Internet Engineering Task Force RMT WG
INTERNET-DRAFT
draft-ietf-rmt-bb-webrc-04.ps
M. Luby/Digital Fountain
V. K Goyal/Digital Fountain
9 December 2002
Expires: June 2003

Wave and Equation Based Rate Control building block

Status of this Document

This document is a product of the IETF RMT WG. Comments should be addressed to the authors, or the WG's mailing list at rmt@lbl.gov.

Abstract

Wave and Equation Based Rate Control provides rate and congestion control for data delivery. Wave and Equation Based Rate Control is specifically designed to support protocols using IP multicast. It provides multiple-rate, congestion-controlled delivery to receivers, i.e., different receivers joined to the same session may be receiving packets at different rates depending on the bandwidths of their individual connections to the sender and on competing traffic along these connections. Wave and Equation Based Rate Control requires no feedback from receivers to the sender, i.e., it is a completely receiver-driven congestion control protocol. Thus, it is designed to scale to potentially massive numbers of receivers attached to a session from a single sender. Furthermore, because 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.

Table of Contents

1. Introduction

This document specifies Wave and Equation Based Rate Control (WEBRC). WEBRC is a congestion control building block that is designed to be massively scalable when used with the IP multicast network service. WEBRC is also suitable as the basis for unicast congestion control, but this is outside the scope of this document. WEBRC is designed to compete fairly with TCP and similar congestion-controlled sessions. WEBRC can be used as a congestion control protocol for any type of data delivery, including reliable content delivery and streaming delivery.

WEBRC is a receiver-driven congestion control protocol in the spirit of [3] and [18]. This means that all measurements and decisions to raise or lower the reception rate are made by each individual receiver, and these decisions are acted upon by sending join and leave messages for channels to the network. A receiver using WEBRC adjusts its reception rate without regard for other concerns such as reliability. This is different from TCP, where the congestion control protocol and the reliability protocol are intricately interwoven.

WEBRC takes the same basic equation-based approach as TFRC [7]. In particular, each WEBRC receiver measures parameters that are plugged into a TCP-like equation to compute the receiver target reception rate, and adjusts its reception rate up and down to closely approximate the target reception rate. The sender sends packets to multiple channels; one channel is called the base channel and the remaining channels are called wave channels. Each wave channel follows the same pattern of packet rate transmission spread out over equally spaced intervals of time. The pattern of each wave is that it starts at a high rate that decreases gradually and continually over a long interval of time. (Picture an infinite sequence of waves.) The receiver increases its reception rate by joining the next wave channel earlier in the descent of the wave than it joined the previous wave channel, and the receiver decreases its reception rate by joining the next wave channel later in the descent of the wave than it joined the previous wave channel.

The wave channels are ordered at each point in time from a lowest layer to a highest layer. At each point in time, the lowest layer is the wave channel that has the smallest rate among all active wave channels and the highest layer is the wave channel that has the highest rate. Because waves are dynamically becoming active and quiescent over time, the designation of which wave channel is at which layer changes dynamically over time. In addition to being joined to the base channel, at each point in time a receiver is joined to a consecutive set of layers starting at the lowest layer and proceeding towards the highest.

WEBRC introduces a natural notion of a multicast round-trip time (MRTT). An MRTT is measured individually by each receiver and averaged 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 the WEBRC equation-based rate control protocol "TCP-friendly." The use of the MRTT also helps to coordinate and equalize the reception rates of proximate receivers joined to a session behind a bottleneck link. This implies that packets for the session that flow through the bottleneck link are on average sent to almost all downstream receivers, and thus the efficiencies of multicast are realized. Furthermore, WEBRC is designed to be massively scalable in the sense that the sender is insensitive to the number of receivers joined to a multicast session.

WEBRC is designed for applications that use a fixed packet size and vary their packet reception rates in response to congestion. WEBRC is designed to be reasonably fair when competing for bandwidth with TCP flows, where a flow is "reasonably fair" if its reception rate is generally within a factor of two of the reception rate of a TCP flow under the same conditions. However

WEBRC has a much lower variation of throughput over time compared to TCP, which makes it more suitable for applications such as telephony or streaming media where a relatively smooth rate is of importance. The penalty of having smoother throughput than TCP while competing fairly for bandwidth is that WEBRC responds more slowly than TCP to changes in available bandwidth.

The receiver measures and performs the calculation of congestion control parameters (e.g., the average loss probability, the average MRTT) and makes decisions on how to increase or decrease its rate based on these parameters. The receiver-based approach is well suited to an application where the sender is handling many concurrent connections and therefore WEBRC is suitable as a building block for multicast congestion control.

The paper [15] and technical report [14] provide much of the rationale and intuition for the WEBRC design and describe some preliminary simulations.

This document describes a building block as defined in RFC3048 [19]. This document describes a congestion control building block that conforms to RFC2357 [16]. This document is a product of the IETF RMT WG and follows the general guidelines provided in RFC3269 [9]. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [2].

2. Rationale

WEBRC provides congestion control for massively scalable protocols using the IP multicast network service. The congestion control that WEBRC provides is common to a variety of applications, including reliable content delivery and streaming applications.

WEBRC is designed to provide congestion control for all packets that are sent to a session. A session comprises multiple channels originating at a single sender that are used for some period of time to carry packets pertaining to the transmission of one or more objects that can be of interest to receivers. The logic behind defining a session as originating from a single sender is that this is the right granularity to regulate packet traffic via congestion control. The rationale for providing congestion control that uses multiple channels within the same session is that this allows the data on the channels to be layered, which in turn allows each receiver to control its reception rate by joining and leaving channels during its participation in the session. There are advantages to layered data for streaming, where the most important data can be sent to the lower layers and incrementally valuable data to the higher layers. For reliable content delivery, as described in [12], an application can send in packets encoded data generated from an object in such a way that the arrival of enough packets by a receiver is sufficient to reliably reconstruct the original object. A primary advantage of WEBRC is that each receiver controls it reception rate independent of other receivers. Thus, for example, a receiver with a slow connection to the sender does not slow down the receivers with faster connections.

There are coding techniques that provide massively scalable reliability and asynchronous delivery which are compatible with WEBRC, e.g., as described in [10]. When combined the result is a massively scalable, reliable, asynchronous content delivery protocol that is network friendly. WEBRC also provides congestion control that is suitable for streaming applications.

