - 2 \cdot \texttt{EL} < \texttt{FIRST}_i < t - \texttt{EL}$.) The TRR_P computed at this epoch boundary can be written as

$$\begin{aligned}\texttt{TRR\_P}_0 &= \zeta \cdot \texttt{RR\_P}_0 + (1-\zeta) \cdot \texttt{TRR\_P}_1 \\ &= \zeta \cdot \texttt{RR\_P}_0 + (1-\zeta)\zeta \cdot \texttt{RR\_P}_1 + (1-\zeta)^2 \cdot \texttt{TRR\_P}_2 \quad (31)\end{aligned}$$

where subscripts represent epochs into the past and ζ is shorthand for $\zeta_{\text{start-up}}$. Each term in Eq. (31) is approximated or bounded to determining a threshold for TRR_P. Rates are approximated as constant over epochs, but not constant over longer periods.

The reception rate in the current epoch should be reflected by ARR_P: $\texttt{RR\_P}_0 \approx \texttt{ARR\_P}$. The join took effect in the previous epoch so that the reception rate was $\texttt{P}^{-\texttt{EL}/\texttt{TSD}} \cdot \texttt{ARR\_P}$ for the last $t - \texttt{FIRST}_i - \texttt{EL}$ seconds of the epoch and was $\texttt{P}^{-\texttt{EL}/\texttt{TSD}} \cdot \texttt{ARR\_P}/\Gamma_{\texttt{NWC}-1}$ for the first $2 \cdot \texttt{EL} - (t - \texttt{FIRST}_i)$ seconds of the epoch. Thus

$$\texttt{RR\_P}_1 \approx \left[\theta + \frac{1-\theta}{\Gamma_{\texttt{NWC}-1}}\right] \texttt{P}^{-\texttt{EL}/\texttt{TSD}} \cdot \texttt{ARR\_P}$$

where

$$\theta = \frac{t - \texttt{FIRST}_i - \texttt{EL}}{\texttt{EL}}.$$

Two epochs ago, TRR_P should have reflected the subscribed rate at that time, $\texttt{P}^{-2 \cdot \texttt{EL}/\texttt{TSD}} \cdot \texttt{ARR\_P}/\Gamma_{\texttt{NWC}-1}$, but by the design of $\zeta_{\text{start-up}}$ could be expected to lag by a factor of $\sqrt{\texttt{P}}$. Thus

$$\texttt{TRR\_P}_2 \lessapprox \sqrt{\texttt{P}} \cdot \texttt{P}^{-2 \cdot \texttt{EL}/\texttt{TSD}} \cdot \texttt{ARR\_P}/\Gamma_{\texttt{NWC}-1}.$$

Finally, in attempting to detect whether TRR_P is too small to be consistent with the subscribed rate, it is important to allow some variation due to the non-fluid nature of the waves and the fact that rates are measured over (potentially short) epochs. The threshold used for leaving start-up is

$$\begin{aligned}R_{\min} \;=\; &\left\{\zeta + (1-\zeta)\zeta\left(\theta + \frac{1-\theta}{\Gamma_{\texttt{NWC}-1}}\right) \cdot \texttt{P}^{-\texttt{EL}/\texttt{TSD}}\right. \\ &\left. + (1-\zeta)^2 \cdot \sqrt{\texttt{P}} \cdot \texttt{P}^{-2\cdot\texttt{EL}/\texttt{TSD}} \cdot \frac{1}{\Gamma_{\texttt{NWC}-1}}\right\} \cdot \texttt{ARR\_P} \\ &- \frac{2}{\texttt{EL}}. \qquad (32)\end{aligned}$$

If TRR_P is below $R_{\min}$, the receiver does not join; it sets the slow start rate to $\texttt{SSR\_P} = \max\{\texttt{TRR\_P}, \texttt{SSMINR\_P}\}$ and initializes the loss event rate variables consistent with the TRR_P. Note that the slow start rate setting is larger than when a packet is lost or when a large MRTT is detected. This is because having a lagging reception rate while not inducing packet loss and not measuring a large MRTT suggests a mild overshoot.

Leaving start-up: Maximum reception rate The final reason to leave start-up mode is that the maximum reception rate has been reached. Note that $\texttt{TRATE\_P} \leq \texttt{MRR\_P}$ so ARR_P will never reach MRR_P. This rule is triggered when $\Gamma_{\texttt{NWC}} \cdot \texttt{ARR\_P}$ first exceeds MRR_P or SR_P. The slow start rate is set to $\texttt{SSR\_P} = \max\{\texttt{TRR\_P}, \texttt{SSMINR\_P}\}$ and the loss event rate variables are initialized consistent with TRR_P.

Leaving start-up: Initializing the loss event rate When the receiver leaves start-up mode, it initializes all of the variables involved in computing LOSSP. This initialization occurs upon the first lost packet or before any packets have been lost. Therefore the initialization cannot reasonably be based on a measured loss rate. Instead, the initializations are intended to make REQN give a specified desired rate.

The first step is to determine the desired LOSSP. LOSSP is then used to determine values for W, X, Y, and Z. It is desired for REQN to equal TRR_P. Thus Eq. (6) is rearranged to obtain

$$p\left(1 + 9p\left(1 + 32p^2\right)\right)^2 = \frac{3/2}{(\texttt{ARTT} \cdot \texttt{REQN})^2}. \qquad (33)$$

There is no closed-form solution for p. A simple iteration that converges to a solution is given in the *ns* code for the receiver. From an estimate p_k, the iteration produces a new estimate p_{k+1} by solving an approximate version of Eq. (33) where $(1+9p(1+32p^2))^2$ is replaced with a Taylor expansion about p_k. To match a desired value of $\texttt{LOSSP} = p^*$, the initializations are $\texttt{W} = 0$, $\texttt{X} = 0$, $\texttt{Y} = 0$, and $\texttt{Z} = 1/p^*$. These values are not unique in giving $\texttt{LOSSP} = p^*$; positive values of X and Y with $\texttt{X}/\texttt{Y} = 1/p^*$ will maintain $\texttt{LOSSP} = \texttt{P}^*$ and will make LOSSP more stable.

Final remark on leaving start-up mode Looking through all of Section VI-D, the most complicated calculation in the receiver operation is that of $R_{\min}$ as the threshold for leaving start-up mode because the reception rate is lagging behind the subscribed rate. (Various expressions involving only P and NWC can be precomputed and stored for use through the duration of a session.) Note that $R_{\min}$ is computed only during start-up mode. Furthermore, a simpler expression could be used at the expense of lesser accuracy in leaving start-up with full usage of available bandwidth and little overshoot. The long-term behavior should not be affected by such a simplification.

Detecting queueing in normal operation After a WEBRC receiver leaves start-up mode, its behavior deviates from being purely equation-based so that it can avoid causing large queue buildups and large bursts of packet loss. It does this by attempting to determine if its reception rate is too constant to be consistent with the decaying rates on all of the channels. This suggests that the reception rate actually reflects a bottleneck bandwidth. The receiver then does not join a wave and it resets the loss event rate variables so that the target rate does not diverge. This rule is most likely to be invoked when a WEBRC session is the only flow over a bottleneck link and the amount of available buffering is large. Since the last hop at the receiver end is often the bottleneck, such a situation is not uncommon.

With a purely equation-based approach, packet losses are necessary to stabilize the reception rate; without losses, $\texttt{LOSSP} \to 0$ so $\texttt{REQN} \to \infty$. When there is a large buffering capacity at the bottleneck link, packet losses may come only after a receiver has increased its subscribed rate well over the available bandwidth. Then the WEBRC session is continuing to overload the bottleneck link until the subscribed rate has decayed below the available bandwidth, which may take a long time. The result is massive packet loss.