WEBRC avoids using techniques that are not massively scalable. For example, WEBRC does not provide any mechanisms for sending information from receivers to senders, although this does not rule out protocols that both use WEBRC and that send information from receivers to senders.

WEBRC provides congestion control that can be tuned for different applications that may have differing application requirements. For example, a content delivery protocol may aggressively strive to use all available bandwidth between receivers and the sender, and thus to maintain fairness it must drastically reduce its rate when there is competing traffic. On the other hand, a streaming delivery protocol may strive to maintain a constant rate instead of trying to use all available bandwidth, and thus it may not reduce its rate as fast when there is competing traffic.

WEBRC does not provide any support beyond congestion control, and thus WEBRC is to be combined with other building blocks to provide a complete protocol instantiation. For example, WEBRC does not provide any means that can be used to identify which session each received packet belongs to. As another example, WEBRC does not provide support for identifying which object each packet is carrying information about.

3. Functionality

A WEBRC session comprises a logically related set of channels originating from a single sender that are used for some period of time to carry data packets with a header carrying WEBRC Congestion Control Information. When packets are received, they are first checked to see that they belong to the appropriate session before WEBRC is applied. A session label defined by a protocol instantiation may be carried in each packet to identify to which session the packet belongs. For example, if LCT [11] is being used with the session, then the sender IP address together with the Transport Session Identifier supported by LCT would be used to determine which session a received packet belongs to. The particular details of how this filtering is performed it outside the scope of this document. In the remainder of this document, references to channels are always within the scope of a single session.

A channel can be uniquely identified at the network layer by a (sender IP address, multicast group address) pair, and this is the address to which the receiver sends messages to join and leave the channel. The channels used by a WEBRC session are mapped uniquely to consecutive channel numbers. In each packet sent to a channel, the channel number that corresponds to the channel is carried in the WEBRC Congestion Control Information. A WEBRC receiver uses the channel number to determine which channel within a session a packet is received from.

At the sender, time is partitioned into time slots, each of duration TSD seconds. There are a fixed number T of time slot indices associated with a session. As time progresses, the current time slot index increases by one modulo T each TSD seconds. The current time slot index CTSI is carried in the WEBRC Congestion Control Information. This allows receivers to perform very coarse-grained synchronization within a session.

WEBRC congestion control is achieved by having the sender send packets associated with a given session to several different channels. Individual receivers dynamically join and leave these channels according to the network congestion they experience. These congestion control adjustments are performed at each receiver independently of all other receivers, without any impact on the sender. A packet sequence number is carried in the WEBRC Congestion Control Information. The packet sequence numbers are consecutively numbered per channel and are used by receivers to measure packet loss.

The channels associated with a session consist of one base channel and T wave channels. The packet rate for each channel varies over time. For the base channel, packets are sent to the channel at a low rate BCR_P at the beginning of a time slot and this rate gradually decreases to P*BCR_P at the end of the time slot, where P < 1 is a constant defined later. This pattern for the base channel repeats over each time slot. For each wave channel i, packets are sent to channel i at a rate that first increases very quickly to a high rate and then decreases over time by a fixed fraction P per time slot until a rate of BCR_P is reached at the end of time slot i. Then, for a period of time 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 T*TSD seconds. Thus, the wave channels are going through the same cyclic pattern of packet rate transmission spaced out evenly by TSD seconds.

Before joining a session, the receivers MUST obtain enough of the session description to start the session. This MUST include the relevant session parameters needed by a receiver to participate in the session and perform WEBRC congestion control. The session description is determined by the sender and is typically communicated to the receivers out of band. How receivers obtain the session description is outside the scope of this document.

When a receiver initiates a session, it first joins the base channel. The packets in the base channel help the receiver orient itself in terms of what the current time slot index is, which in turn allows the receiver to know the relative rates on the wave channels. The receiver remains joined to the base channel for the duration of its participation in the session.

At each point in time the active (non-quiescent) wave channels are ordered into layers, where the lowest layer is the active wave channel whose wave is nearest to completion and the highest layer is the active wave channel whose wave is furthest from completion. (This is almost the same as saying that the lowest layer has the lowest rate and the highest layer has the highest rate. The possible deviation from this is due to the optional non-exponential beginnings of the waves as described in [6].) Each time a wave channel becomes active, it is the highest layer. At the end of each time slot the lowest layer wave channel becomes quiescent, and thus all active wave channels move down a layer at this point in time. At each point in time a receiver is joined to the base channel and a consecutive set of layers starting with the lowest. Each time a receiver joins a wave channel it joins the lowest layer not yet joined. A receiver always leaves the lowest layer when it becomes quiescent.

After joining a session the receiver adjusts its rate upwards by joining wave channels in sequence, starting with the lowest layer and moving towards the highest. The rates on the active wave channels are decreasing with time, so the receiver adjusts its rate downwards simply by refraining from joining additional wave channels. Since the layer ordering among the channels changes dynamically over time depending on the current time slot index, it is important that the receiver continually monitor the current time slot index contained in received packets. The reception rate at the receiver is determined by how early each wave channel is joined by the receiver: the earlier the receiver joins a channel with respect to when its wave started, the higher the reception rate.

Once the receiver is joined to a wave channel, the receiver remains joined to the wave channel until the channel goes quiescent, at which point the receiver MUST leave the channel.

The way the receiver adjusts its reception rate is inspired by TFRC [7]. The receiver at all points in time maintains a target reception rate, and the receiver is allowed to join the next wave channel if after joining its anticipated reception rate from all the layers it is joined to would be at most its target reception rate. The target rate is continually updated based on a set of measured parameters. The primary parameters are an estimate LOSSP of the average loss probability and an estimate ARTT of the average multicast round-trip time.

3.1. Sender Operation

The sender operation is by design much simpler than the receiver operation.

3.1.1. Sender inputs and initialization

The primary input to the sender for the session is SR_b. SR_b is an upper bound to the sender transmission rate in bits per second at any point in time (with some reasonable granularity) in aggregate to all channels. Naturally, this is then also the maximum rate in bits per second that any receiver could receive data from the session at any point in time. It is RECOMMENDED that the sender transmission rate in aggregate to all channels be made constant as described in [6]. It is also RECOMMENDED that the session description indicate whether the aggregate transmission rate is constant, unless there is no ambiguity.

The secondary inputs to the sender are listed below. These inputs are secondary because their values will generally be fixed to default values that will not change, because they will be derived from SR_b, or because they are chosen based on non-WEBRC considerations.

- o LENP_B is the length of packets in bytes sent to the session. 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.
- o BCR_P is the transmission rate on the base channel at the beginning of a time slot in packets per second. The default value for BCR_P is 1.
- o TSD is the time slot duration measured in seconds. The RECOMMENDED value for TSD is 10.
- o QD is the minimum quiescent period duration measured in seconds. The RECOMMENDED value for QD is 300.
- o P is the multiplicative drop in every channel rate over each time slot. The default value for P is 0.75.
- o N is the duration in time slots for each wave. N is also the number of wave channels active at any time. (A wave channel is called active when it is not quiescent.) A sender may choose any value that allows it to produce waves that substantially follow the required exponential shape described in Section 3.1.2. A RECOMMENDED mechanism for relating N to SR_b, BCR_P and P is described in [6].

From these inputs the following fixed sender parameters can be derived as follows.

- o SR_P = SR_b/(8*LENP_B) is the sender transmission rate in packets per second.
- o BCR_b = 8*LENP_B*BCR_P is the rate of the base channel at the beginning of a time slot in bits per second.
- o L = ceil(BCR_P*TSD*(P-1)/log(P)) is the number of base channel packets sent in each time slot.
- o Q = ceil(QD/TSD) is the number of quiescent time slots per cycle for a wave channel.

- o $T = N + Q$ is the total number of time slots in a cycle. T is also the total number of wave channels.
- o For the base channel CN = T and for the wave channels CN = 0,1,...,T-1. The sender has the description of the channels assigned to the session and the mapping between the channels and the CNs.
- o $C = TSD*T$ is the total duration of a cycle in seconds.

3.1.2. Sending packets to the session

The sender keeps track of the current time slot index CTSI. The value of CTSI is incremented by 1 modulo T each TSD seconds. The value of CTSI is placed into each packet in the format described in Section 5. For each packet sent to the session, the sender also places the channel number CN of the channel into the packets in the format described in Section 5. Recall that CN = T for the base channel and CN = 0,1,...,T-1 for the wave channels.

For each packet sent to the session, the sender calculates a packet sequence number PSN and places it into the packet. The value of PSN is scoped by CN, and the value of PSN is consecutively increasing within each channel. Furthermore, for each wave channel, the last packet sent before the channel becomes quiescent must have the maximum possible PSN value. When the short format for Congestion Control Information is used (see Section 5.1), this implies that for any wave channel the last PSN value sent to the channel just before the channel becomes quiescent is $2^{16}-1 = 65\,535$. Similarly, when the long format for Congestion Control Information is used (see Section 5.2), the PSN for the final packet of any wave is $2^{32}-1 = 4\,294\,967\,295$. The PSN of the initial packet of a wave thus depends on TSD, P, BCR_P and SR_P. For the base channel, the first packet of each time slot has a PSN congruent to zero modulo L. Hence, instead of $2^{16} - 1$ or $2^{32} - 1$ being the highest PSN used (depending on the choice of short format or long format Congestion Control Information), the highest PSN is one less than the largest multiple of L that does not exceed 2^{16} (short format) or 2^{32} (long format). The format for the PSN within packets is described in Section 5.

The rate at which packets are sent to the base channel starts at BCR_P packets per second at the beginning of each time slot and decreases exponentially to P*BCR_P at the end of that time slot.

The packet rate for the wave channels is more complicated. Each wave channel carries a sequence of waves separated by quiescent periods. On each wave channel each wave is active during N time slots followed by a quiescent period of Q time slots. The waves on wave channel i end at the ends of time slots with CTSI i. Therefore wave channel i is active during time slots i-N+1 modulo T, i-N+2 modulo T, ..., i and is quiescent for time slots i+1 modulo T, i+2 modulo T, ..., i+Q modulo T. Wave channel i first becomes active within time slot i-N+1 modulo T at a point in time that may depend on the value of SR_b.

For a substantial fraction of the duration of a wave ending at the end of a wave, the packet rate MUST decrease exponentially by a factor of P per TSD seconds, with a rate of BCR_P at the end of the last active time slot. At the beginning of each wave, the rate MAY deviate from this exponential form so that the total sending rate in aggregate to all of the channels is constant. A RECOMMENDED design for the beginnings of waves to achieve this goal is described in [6].

3.2. Receiver Operation

The bulk of the complexity in WEBRC is in the receiver operation. For ease of explanation, suppose for the moment that during the reception there is no packet loss and packets are arriving at exactly the rate at which they were sent. The sender transmission rate to the channels is designed so that the receiver reception rate behaves as follows.

Upon entering a session, the receiver immediately joins the base channel. When the receiver wants to increase its rate, it joins consecutive layers starting with the lowest and moving towards the highest. (Recall that the designations of lowest to highest change as waves become active and quiescent.) When the receiver wants to maintain its current reception rate and it is already joined to the lowest NWC layers, if the receiver joins channel i-1+NWC modulo T sometime during time slot i then the receiver joins channel i+NWC modulo T TSD seconds later in time slot i+1. When the lowest layer becomes quiescent the receiver leaves the channel.

Suppose the receiver wants to decrease its rate till it is joined to just the base channel. Assume that a receiver is joined to the lowest NWC < N-2 layers at the beginning of time slot i, i.e., wave channels i, i+1 modulo T,..., i+NWC-1 modulo T. Then, the aggregate packet reception rate of the receiver over the next NWC time slots will behave as follows if the receiver does not join any wave channels during this time. At the beginning of time slot i the receiver reception rate is BCR_P*(1 + (1/P) + (1/P)^2 + ... + (1/P)^NWC). Then the receiver reception rate decreases by a factor of P over the duration of each time slot, and at the end of each time slot the reception rate decreases by an additive amount of P*BCR_P. At the end of time slot i+NWC-1 mod T, the receiver reception rate is BCR_P*(1+P), and at the beginning of time slot i+NWC mod T the receiver is joined only to the base channel and its reception rate is BCR_P.

3.2.1. Receiver inputs and initialization

Before joining a session the receiver MUST know the mapping between the CNs and the channels. Upon joining the session or shortly thereafter, it SHOULD have the values of LENP_B, BCR_P, TSD, P, N, L, Q and T. Some of these values may be computed or measured once the receiver has joined the session. For example, the receiver MAY obtain LENP_B and T from the first packet received from the base channel, and the receiver MAY 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 that do not change from session to session, and the value of N MAY be computed as T-Q. The receiver SHOULD know whether the sender is employing a technique to produce constant aggregate rate as described in [6].