The mechanism currently implemented in WEBRC for detecting queueing is very simple. Outside of start-up mode, the receiver tracks the maximum `RR_P` since the last join was issued; this is denoted $\mathtt{RR\_P}_{\max}$. When `TRATE_P` is at least $\Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P}$ but not larger than `SR_P`, the receiver applies one additional condition before joining the next wave: it checks

$$\mathtt{RR\_P} > \max\left\{\mathtt{RR\_P}_{\max} - \frac{2}{\mathtt{EL}},\ \mathtt{P}\cdot\mathtt{RR\_P}_{\max}\right\}.$$

If this inequality holds, the receiver infers that its reception rate reflects the emptying of queues. It does not join and it resets the loss event rate variables so that $\mathtt{REQN} = \Gamma_{\mathtt{NWC}} \cdot \mathtt{ARR\_P}$. This reset value is chosen so that if there are no losses in the next epoch, the receiver will again have a high enough target rate to justify a join.

Detecting queueing in this manner works very well when there is no competing traffic. Competing traffic will add to the variance of `RR_P` and hence should make $\mathtt{RR\_P}_{\max}$ larger in relation to the local average of `RR_P`. This should make it harder to trigger this rule and less like that this rule would be triggered at consecutive epoch boundaries.

VII. Simulation Results

We have simulated the WEBRC protocol extensively with *ns* [20]. These simulations are sampled and summarized here. Our implementation of WEBRC and some simulation scripts are available on-line at http://www.digitalfountain.com/technology/library/webrc/ so that the reader can verify our results and experiment further. Except as specified, the parameters given in Tables I and III and drop tail queues were used.

The specifics of the protocol have remained in flux as we have written this document so some of the simulation results presented here were obtained with superseded versions of the protocol. However, we believe that these results fairly represent the performance of the current WEBRC protocol. More simulation is needed and is underway; results will be presented in future reports.

A. Single-receiver sessions without dynamic competition

A.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

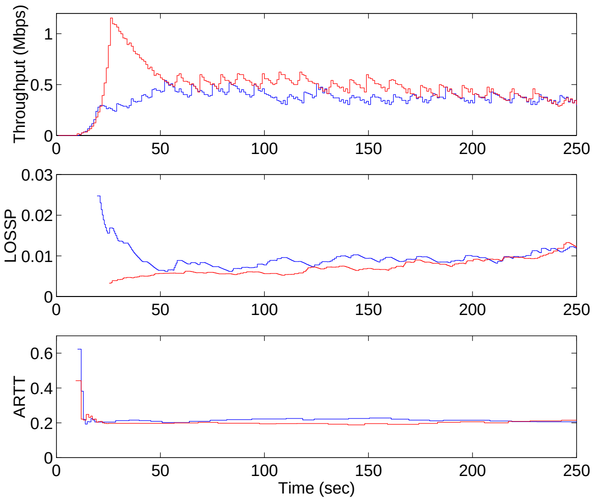


Fig. 21. Two trials of a single-receiver WEBRC session on a link with 1% random packet loss and 0.2 sec RTT. `LOSSP` and `ARTT` converge to the correct values, with a small amount of steady-state variation. The throughput varies around the expected value, approximately periodically with period `TSD`.

at a random time uniformly distributed over $[0, \mathtt{TSD}]$ to eliminate any phase synchronization between sender and receiver. Each simulation runs for 500 seconds.

Fig. 21 shows the first 250 seconds of two such experiments in which the packet loss probability is 1% and the MRTT is 0.2 sec. The three panels give `ARTT`, `LOSSP`, and the throughput.[16] 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`.

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

[16] 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.

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.

Consider the idealized situation where the epoch length is very short, waves are joined instantly, and the receiver reaches a target rate R with each join. Then the steady-state reception rate $R(\tau)$ packets per second will be periodic with period $\mathtt{TSD}$ and will decay exponentially except for jumps up to rate R at joins and drops down by $\mathtt{P}$ at time slot boundaries. Letting t denote the time from each join to the next time slot boundary, the average throughput $\bar{R}$ is given by

$$\begin{aligned}\bar{R} &= \frac{1}{\mathtt{TSD}}\int_0^{\mathtt{TSD}} R(\tau)\,d\tau \\ &= \int_0^t R\cdot \mathtt{P}^{\tau/\mathtt{TSD}}\,d\tau + \int_t^{\mathtt{TSD}} (R\cdot \mathtt{P}^{t/\mathtt{TSD}} - \mathtt{P})\mathtt{P}^{(\tau-t)/\mathtt{TSD}}\,d\tau.\end{aligned}$$

Evaluating the integrals and noting that $t \in [0, \mathtt{TSD}]$ yields

$$\frac{1-\mathtt{P}}{\ln(1/\mathtt{P})}(R-\mathtt{P}) \le \bar{R} \le \frac{1-\mathtt{P}}{\ln(1/\mathtt{P})}R. \qquad (34)$$

Thus, the difference between peak and average reception rates reduces the throughput of WEBRC approximately by the factor $(1-\mathtt{P})/\ln(1/\mathtt{P})$. For the default values of $\mathtt{P} = 0.75$ and $\mathtt{TSD} = 10$ 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 $\mathtt{ARTT}$ from the beginning of a loss event. This has only a mild effect on throughput, except when p is large. One way to estimate p is as $1/K$, where K is the average interval between losses. (This is a biased estimate, but the bias is very small for small p.) In computing loss event rate, the interloss interval is incremented for all the packets received and lost during a loss event. Thus, assuming for simplicity a constant rate of R packets per second, the loss event rate is approximated as $(R\cdot\mathtt{MRTT}+K)^{-1}$. Using Eq. (2) to approximate R, the expected receiver estimate is given approximately by

$$\mathtt{LOSSP} = p\Bigg/\left(1+\sqrt{\frac{3p}{2}}\right). \qquad (35)$$

$\mathtt{LOSSP}$ is thus expected to be slightly smaller than p, and the ratio between the two approaches 1 as p approaches zero. A relatively large loss probability of $p = 0.02$ results in only 17% difference between p and $\mathtt{LOSSP}$. The effect on the throughput is the square root of this, or about 8%. The difference between loss event rate and packet loss probability is not a flaw; it is consistent with "emulating the best behavior of a conformant TCP implementation" [6].

While the difference between packet loss fraction and loss event rate tends to slightly increase the throughput of a WEBRC session, the prohibition against joining during loss events counteracts this effect somewhat. The expected time between the end of a loss event and the next loss is $(p\cdot R)^{-1}$, which by using Eq. (2) is approximately

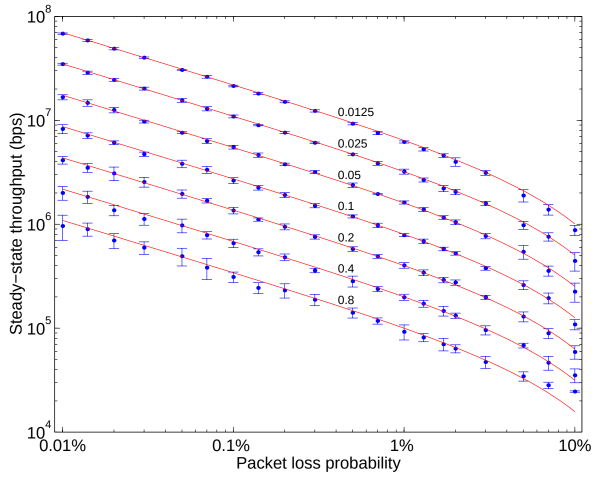


Fig. 22. Average steady-state throughputs from simulations of single-receiver WEBRC sessions in which the packet loss probability and RTT are varied. The simulation results (·'s with error bars indicating ± one standard deviation) are compared to the desired throughput from Eq. (1), adjusted according to the upper bound of Eq. (34). Curves are labeled with the MRTT in seconds.