When a receiver first joins a session, it MUST first join just the base channel and start receiving packets to determine the current time slot index. If during the course of the session the receiver continually loses a high fraction of the packets from the base channel even when the receiver is only joined to the base channel, the receiver SHOULD leave the session.

The receiver MAY also have other individually set parameters that may be used to determine its behavior. One such parameter is MRR_b:

o MRR_b is the maximum receiver reception rate in bits per second. This may be used to determine the maximum reception rate this receiver is willing to reach. Thus, the maximum reception rate that the receiver can possibly achieve in the session is the minimum of SR_b and MRR_b. A recommended value of MRR_b for a receiver 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., MRR_P = MRR_b/(8*LENP_B).

3.2.2. Receiver measurements and calculations

As outlined in the introduction, the way a receiver adjusts its reception rate is inspired by TFRC [7]. The receiver at all points in time maintains a target reception rate, and the receiver is allowed to join the next wave channel if joining would increase its reception rate to at most its target reception rate. The target rate is continually updated based on a set of measured parameters.

Two primary parameters are the estimate LOSSP of the average loss probability and the estimate ARTT of the average MRTT. Both LOSSP and ARTT are moving averages of measurements based on discrete events. For many of the other estimates calculated by WEBRC, using an exponentially weighted moving average (EWMA) with a fixed averaging fraction is sufficient. However, the calculations of LOSSP and ARTT require a more general and sophisticated filtering approach.

3.2.2.1. Average loss probability

The design of TFRC [7] 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 with 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.

The estimate LOSSP of the average loss probability of the receiver is maintained in a manner somewhat similar to that described in TFRC [7]. The WEBRC receiver estimates the inverse of the average loss probability by applying two EWMA filters to the packet reception measurements, a time-based filter with smoothing constant 0 < Nu < 1 and a loss-based filter with smoothing constant 0 < Delta < 1. The recommended values for the smoothing constants are Nu = 0.3 and Delta = 0.3. The reason for the time-based 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, the time-based filter 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. Intuitively, the time-based filter averages packet reception events such that the events are smoothed out over an interval of time proportional to TSD/Nu seconds. The loss-based filter, similar to what is suggested in TFRC, is applied to the output of the time-based filter to produce the estimate of the inverse of the average loss probability. Intuitively, the loss-based filter averages loss events such that each loss event is averaged in with weight Delta.

As described later, LOSSP is initialized at the end of slow start and occasionally reset due to other events. Let W and X be counts of packets, let Y be a count of loss events and let Z be the long-term estimate of the inverse of the average loss probability. Whenever the value of LOSSP is initialized or reset, the values of W, X, Y and Z are also initialized or reset.

Recall that TSD is the duration of a time slot. The epoch length EL is the duration of time between decisions to adjust the reception rate. Generally EL is much smaller than TSD, and the RECOMMENDED values are EL = 0.5 seconds and TSD = 10 seconds.

Define G = Nu*EL/TSD as the amount of time-based smoothing to perform at the end of each epoch. The update rules for W, Y, Z and LOSSP are the following:

- o At the end of each epoch, adjust X, Y and Z and compute LOSSP as follows:
  - Z = Z*(1-Delta)^(G*Y) + X/(Y+1)*(1-(1-Delta)^(Y+1))
  - X = X*(1-G)
  - Y = Y*(1-G)
  - Z1 = Z*(1-Delta)^Y + X/(Y+1)*(1-(1-Delta)^(Y+1))
  - Z2 = Z*(1-Delta)^(Y+1) + (X+W+1)/(Y+2)*(1-(1-Delta)^(Y+2))
  - LOSSP = 1/max{Z1,Z2,1}

- o For each packet event (whether it is a received packet or a lost packet), W = W + 1

- o At the beginning of each loss event, update X, Y and Z as follows:
  - X = X + W
  - W = 0
  - Y = Y + 1

The intuition behind these update rules is the following. If just loss-filtering were used to update Z, then Z would be decreased by a multiplicative amount 1 - Delta for each loss event and Z would be increased by an additive amount Delta for each packet. To smooth out loss events over more than one time slot, these adjustments are filtered into Z over time, at the rate of a fraction G at the end of each epoch. Thus, the variables X and Y are counts of the portions of the packets and loss events, respectively, that have not yet been filtered into the long-term memory Z. W is the count of packets since the last loss event started. This explains why W is increased by one for each packet and Y is increased by one for each loss event. At the end of each epoch a fraction G of both X and Y are filtered into Z according to the loss-filter rule described above, and then the same fraction G is removed from both X and Y to account for the fact that this portion has been filtered into Z. The LOSSP calculation combines the short-term history (X,Y) with the long-term history Z and also allows the arrivals since the last loss W to have some influence. The value of Z2 is what Z1 would become were the next packet to be lost.

To reset the loss calculation to a value LOSSP = a, the state variables are set as follows:

W = 0
X = 0
Y = 0
Z = 1/a

3.2.2.2. Average round-trip time

The receiver maintains an average round-trip time, ARTT, as a measurement-based filter of MRTT measurements using a smoothing constant 0 < Alpha < 1. The RECOMMENDED value for Alpha is 0.25.

Each time the receiver joins a channel (either the base channel upon entering a session or wave channels continually), it makes a measurement of the multicast round-trip time MRTT as follows. Let V be an auxiliary variable that is used that keep track of the average of the square of the MRTT measurements. When the receiver sends the join for the channel it records the current time JoinTime and sets a Boolean variable JOINING to true. When the first packet is received from the channel the receiver records the current time FirstTime and resets the value of JOINING to false. If it is the base channel that has been joined, ARTT is set to FirstTime-JoinTime and V is set to ARTT*ARTT. Otherwise, the value of MRTT is set to (FirstTime - JoinTime) - log(1/P)/2/(1-P)/BCR_P * P^NWC. (Note that this value can be negative.) Then, ARTT is updated as follows. Let Omega = Alpha*ARTT*ARTT/V, and at the Kth MRTT measurement let Rho = Omega/(Omega+(1-Omega)*(1-(1-Omega)^K)). (Note that as K grows Rho approaches Omega.) Then, V is updated to (1-Rho)*V+Rho*MRTT*MRTT and ARTT is updated to max{P*ARTT,(1-Rho)*ARTT+Rho*MRTT}.

Usually ARTT is updated to the second term in the max, and in this case ARTT is the EWMA of the previous value of ARTT and the new MRTT, with a weighting on the new MRTT that as K grows is proportional to the square of the previous ARTT divided by the previous average V of the square of the MRTT. Thus, if there is not much variance in the previous MRTTs relative to the square of their average then the new MRTT will be filtered into ARTT with a high weight, whereas if there is a lot of variance in the previous MRTTs relative to the square of their average then the new MRTT will be filtered into ARTT with a low weight. The intuitive rationale for this is that in general the number of measurements needed to compute a meaningful average for a random variable is proportional to its variance divided by the square of its average; see, e.g., [4]. By making the weight factor depend on previous measurements in this way, the appropriate weight to use to average the new MRTT into the ARTT self-adjusts automatically to the variability in the measurements.

3.2.2.3. Rate Equation

The receiver calculates the reception rate REQN based on the TCP equation as follows: REQN = 1/(ARTT*sqrt{LOSSP}(0.816 + 7.35*LOSSP*(1+32*LOSSP^2))). This equation comes from TFRC [7].

3.2.2.4. Epochs

The receiver makes decisions on whether or not to join another wave channel at equally spaced units of time called epochs. The duration of an epoch in seconds, EL, is set to be a small fraction of TSD, so that decisions to join a channel can be made at a much finer granularity than TSD. A standard setting is EL = TSD/20. Thus, with the recommended setting of TSD = 10, it is RECOMMENDED that EL = 0.5.

3.2.2.5. Average reception rate

There are two averaged reception rates maintained by the receiver: TRR_P, the true reception rate, and ARR_P, the anticipated reception rate. These are used for different purposes and thus are calculated quite differently. Recommended values for the filtering weights Beta and Zeta are provided at the end of this subsection.

In start-up mode, the true reception rate TRR_P is used to ensure that the receiver does not increase its reception rate too quickly above its current reception rate. In the transition from start-up mode to normal operation and in normal operation, TRR_P is used in setting the slow start rate. TRR_P is calculated based on the measurement of RR_P, where RR_P is the receiver reception rate in packets per second measured at the beginning of an epoch averaged over the epoch that just ended. TRR_P is initialized to BCR_P + k*log(P)/TSD when the first base channel packet of the session arrives, where k is the PSN of the packet reduced modulo L. TRR_P is updated to (1-Zeta)*TRR_P + Zeta*RR_P at the beginning of each epoch after RR_P is measured for the previous epoch.

The anticipated reception rate ARR_P is the receiver's estimate of the total instantaneous rate of the currently joined channels. It is used to compare against the target rate to decide whether or not the receiver should increase its reception rate by joining the next higher unjoined layer. ARR_P is calculated based on a measurement IRR_P and on the number of joined wave channels NWC. The ideal reception rate IRR_P is the reception rate in packets per second including both received and lost packets; like RR_P, it is measured at the beginning of the epoch and averaged over the previous epoch. ARR_P, IRR_P and NWC are updated as follows:

- o NWC is initialized to 0.
- o When the first base channel packet arrives, ARR_P is set to BCR_P + k*log(P)/TSD, where k is the PSN of the packet reduced modulo L.
- o At the beginning of each epoch, IRR_P is measured over the previous epoch and then ARR_P is updated to P^(EL/TSD)*(1-Beta)*ARR_P + Beta*IRR_P.
- o When a join is made to the next higher unjoined layer, NWC is updated to NWC+1 and then ARR_P is multiplicatively increased by the factor ((1/P)^(NWC+1)-1)/((1/P)^NWC-1). (Joins happen at epoch boundaries; this adjustment is in addition to the adjustment above.)
- o Each time a next time slot index is detected, ARR_P is additively increased by (1-P)*BCR_P to account for the change in rate on the base channel. In addition, the bottom layer in the previous time slot has just gone quiescent and thus a message to leave this layer has been sent, ARR_P is additively decreased by BCR_P and NWC is decremented by 1.

Consider for the moment what happens if Beta = 0 and ARR_P is an accurate estimate of the total rate of the joined channels. The adjustments to ARR_P upon joining and leaving wave channels, with the passage of epochs, and with the detection of time slot changes will then cause ARR_P to remain an accurate estimate. In practice, Beta MUST be positive; allowing an influence of IRR_P prevents ARR_P from drifting away from being an accurate estimate of the total joined rate.

The motivation for separate estimates TRR_P and ARR_P is as follows. ARR_P is needed for comparison with the TFRC-inspired target rate because there is no lag before it reflects the potential rate increase resulting from joining the next higher layer and because it measures the total possible impact on the network since it also includes lost packets. TRR_P is needed because it reflects the rate of data arriving at the receiver and this is used to ensure that there is not a large gap between the joined rate and the receiving rate.

The recommended values for Beta and Zeta depend on whether the receiver is in start-up mode (SSR_P = infinity). In start-up mode, it is RECOMMENDED that Beta = (1 - P^(0.25))/2 and Zeta = sqrt(P)/(1 + sqrt(P)). In normal operation, it is RECOMMENDED that Beta = 1 - (P/(1+P))^(EL/TSD) and Zeta = 2*EL/(4+TSD).

3.2.2.6. Slow start

WEBRC uses a slow start mechanism to quickly ramp up its rate at both the beginning of the session and in the middle of a session when the rate drops precipitously. To enact this, the receiver maintains the following parameters:

- o SSMINR_P is the minimum allowed slow start threshold rate in packets per second. The recommended value for SSMINR_P is BCR_P*(1+1/P+1/P^2).
- o SSR_P is the slow start threshold rate in packets per second. It is adjusted at the beginning of loss events as described in Section 3.2.3.4. SSR_P is initialized to infinity and is first set to a finite value when the receiver leaves the initial start-up period as described below.