$t_{\mathrm{RTT}}/\sqrt{3p/2}$. The fraction of time in the loss event state is thus approximately

$$\frac{t_{\mathrm{RTT}}}{t_{\mathrm{RTT}}+\frac{t_{\mathrm{RTT}}}{\sqrt{3p/2}}} \approx \sqrt{\frac{3p}{2}}.$$

For large values of p, this is significant enough to cause the receiver to miss some of its opportunities to increase its rate.

The results shown in Fig. 21 are consistent with our analysis. Evaluating Eq. (35) 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. (34) 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. 22.[17]

To assess whether WEBRC achieves the desired throughput more generally, eight trials were conducted for each RTT in $\{0.0125$ sec, 0.025 sec, $\ldots$, 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. (34), in Fig. 22. 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.

These experiments are quite convincing in showing that the dynamics of WEBRC, including the measurements of loss and MRTT and the process of joining and leaving wave channels, make the throughput follow Eq. (1) as desired. However, this does not mean that WEBRC has the same

[17]Here and in subsequent experiments we define the "steady-state throughput" as the average throughput over the last half of the simulation and we use simulation durations that are sufficient to make this period qualitatively appear as a steady state.

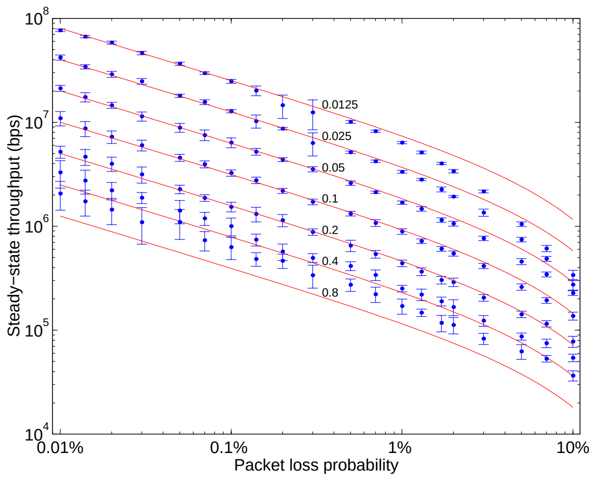


Fig. 23. Results of repeating the experiments reported in Fig. 22 for TCP. Average steady-state throughputs from simulations of TCP sessions in which the packet loss probability and RTT are varied are shown. Curves shown for comparison are computed with Eq. (1).

throughput as TCP. Repeating the same set of simulations with TCP in place of WEBRC reveals that TCP throughput is harder to predict than WEBRC throughput. Referring to the results in Fig. 23, the variance of WEBRC throughputs is lower than that of TCP throughputs except with very high packet loss. Also, Eq. (1) is simply not as predictive of TCP throughput *in ns simulations* as one might hope. It underestimates throughput when RTT is large and overestimates throughput when the loss is high and RTT is small. This points out the difficulty of assessing the TCP fairness of an Eq. (1)-based rate control protocol using *ns*.

A.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. 24 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 VI-D.4. 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

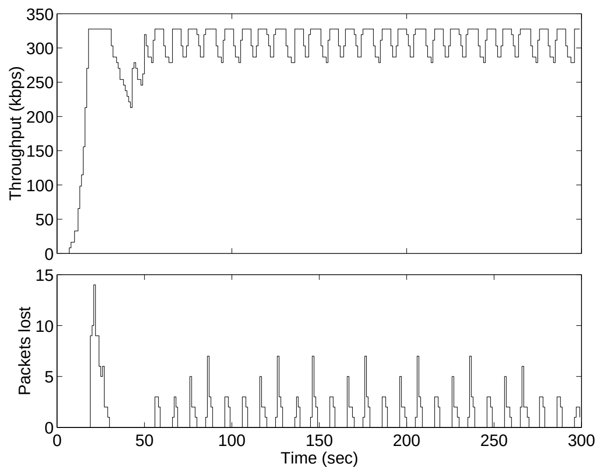


Fig. 24. Throughput and packet loss of a WEBRC session where the bottleneck bandwidth is 320 kbps, RTT is 0.1 sec, and buffer length is 4 packets.

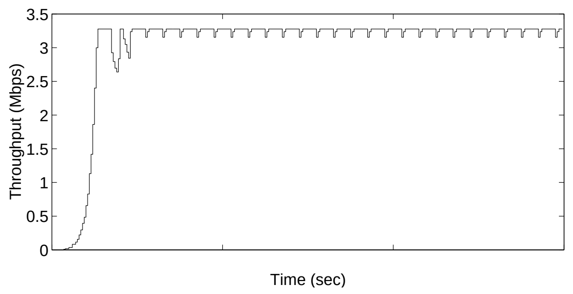


Fig. 25. Throughput of a WEBRC session where the bandwidth is 3.2 Mbps, RTT is 0.1 sec, and buffer length is 160 packets. There is no packet loss.

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 III-C can allow a WEBRC session to get nearly 100% bandwidth utilization. Fig. 25 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 `TRR_P` is lagging behind `ARR_P`—specifically that `TRR_P` falls below $R_{\min}$ computed with Eq. (32).

Fig. 26 summarizes a set of experiments in which the buffer length is varied. The bottleneck bandwidth is 1 Mbps (125 packets per second) and the MRTT is 0.1 sec. Eight simulations were run for each of seven buffer lengths 3, 6, 12, …, 192. The bandwidth utilization is high and increases with larger buffer lengths. For low buffer lengths,

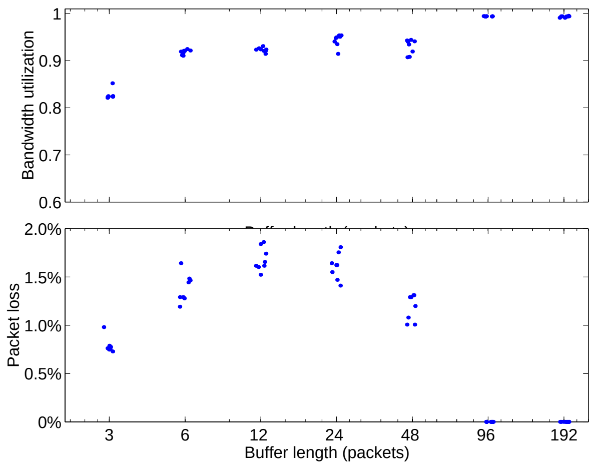


Fig. 26. Steady-state bandwidth utilization and packet loss percentages of WEBRC for various buffer lengths. The bottleneck bandwidth is 1 Mbps and the MRTT is 0.1 sec. Each configuration is simulated eight times, and a small dither is added to the horizontal components to make the points more discernable.

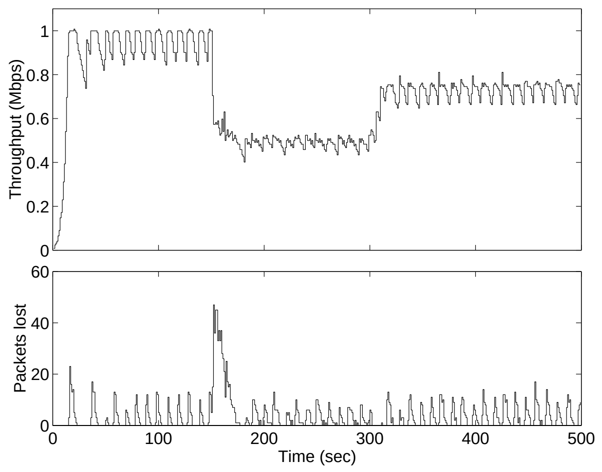


Fig. 27. A single-receiver WEBRC session responding to changes in available bandwidth. Initially, 1 Mbps is available. After 150 seconds a 500 kbps constant bit rate flow starts to compete, and after another 150 seconds it reduces its rate to 250 kbps.

packet losses correspond to the throughput roughly in accordance with Eq. (1). When the buffer lengths are large, WEBRC is able to stabilize without packet loss using the mechanism described in Section III-C; hence there is no packet loss.