At the beginning of a session, the receiver cannot compute a meaningful target rate from its measurements. Thus, it uses SSR_P = infinity until one of the following events causes an end to this start-up mode:

- o A packet loss is detected. In this case the value of SSR_P is updated to max{SSMINR_P, P*TRR_P} as with the beginning of any other loss event.
- o A sharp increase in MRTT is detected. While SSR_P = infinity the receiver MUST compute, in the notation of Section 3.2.2.2, differences in successive measurements of (FirstTime-JoinTime) from successive waves and MUST set SSR_P to max{SSMINR_P, P*TRR_P} when a large increase in (FirstTime-JoinTime) is observed. It is RECOMMENDED that an increase in (FirstTime-JoinTime) be considered large if it exceeds (1-P^(NWC+1))/(P*log(P)) / ARR_P.
- o The maximum reception rate is reached. When SSR_P = infinity, if (P^(-NWC-2)-1)/(P^(-NWC-1)-1)*ARR_P exceeds MRR_P or SR_P, the receiver MUST set SSR_P to max{SSMINR_P, TRR_P}.
- o TRR_P is not increasing consistent with the last join of a wave channel. While SSR_P = infinity, it is RECOMMENDED that the receiver wait at least one full epoch after the first packet of a wave is received before joining the next wave. If the TRR_P after that full epoch is greatly below ARR_P the receiver SHOULD NOT join and SHOULD then set SSR_P to max{SSMINR_P, TRR_P}. It is RECOMMENDED that TRR_P be considered greatly below ARR_P if TRR_P < c * ARR_P - 2/EL, where c = Zeta + (1-Zeta)*(P^(-EL/TSD))*(Zeta + (1-Zeta)*sqrt(P)*(P^(-EL/TSD)))/g with g = (P^(-NWC-1)-1)/(P^(-NWC)-1).

In any of these four cases, the variables associated with LOSSP are reset to make REQN, calculated as in Section 3.2.2.3 with the current value of ARTT, equal TRR_P.

3.2.2.7. Target rate

In typical operation, SSR_P has a finite value and the target rate TRATE is computed as TRATE = min{max{SSR_P, REQN}, MRR_P}. When SSR_P = infinity, TRATE is computed as TRATE = min{4*TRR_P, MRR_P}.

3.2.3. Receiver events

There are various receiver events, some of which are triggered by the passing of time on the receiver, and others by events such as packet reception, detection of packet loss, reception of a first packet from a channel, and exceptional time-outs.

3.2.3.1. Packet reception

Most packet reception events require the receiver to merely register the reception for later calculation of RR_P and IRR_P (see Section 3.2.2.5) and increment W for later calculation of LOSSP (see Section 3.2.2.1). Additional actions, described in the following three subsections, are required if the packet is the first packet received in response to a join operation, the CTSI of the packet indicates a time slot change, or the CN and PSN of the packet indicate a packet loss.

3.2.3.2. First packet after join

When channel i is the most recently joined channel and the Boolean variable JOINING is true, the reception of a packet with PSN = i is a special event because it is the first packet received in response to the most recent join. MRTT is calculated and ARTT and V are updated as described in Section 3.2.2.2, and JOINING is set to false. The first received packet of the session furthermore necessitates initialization of ARR_P and TRR_P as described in Section 3.2.2.5.

3.2.3.3. Time slot change

This is an event that is triggered by the reception of a packet with a CTSI value that is one larger modulo T than the previous CTSI value. When a packet with a new CTSI = i is received, a leave is sent for the lowest layer in the previous time slot, i.e., wave channel i-1 modulo T, NWC is updated to NWC-1, and ARR_P is updated to ARR_P - P*BCR_P as described in Section 3.2.2.5. If the channel for which the leave is sent is also the most recently joined wave channel and JOINING is true, then JOINING is set to false.

It is possible due to packet reordering for some packets from the previous time slot to be received after packets from the current time slot. It is RECOMMENDED that measures be put into place to handle this situation appropriately, i.e., to not trigger a time slot change in this situation. One simple mechanism for this is as follows: Compute the difference i-j modulo T, where i is the CTSI of the received packet and j is the current CTSI of the receiver. A difference of zero is, of course, not a time slot change. In addition, a very large difference, for example a difference larger than T-Q/2, should also not trigger a time slot change.

3.2.3.4. Loss event

Each time the receiver detects a lost packet (based on the sequence numbers in the packets scoped by the channel number), the receiver records the start of a new loss event, and sets a Boolean variable LOSS_EVENT to true that will automatically reset to false after ARTT seconds. All subsequent packet loss for a period of ARTT seconds is considered as part of the same loss event. When a start of a loss event is detected, the value of SSR_P is updated to max{SSMINR_P, P*TRR_P}.

It is RECOMMENDED that the receiver account for simple misordering of packets without inferring a loss.

3.2.3.5. Epoch change

This is an event that is triggered by the passage of time at the receiver, which occurs each EL seconds. When this happens, TRR_P and ARR_P are computed as described in Section 3.2.2.5.

Immediately after these updates, a decision is made about whether to join the next higher layer as described in Section 3.2.3.6.

3.2.3.6. Join the next higher layer

At the beginning of each epoch, after updating the values of ARR_P and TRR_P as described in Section 3.2.2.5, the receiver decides whether or not to join the next higher layer as follows:

- o If the first base channel packet has not yet arrived the receiver does not join.
- o If there is a loss event in progress (LOSS_EVENT = true) the receiver does not join.
- o If a join of a channel is in progress (JOINING = true), the receiver does not join.
- o If NWC = N the receiver does not join.
- o If the receiver is employing the OPTIONAL rule described in Section 3.2.2.6, SSR_P = infinity, and a full epoch has not passed since the first packet arrival on the most recently joined wave channel then the receiver does not join.
- o If the receiver is employing the OPTIONAL rule described in Section 3.2.2.6, SSR_P = infinity, and a full epoch has passed since the first packet arrival on the most recently joined wave channel, then the receiver checks if TRR_P is greatly below ARR_P as described in Section 3.2.2.6. If TRR_P is greatly below ARR_P the receiver does not join.
- o The receiver calculates REQN as described in Section 3.2.2.3.
- o The receiver calculates TRATE as described in Section 3.2.2.7.
- o If the sender is not sending at constant aggregate rate and TRATE < ARR_P*((1/P)^{NWC+2}-1)/((1/P)^{NWC+1}-1), the receiver does not join. If the sender is sending at constant aggregate rate and neither TRATE >= ARR_P*((1/P)^{NWC+2}-1)/((1/P)^{NWC+1}-1) nor TRATE >= SR_P is true, the receiver does not join.
- o If the sender is producing constant aggregate rate and TRATE >= SR_P, the receiver joins the next wave channel. Otherwise if SSR_P is finite the receiver MAY apply one additional OPTIONAL check before deciding to join.

  It is RECOMMENDED that the receiver not join if the value of RR_P is not sufficiently lower than the maximum value of RR_P observed since the last join. It is RECOMMENDED that RR_P is sufficiently low to allow a join if RR_P <= max{RRmax-2/EL,P*RRmax}, where RRmax is the maximum measured RR_P since the last join.

  If the receiver does not join because RR_P is not sufficiently small then a value of LOSSP is calculated so as to make the value of the REQN equation given in Section 3.2.2.3 evaluate to ARR_P*((1/P)^(NWC+2)-1)/((1/P)^(NWC+1)-1) with respect to the current value of ARR_P. Then, the variables associated with LOSSP are reset based on this calculated value of LOSSP as described at the end of Section 3.2.2.1.

Suppose the receiver has decided to join and CTSI = i. The receiver joins the next higher wave channel, i.e., the wave channel with CN = i+NWC modulo T, increments NWC by 1, and then updates ARR to ARR_P*((1/P)^{NWC+1}-1)/((1/P)^NWC-1) as described in Section 3.2.2.5. The time of the join is recorded for use in updating ARTT as described in Section 3.2.2.2.

3.2.3.7. Join timeout

When no packet arrives in response to the join of channel for a long period of time, the join times out. The receiver sets JOINING to false, updates ARR to ARR_P*((1/P)^NWC-1)/((1/P)^{NWC+1}-1), and then decrements NWC by 1.

The RECOMMENDED threshold for a join timeout is max{2*V/ARTT,10*ARTT} seconds.

3.2.3.8. Exceptional timeouts

These are timeouts when the packet reception behavior is far from what it should be and these MUST trigger the receiver to leave the session. Exceptional timeouts include

- o No packets are received for a long period. A RECOMMENDED threshold is max{10,TSD} seconds.
- o There is no change in time slot index for a long period. A RECOMMENDED threshold is max{20,2*TSD} seconds.

4. Applicability Statement

WEBRC is intended to be a congestion control scheme that can be used in a complete protocol instantiation that delivers objects and streams (both reliable content delivery and streaming of multimedia information). WEBRC is most applicable for delivery of objects or streams of substantial length, i.e., objects or streams that range in length from hundreds of kilobytes to many gigabytes, and whose transfer time is on the order of tens of seconds or more.

4.1. Environmental Requirements and Considerations

WEBRC can be used with both multicast and unicast networks. However, the scope of this document is limited to multicast. WEBRC requires connectivity between a sender and receivers, but does not require connectivity from receivers to the sender.

WEBRC inherently works with all types of networks, including LANs, WANs, Intranets, the Internet, asymmetric networks, wireless networks, and satellite networks. Thus, the inherent raw scalability of WEBRC is unlimited. However, in some network environments varying reception rates to receivers may not be advantageous. For example, the network may have a dedicated fixed amount of bandwidth allocated to the session and there may be no effective way for receivers to dynamically vary the set of channels they are joined to, e.g., in a satellite network.

Receivers join and leave channels using the appropriate multicast join and leave messages. For IPv4 multicast, IGMP messages are used by receivers to join and leave channels. For IPv6, MLDv2 messages are used by receivers to join and leave channels. This is the only dependency of WEBRC on the IP version.

WEBRC requires receivers to be able to uniquely identify and demultiplex packets associated with a session in order to effectively perform congestion control over all packets associated with the session. How receivers achieve this is outside the scope of this document.

WEBRC is presumed to be used with an underlying network or transport service that is a “best effort'' service that does not guarantee packet reception, packet reception order, and which does not have any support for flow or congestion control. For example, the Any-Source Multicast (ASM) model of IP multicast as defined in RFC1112 [5] is such a best effort network service. While the basic service provided by RFC1112 is largely scalable, providing congestion control or reliability should be done carefully to avoid severe scalability limitations, especially in the presence of heterogeneous sets of receivers.

There are currently two models of multicast delivery, the Any-Source Multicast (ASM) model as defined in RFC1112 [5] and the Source-Specific Multicast (SSM) model as defined in [8]. WEBRC works with both multicast models, but in a slightly different way with somewhat different environmental concerns. When using ASM, a sender S sends packets to a multicast group G, and the WEBRC channel address consists of the pair (S,G), where S is the IP address of the sender and G is a multicast group address. When using SSM, a sender S sends packets to an SSM channel (S,G), and the WEBRC channel address coincides with the SSM channel address.

A sender can locally allocate unique SSM channel addresses, and this makes allocation of channel addresses easy with SSM. To allocate channel addresses using ASM, the sender must uniquely chose the ASM multicast group address across the scope of the group, and this makes allocation of WEBRC channel addresses more difficult with ASM. This is an issue for WEBRC because several channels are used per session.

WEBRC channels and SSM channels coincide, and thus the receiver will only receive packets sent to the requested WEBRC channel. With ASM, the receiver joins a channel by joining a multicast group G, and all packets sent to G, regardless of the sender, may be received by the receiver. Thus, SSM has compelling security advantages over ASM for prevention of denial of service attacks. In either case, receivers SHOULD use mechanisms to filter out packets from unwanted sources.

WEBRC assumes that the packet route between the sender and a particular receiver is the same for all channels associated with a session. For SSM this assumption is true because the multicast tree is a shortest path tree from each receiver to the sender and generally this path changes infrequently. For ASM there are some issues that if not properly considered may invalidate this assumption. With ASM, the packet route between the sender and receivers may initially be through the Rendezvous Point (RP) and then switch over to the shortest path to the sender as packets start flowing in a channel. The first issue is that the RP may not be the same for all channels associated with a session, and thus the first packets sent to the channels may follow a route that depends on the RP of the channel. This depends on the RP configuration for the sender. If the sender registers all channels associated with the session with the same RP then the assumption is true, but if the sender registers different channels with different RPs then the assumption may not be true. Thus, it is RECOMMENDED that the sender register all channels associated with a session with the same RP. Another issue is that when the channel switches over from the RP to the sender-based tree then the route to the receivers may vary within a channel. Furthermore, this may cause either the receipt of duplicate packets at receivers or loss of packets depending on the smoothness of the switchover. Thus, it is RECOMMENDED that the RP be placed as close as possible to the sender. The best location for the RP is that it be the first-hop router closest to the sender, in which case the path to the sender and the path to the RP is the same for each receiver and the problems mentioned above are eliminated. The consequences of this assumption not being true are that the receiver reaction to congestion may not be appropriate. Generally, the WEBRC receiver will act conservatively and reduce its reception rate too much if this assumption is not true, but there can be cases where the receivers will act inappropriately.