A.3 Responsiveness to network changes

The results of the two previous subsections 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.

Fig. 27 shows the throughput and packet loss of the WEBRC session. Notice that the initial behavior is no different than in Fig. 24 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.

B. *Sharing with TCP*

Having similar behavior as TCP in isolation, as demonstrated in Section VII-A, 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 [9].

The experiments in which WEBRC sessions compete with TCP sessions are divided so that in the first set the desired behavior is for all the sessions to have approximately equal throughput and in the second set “fairness” requires a comparison of RTTs. The network topology for these experiments is for all senders to be connected to a single router, which is connected to another router with a link of limited bandwidth. The latter router is connected to all of the receivers.

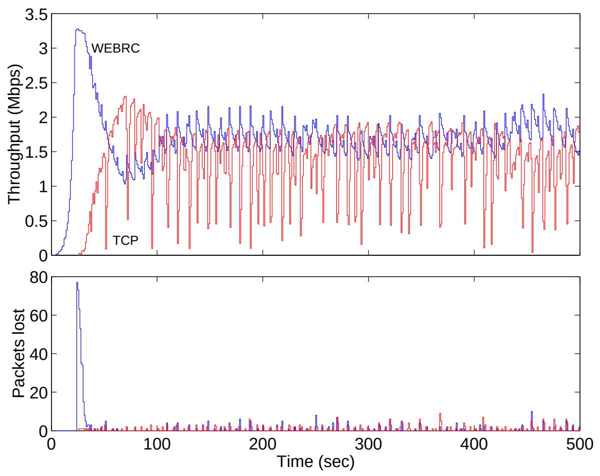
B.1 Sharing under identical conditions

When a WEBRC session and a TCP session share a bottleneck link and have the same RTT, we would like for them to have the same steady-state throughput despite complicated dynamic behavior. More precisely, recall that a WEBRC session uses a polite standard of fairness where it tries to make its *peak* rate TCP-friendly. Therefore its throughput is approximately $(1-\texttt{P})/\ln(1/\texttt{P})$ times the throughput of a TCP session under the same packet loss and RTT (see Eq. (34)). The ideal behavior to observe would thus be for the WEBRC session to get a fraction

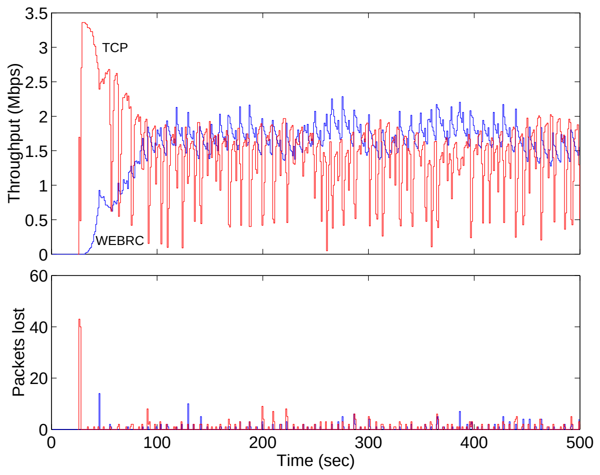
$$\frac{R_{\text{WEBRC}}}{R_{\text{WEBRC}}+R_{\text{TCP}}}=\frac{\frac{1-\texttt{P}}{\ln(1/\texttt{P})}}{\frac{1-\texttt{P}}{\ln(1/\texttt{P})}+1}=\frac{1-\texttt{P}}{1-\texttt{P}+\ln(1/\texttt{P})}$$

of the total throughput. With the default value $\texttt{P}=0.75$, this is 46.5% of the throughput.

Fig. 28 shows two simulations with one WEBRC session and one TCP session. In both cases the bottleneck band-



(a) WEBRC session starts first.



(b) TCP session starts first.

Fig. 28. A WEBRC session and a TCP session share a 3.2 Mbps link. Each has 0.1 sec RTT.

width 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%.

To assess fairness more thoroughly, competing WEBRC and TCP sessions with the same RTTs were simulated for all combinations of six bottleneck bandwidths, eight relative start times between the sessions, and seven buffer factors (as defined in Section VII-A.2). The results are summarized in Fig. 29. The average throughput seems uncorrelated with the starting times of the sessions. A slight trend of TCP fairing better as the buffer sizes increase is apparent and warrants further study. Most interesting is the dependence on the bottleneck bandwidth. The lowest bottleneck bandwidth is 50 kbps. The “fair” share for the WEBRC session is only 23 kbps. Since this rate is so close to the base channel rate, the WEBRC session does not drop its rate as far and as often as the TCP session; this makes the WEBRC session get more than its share of the bottleneck bandwidth. For fairness, the base channel rate should be set quite a bit lower than the lowest anticipated bottleneck bandwidth.

The simulations demonstrate the desired fairness on average, but there is significant variation in the experiments. In particular, when the experiment is repeated with two competing TCP sessions there is still significant—though less—variation in the throughput ratios (see Fig. 30). The deviation from fairness when two WEBRC sessions compete over a bottleneck is somewhat more than when two TCP sessions compete (see Fig. 31). The last phenomenon may be because `TSD`-periodic joins allow one session to consistently utilize more of the network buffering than the other; TCP would not fall into this periodic behavior.

B.2 Sharing in proportion to RTT

To see how well WEBRC and TCP sessions share in inverse proportion to their MRTTs and RTTs, the sharing experiments were repeated with each combination of WEBRC MRTT and TCP RTT in {0.05 sec, 0.1 sec, 0.2 sec, 0.4 sec}. The number of simulations summarized in the previous subsection is already quite large; introducing a sampling of MRTTs for the WEBRC session and of RTTs for the TCP session gives an unmanageable number of combinations of parameters. Thus, we fixed the bottleneck bandwidth to 1 Mbps and the buffer factor B to 2. Sixteen relative start times were used.

Fig. 32 shows the results of these experiments, as the mean and standard deviation of sixteen simulations (the sixteen relative start times). With an RTT ratio

$$\rho = \frac{(\text{MRTT of WEBRC})}{(\text{RTT of TCP})},$$

the designed behavior is for the throughput ratio to be

$$\frac{R_{\text{WEBRC}}}{R_{\text{TCP}}} = \frac{1}{\rho} \cdot \frac{1 - \mathtt{P}}{\ln(1/\mathtt{P})}.$$

Ratios predicted by this formula are shown with dashed lines. The trends in Fig. 32 follow the predictions, but the actual throughput ratios differ from the predictions by as much as a factor of 3.

C. Receiver interactions

Without any explicit communication between receivers, WEBRC receivers in the same session below the same bottleneck link tend to coordinate their joins and thus have nearly equal reception rates. This is largely due to the effect, discussed in detail in Section IV-I, of convergence of

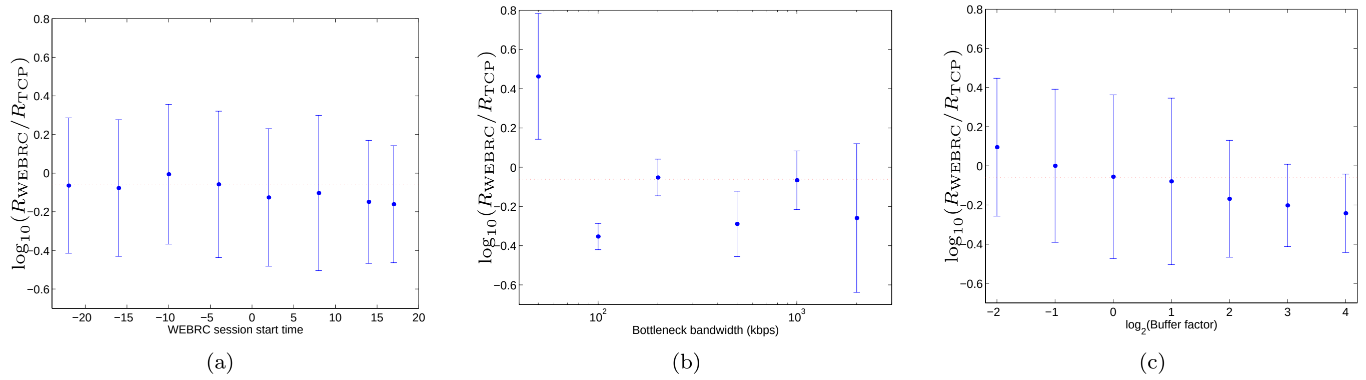


Fig. 29. Summary of experiments in which one WEBRC session and one TCP session share a bottleneck link. Both have RTTs of 0.1 sec. The average throughputs R_{WEBRC} and R_{TCP} are compared by computing the mean and standard deviation of $\log_{10}(R_{\mathrm{WEBRC}}/R_{\mathrm{TCP}})$. (a) Dependence on the start time of the WEBRC session relative to the TCP session starting at time 0. (b) Dependence on bottleneck bandwidth. (c) Dependence on the buffer factor. In all three plots, the dashed line represents the ratio $(1-\mathtt{P})/\ln(1/\mathtt{P})$ predicted by analysis.

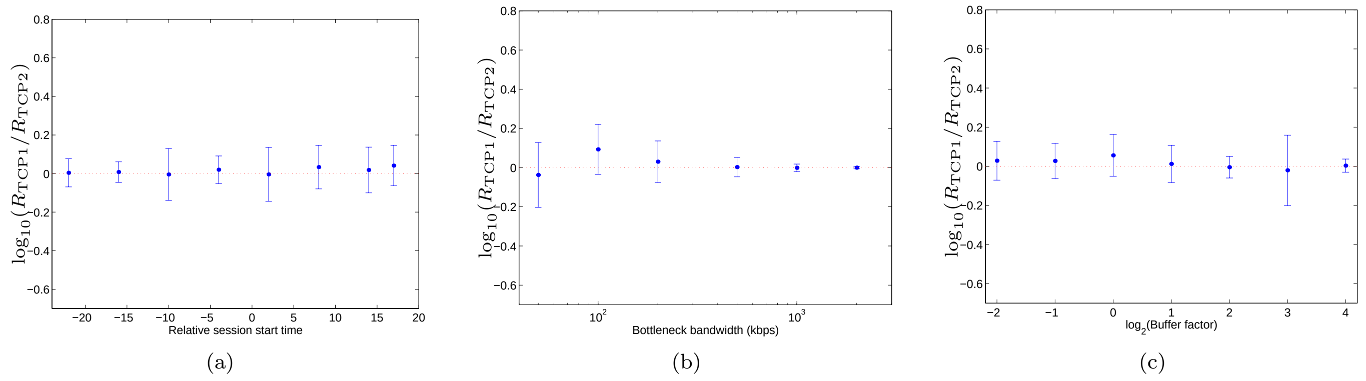


Fig. 30. Summary of experiments in which two TCP sessions share a bottleneck link. Both have RTTs of 0.1 sec. The average throughputs R_{TCP1} and R_{TCP2} are compared by computing the mean and standard deviation of $\log_{10}(R_{\mathrm{TCP1}}/R_{\mathrm{TCP2}})$. (a) Dependence on the relative start times of the sessions. (b) Dependence on bottleneck bandwidth. (c) Dependence on the buffer factor. In all three plots, the dashed line represents equal sharing.

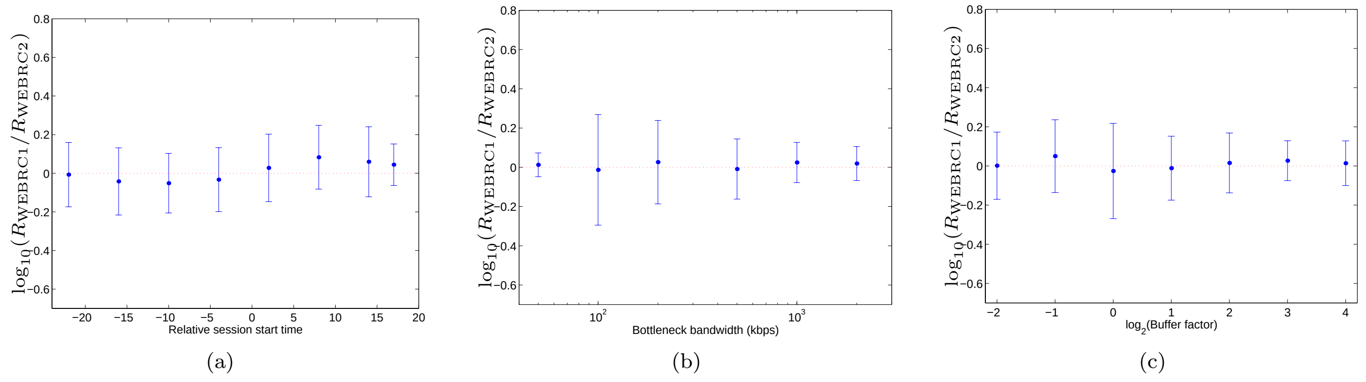


Fig. 31. Summary of experiments in which two WEBRC sessions share a bottleneck link. Both have MRTTs of 0.1 sec. The average throughputs R_{WEBRC1} and R_{WEBRC2} are compared by computing the mean and standard deviation of $\log_{10}(R_{\mathrm{WEBRC1}}/R_{\mathrm{WEBRC2}})$. (a) Dependence on the relative start times of the sessions. (b) Dependence on bottleneck bandwidth. (c) Dependence on the buffer factor. In all three plots, the dashed line represents equal sharing.

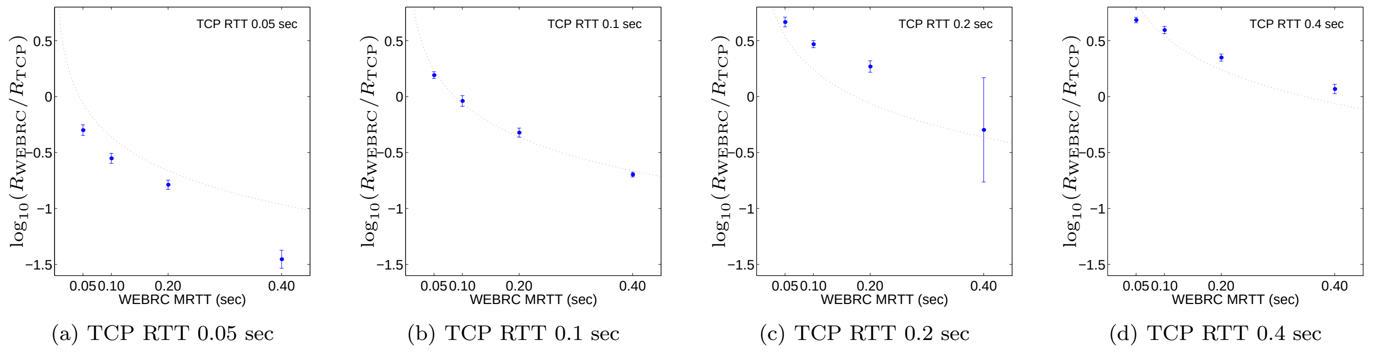


Fig. 32. Summary of experiments in which one WEBRC session and one TCP session sharing a bottleneck link possibly have different round trip times. The average throughputs R_{WEBRC} and R_{TCP} are compared by computing the mean and standard deviation of $\log_{10}(R_{\mathrm{WEBRC}}/R_{\mathrm{TCP}})$. Dashed lines indicate the designed behavior.

MRTT values of the receivers. This convergence and a self-stabilizing property that comes from the use of loss rates are demonstrated in the subsections below.

C.1 Coordination of two receivers

Consider two receivers in the same WEBRC session as illustrated in Fig. 7. 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, respectively.

Fig. 33(a) 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. 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%.

The same type of coordination occurs if Receiver B is the first to enter the session. This is shown in Fig. 33(b).

C.2 Loss-based equalization

Again consider receivers below a common bottleneck link in the same WEBRC session and suppose their MRTTs are equal. The dependence of the rate on the loss estimate provides another rate-equalizing mechanism.

Suppose Receiver A has reception rate R_A and Receiver B has reception rate R_B, with $R_A/R_B = c \geq 1$. Normally, one would expect more losses early in a wave when the rate is higher. Hence, the higher rate receiver would have slightly higher loss and thus the reception rates are pushed together over time. In the worst-case scenario,

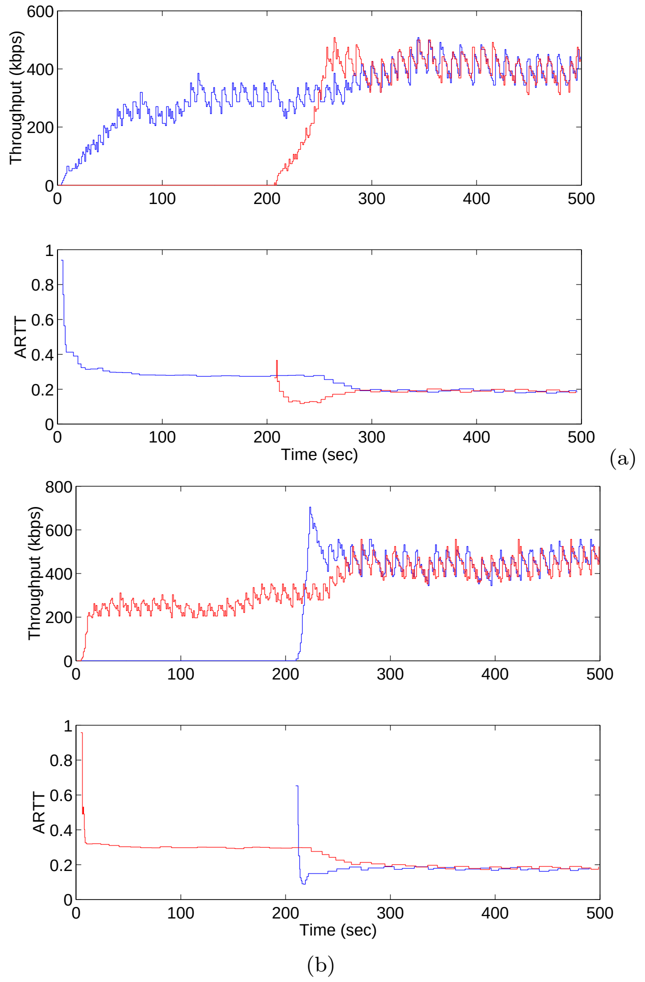


Fig. 33. The coordination of two receivers in the scenario depicted in Fig. 7. With reference to Fig. 7, $X = 0.2$ sec, $Y = 0.05$ sec, and $Z = 0.1$ sec. The shared link has 1% random packet loss. (a) Receiver A starts first. (b) Receiver B starts first.

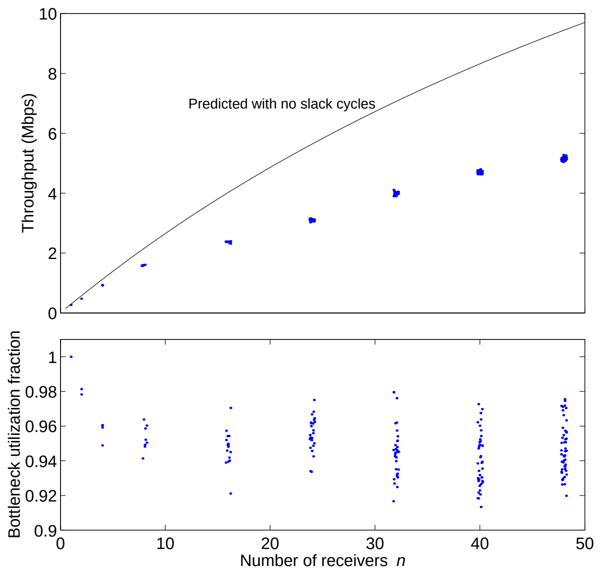


Fig. 34. The effect of the number of receivers sharing a common bottleneck on session throughput and efficiency of use of the bottleneck link. A shared link with 2% random packet loss and 0.198 sec RTT connects the sender to a router very close (0.002 sec RTT) to all of the receivers.

all lost packets are those common to the two receivers.[18] The loss fraction for the lower-rate receiver is higher, and hence the lower-rate receiver stays at a lower rate. While this argument may make it seem that the rates do not stabilize, there is still an equalizing effect.

When the receivers have a rate multiple of c as above, the loss fractions can differ by at most a factor of c: $p_B/p_A \leq c$. Since the target rate depends on $\sqrt{p}$, a rate multiple of $c \geq 1$ for one time interval tends to push the rate multiple to *at most* $\sqrt{c} \leq c$. Thus even in the case where no packets destined only for Receiver A are lost, the form of the target rate equation tends to equalize the rates of these receivers.

C.3 Effect of receiver density

To verify that a multi-receiver WEBRC session shares fairly as described in Section IV-K, 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 IV-E, 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.

Fig. 34 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), (34) and (35), 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. 34 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.

[18] This is plausible if the part of each wave destined only for Receiver A is absorbed in router buffers but buffer overflows start only after the join of Receiver B has taken effect.

Figure	P	TSD	SR_b	N	Q	T
27	0.75	10	$2 \cdot 10^6$	16	30	46
35(a)	0.5	10	$2 \cdot 10^6$	8	30	38
35(b)	0.875	10	$2 \cdot 10^6$	27	30	57
36(a)	0.75	5	$2 \cdot 10^6$	16	60	76
36(b)	0.75	20	$2 \cdot 10^6$	16	15	31

TABLE IV
PARAMETER VALUES USED IN SECTIONS VII-D.1 AND VII-D.2

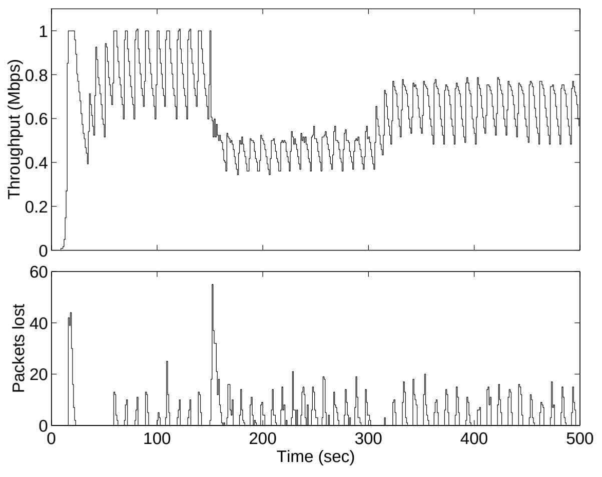
D. *Sensitivity to default parameters*

All of the simulations reported thus far have used the default parameters of P = 0.75, TSD = 10, and EL = 0.5. There is a trade-off associated with each parameter value. This final section of simulations briefly describes these trade-offs and demonstrates them experimentally. The effect on the numbers of active and quiescent channels is demonstrated in Table IV.

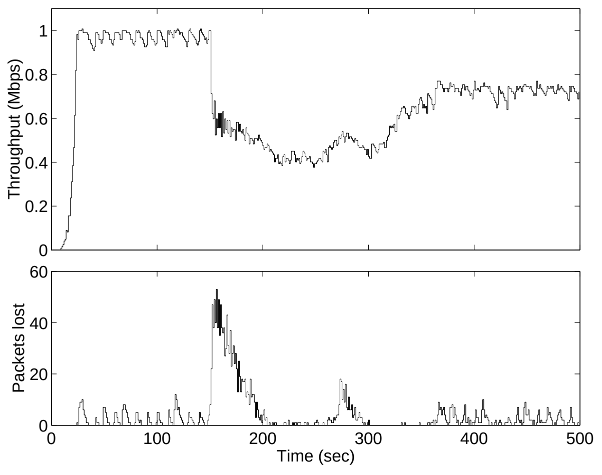
Repeating all of the experiments of the previous subsections while varying these parameters would be a large computational effort, so we primarily use the experiment of Section VII-A.3 to demonstrate the effects of the parameters. This experiment shows slow start, steady-state bandwidth utilization, and rate reduction in the face of competing traffic.

D.1 Varying P

The rate decrease factor P enters many computations at both the sender and the receiver. The advantages of decreasing P are to increase the range of reception rates for a fixed number of active wave channels N (or conversely to minimize N for a given dynamic range SR_b/BCR_b) and to be able to increase and decrease the reception rate more quickly. However, decreasing P also makes the saw-toothed

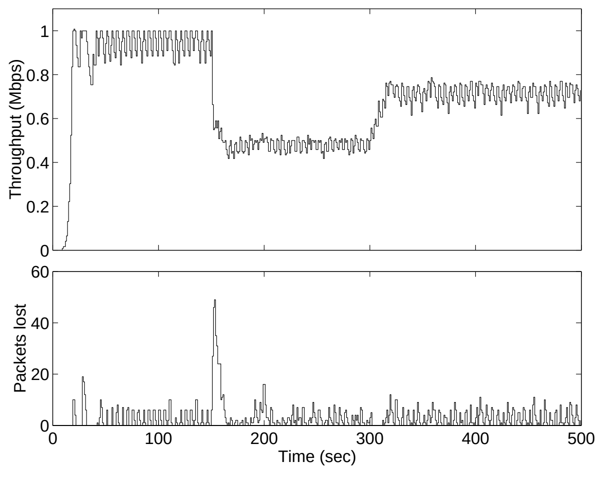


(a) $\mathtt{P} = 0.5$

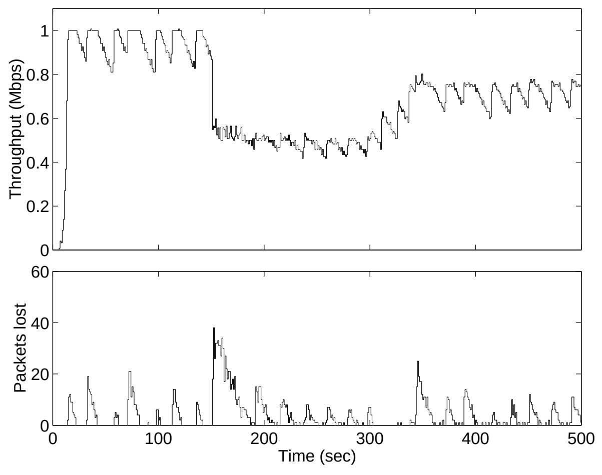


(b) $\mathtt{P} = 0.875$

Fig. 35. Experiments to illustrate the effect of varying `P`. (Compare to Fig. 27 for the default value $\mathtt{P} = 0.75$.)



(a) $\mathtt{TSD} = 5$



(b) $\mathtt{TSD} = 20$

Fig. 36. Experiments to illustrate the effect of varying `TSD`. (Compare to Fig. 27 for the default value $\mathtt{TSD} = 10$.)

nature of the reception rate more pronounced. This decreases the bandwidth utilization (see Eq. (34)) and the increased burstiness may have a negative impact on competing traffic.

Fig. 35 shows the result of repeating the experiment depicted in Fig. 27 with $\mathtt{P} = 0.5$ and $\mathtt{P} = 0.875$. The experiments confirm the expected behavior. A smaller value of `P` allows the subscribed rate to vary more quickly; faster increases are apparent in slow start and when the competing flow reduces its rate, and a faster decrease leads to far less packet loss when the competing traffic is introduced. On the other hand, a larger value of `P` makes the steady-state reception rate much smoother and increases the bandwidth utilization.

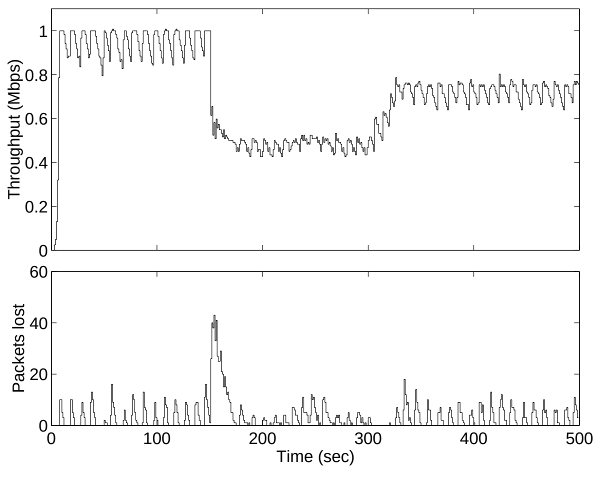
D.2 Varying `TSD`

A key motivation for improving upon FLID-DL was to reduce the frequencies of joins and leaves. Recall that on average, a WEBRC receiver issues one IGMP join and one IGMP leave each `TSD` seconds. The choice of `TSD` is dictated primarily by the frequency of joins and leaves that is considered reasonable. With current router technology, $\mathtt{TSD} = 10$ apparently does not cause an unreasonable amount of IGMP and PIM SM message processing. If IGMP and PIM message processing burden were not an issue, the choice of `TSD` would be influenced most by the number of quiescent channels and the desired smoothness of the reception rate.

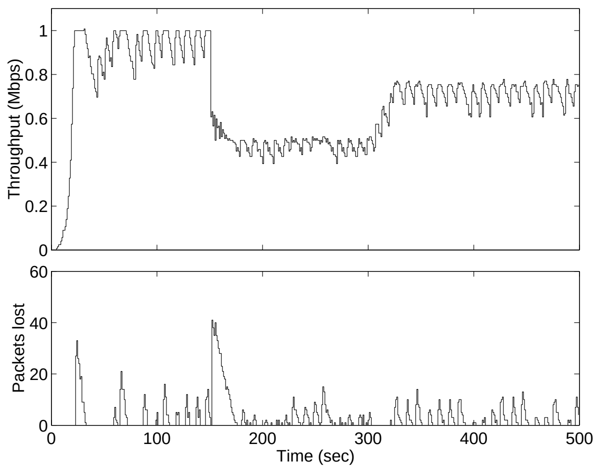
Fig. 36 shows the effect of varying `TSD`. Slow start is unaffected by `TSD`: rate increases are at each epoch boundary (unless `JOINING` is true) and the increase factors depend only on `P`. Rate decreases are potentially sharper when `TSD` is small, as revealed by the number of packet losses when the competing traffic is introduced at time 150. Finally, while `TSD` does not affect the saw-tooth shape of the rates, having a smaller value for `TSD` compresses the saw-tooth shape in time and hence leads to less variation in buffer occupancies.

D.3 Varying `EL`

The epoch length `EL` is the time granularity at which a WEBRC receiver joins waves and is also the interval at which the many receiver computations are made. Thus, the trade-off in the selection of `EL` is to balance the compu-



(a) $\texttt{EL} = 0.25$



(b) $\texttt{EL} = 1.0$

Fig. 37. Experiments to illustrate the effect of varying EL. (Compare to Fig. 27 for the default value $\texttt{EL} = 0.5$.)

tational burden of the epoch boundary computations with the granularity of the rate control.

Varying EL within a reasonable range has little effect on performance. Fig. 37 shows the effect of increasing or decreasing EL by a factor of two from the default value 0.5. The only discernible difference is in slow start. Since the MRTT in these experiments is relatively small (0.05 sec), joining is very rarely inhibited by the join timer. Hence, the rate increase in slow start is by one channel per epoch and the time to reach any particular rate is inversely proportional to EL. Very small values of EL are unjustified because the default value is sufficient to have rate control within a factor of $\texttt{P}^{\texttt{EL}/\texttt{TSD}} \approx 0.986$.

VIII. 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 [13]. 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.

Additional testing and tuning of WEBRC needs to be done before it can be adopted as an Internet standard. For example, extensive performance tests need to be done using dummynet, and then performance data needs to be collected from real-world deployment.

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 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.

Appendix

Averaging of Variables with Unknown Variance

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 [6], [8] 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 sequence 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 ith estimate a_i:

$$\begin{aligned} \omega_{i-1} &= \alpha \cdot a_{i-1}^2 / v_{i-1} \\ a_i &= (1-\omega_{i-1}) \cdot a_{i-1} + \omega_{i-1} \cdot m_i \\ v_i &= (1-\omega_{i-1}) \cdot v_{i-1} + \omega_{i-1} \cdot m_i^2 \end{aligned}$$

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 where the m_is are MRTT measurements to produce the average value called `ARTT`. 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 reciprocal of the average of X. Let $\delta \in (0, 1)$ be a smoothing constant. Set $Z_0 = 0$ and do the following to produce the ith estimate Z_i. If $m_i = 0$, then $Z_i = Z_{i-1} + \delta$; else $m_i = 1$ and $Z_i = (1-\delta) \cdot (Z_{i-1} + \delta)$. 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.*, for small p essentially the reciprocal of p.

This estimator can be directly applied to estimate the reciprocal 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. (Although TFRC uses the weighting strategy summarized on page 25, it is stated in [6] that EWMA works reasonably well.)
- In the simple TCP equation, the reception rate is proportional to the reciprocal of the square root of the loss probability. A loose translation of the above estimate into a direct estimate of this quantity yields the following. Let $\delta' = \delta/2$. Set $G_0 = 0$. Do the following to produce the ith 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 reciprocal 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 Z_i is applied to this time-smoothed average to produce the estimate of the reciprocal of the average loss probability, similar to TFRC.

Acknowledgments

Gavin Horn and Simon Skaria were involved in early phases of the design and simulation of WEBRC. Several recent refinements have been inspired by feedback from Robert Chapman, Armin Haken, and Diane Hernek. We also thank John Byers and the anonymous SIGCOMM 2002 reviewers for their extensive and helpful comments.

References

[1] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver. FLID-DL: Congestion control for layered multicast. In *Proc. 2nd Int. Workshop Netw. Group Comm.*, pages 71–81, Stanford, CA, November 2000.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM*, pages 56–67, Vancouver, September 1998.

[3] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for Monte Carlo estimation. *SIAM J. Comput.*, 29(5):1484–1496, April 2000.

[4] R. Denda. The fairness challenge in computer networks. Technical report, Univ. Mannheim, June 2000.

[5] 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, August 1999.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications: the extended version. Technical Report TR-00-003, Int. Comp. Sci. Instit., Berkeley, CA, March 2000.

[7] V. K Goyal. Constant-rate server output in WEBRC. Technical Report DF2002-03-001, Digital Fountain, March 2002. Available on-line at `http://www.digitalfountain.com/technology/`.

[8] M. Handley, J. Padhye, S. Floyd, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification. IETF Transport Area Working Group Internet-Draft, April 2002. Available on-line at `http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-tfrc-04.txt`. Work in progress. Expires October 2002.

[9] 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.

[10] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. Asynchronous layered coding protocol instantiation. IETF Reliable Multicast Transport Working Group Internet-Draft, April 2002. Available on-line at `http://www.ietf.org/internet-drafts/draft-ietf-rmt-pi-alc-08.txt`. Work in progress. Expires October 2002.

[11] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. Layered coding transport building block. IETF Reliable Multicast Transport Working Group Internet-Draft, February 2002. Available on-line at `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-lct-04.txt`. Work in progress. Expires August 2002.

[12] M. Luby and V. K Goyal. Wave and equation based rate control building block. IETF Reliable Multicast Transport Working Group Internet-Draft, March 2002. Document `draft-ietf-rmt-bb-webrc-01.txt` superceded by [13].

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

[14] M. Luby, V. K Goyal, and S. Skaria. Wave and equation based rate control: A massively scalable receiver driven congestion control protocol. IETF Reliable Multicast Transport Working Group Internet-Draft, October 2001. Document `draft-ietf-rmt-bb-webrc-00.txt` superceded by [12].

[15] M. Luby, V. K Goyal, S. Skaria, and G. B. Horn. Wave and equation based rate control using multicast round trip time. *Comp. Comm. Rev.*, 32(4):191–214, October 2002. (This issue is Proc. ACM SIGCOMM 2002.).

[16] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. Forward error correction building block. IETF Reliable Multicast Transport Working Group Internet-Draft, February 2002. Available on-line at `http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-fec-06.txt`. Work in progress. Expires August 2002.

[17] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. The use of forward error correction in reliable multicast. IETF Reliable Multicast Transport Working Group Internet-Draft, February 2002. Available on-line at `http://www.ietf.org/internet-drafts/draft-ietf-rmt-info-fec-02.txt`. Work in progress. Expires August 2002.

[18] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Comp. Comm. Rev.*, 27(3), July 1997.

[19] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proc. ACM SIGCOMM*, pages 117–130, Stanford, CA, August 1996.

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

[21] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM*, pages 303–314, Vancouver, September 1998.

[22] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. In *Proc. 9th Int. Workshop Netw. Operating Syst. Supp. for Dig. Aud. Vid. (NOSSDAV)*, Basking Ridge, NJ, June 1999.

[23] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Proc. IEEE INFOCOM*, volume 3, pages 1337–1345, New York, March 1999.

[24] L. Rizzo. pgmcc: A tcp-friendly single-rate multicast congestion control scheme. In *Proc. ACM SIGCOMM*, pages 17–28, Stockholm, Sweden, August 2000.

[25] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proc. 8th Int. Workshop Netw. Operating Syst. Supp. for Dig. Aud. Vid. (NOSSDAV)*, Cambridge, UK, July 1998.

[26] 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, March–April 1998.

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

[28] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *Proc. ACM SIGCOMM*, pages 275–286, San Diego, CA, August 2001.