5. Packet Header Fields

Packets sent to a session using WEBRC MUST include Congestion Control Information fields as specified in this section. This document specifies short and long formats for the Congestion Control Information, and it is RECOMMENDED that protocol instantiations use one of these two formats. Other formats for the Congestion Control Information fields MAY be used by protocol instantiations, but all protocol instantiations are REQUIRED to use these fields in a format that is compatible with the interpretations of these fields. Thus, if a protocol does use a different format for the fields in the Congestion Control Information then it MUST specify the lengths and positions of these fields within the packet header.

All integer fields are carried in "big-endian" or "network order" format, that is, most significant byte (octet) first. All constants, unless otherwise specified, are expressed in base ten.

5.1. Short Format Congestion Control Information

The short format for the Congestion Control Information is shown in Fig. 1. The total length of the short format is 32-bits.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      CTSI     | Channel Number|     Packet Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 1 - Short format for Congestion Control Information

The function of each field in the Congestion Control Information is the following.

Current Time Slot Index (CTSI): 8 bits

CTSI indicates the index of the current time slot. This must be sent in each packet within the session. The Current Time Slot Index increases by one modulo T each TSD seconds at the sender, where T is the number of time slots associated with the session and TSD is the time slot duration. Note that T is also the number of wave channels associated with the session, and thus T MUST be at most 255.

Channel Number (CN): 8 bits

CN is the channel number that this packet belongs to. CN for the base channel is T, and the CNs for the wave channels are 0 through T-1. Thus, T+1 channels in total are used, and thus T MUST be at most 255.

Packet Sequence Number (PSN): 16 bits

The PSN of each packet is scoped by its CN value. The sequence numbers of consecutive packets sent to the base channel are numbered consecutively modulo 2^16. The same sequence of PSNs are used for each wave channel in each cycle. The sequence numbers of consecutive packets sent to a wave channel are numbered consecutively modulo 2^16 within each cycle, ending with the last packet sent to the channel before the channel goes quiescent with PSN = 2^16-1.

5.2. Long Format Congestion Control Information

The long format for the Congestion Control Information is shown in Fig. 2. The total length of the long format is 64-bits.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             CTSI              |        Channel Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Packet Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 2 - Long format for Congestion Control Information

The meaning of each field for the long format is the same as for the short format, the only difference is that each field is twice as long.

Current Time Slot Index (CTSI): 16 bits

CTSI indicates the index of the current time slot. This must be sent in each packet within the session. The Current Time Slot Index increases by one modulo T each TSD seconds at the sender, where T is the number of time slots associated with the session and TSD is the time slot duration. Note that T is also the number of wave channels associated with the session, and thus T MUST be at most 65 535.

Channel Number (CN): 16 bits

CN is the channel number that this packet belongs to. CN for the base channel is T, and the CNs for the wave channels are 0 through T-1. Thus, T+1 channels in total are used, and thus T MUST be at most 65 535.

Packet Sequence Number (PSN): 32 bits

The PSN of each packet is scoped by its CN value. The sequence numbers of consecutive packets sent to the base channel are numbered consecutively modulo 2^32. The same sequence of PSNs are used for each wave channel in each cycle. The sequence numbers of consecutive packets sent to a wave channel are numbered consecutively modulo 2^32 within each cycle, ending with the last packet sent to the channel before the channel goes quiescent with PSN = 2^32-1.

6. Requirements From Other Building Blocks

As described in RFC3048 [19], WEBRC is a building block that is intended to be used, in conjunction with other building blocks, to help specify a protocol instantiation.

WEBRC does not provide higher level session support, e.g., how receivers obtain the necessary session description and how the receivers demultiplex received packets based on their session. There is support provided by other building blocks that can be used in conjunction with WEBRC to provide some of this support. For example, LCT [11] can provide some of the higher level in-band session support that may be needed by receivers, and the WEBRC Congestion Control Information (CCI) required in each packet can be carried in the CCI field of the LCT header [11].

WEBRC does not provide any type of reliability, and in particular does not provide support for retransmission of loss packets. Reliability can be added by independent means, such as by the use of FEC codes as described in [12] and specified in the FEC building block [13].

7. Security Considerations

WEBRC can be subject to denial-of-service attacks by attackers that try to confuse the congestion control mechanism for receivers by injecting forged packets into the multicast stream. This attack most adversely affects network elements and receivers downstream of the attack, and much less significantly the rest of the network and other receivers. Because of this and because of the potential attacks due to the use of FEC described above, it is RECOMMENDED that Reverse Path Forwarding checks be enabled in all network routers and switches along the path from the sender to receivers to limit the possibility of a bad agent injecting forged packets into the multicast tree data path.

It is also RECOMMENDED that packet authentication be used to authenticate each packet immediately upon receipt before the receiver performs any WEBRC actions based upon its receipt. Unfortunately, there are currently no practical multicast packet authentication schemes that offer instant packet authentication upon receipt. However, TESLA [17] can be used to authenticate each packet a few seconds after receipt. Thus, TESLA could be used in conjunction with WEBRC to authenticate packets and for example terminate the session upon detection of a forged packet. However, it is RECOMMENDED that the normal WEBRC receiver responses to received packets are allowed to occur immediately and are not delayed by the TESLA authentication process. This is because the overall WEBRC performance would be greatly degraded if the receiver delayed its WEBRC response to packet receipt for several seconds.

A receiver with an incorrect or corrupted implementation of WEBRC may affect health of the network in the path between the sender and the receiver, and may also affect the reception rates of other receivers joined to the session. It is therefore RECOMMENDED that receivers be required to identify themselves as legitimate before they receive the session description needed to join the session.

Another vulnerability of WEBRC is the potential of receivers obtaining an incorrect session description for the session. The consequences of this could be that legitimate receivers with the wrong session description are unable to correctly receive the session content, or that receivers inadvertently try to receive at a much higher rate than they are capable of, thereby disrupting traffic in portions of the network. To avoid these problems, it is RECOMMENDED that measures be taken to prevent receivers from accepting incorrect session descriptions, e.g., by using source authentication to ensure that receivers only accept legitimate session descriptions from authorized senders.

8. IANA Considerations

No information in this specification is subject to IANA registration.

9. Intellectual Property Issues

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

10. References

[1] S. Bradner, ‘‘The Internet Standards Process -- Revision 3,’’ RFC2026, October 1996.

[2] S. Bradner, ‘‘Key words for use in RFCs to Indicate Requirement Levels,’’ RFC2119, March 1997.

[3] J.W. Byers, G. Horn, M. Luby, M. Mitzenmacher, W. Shaver. ‘‘FLID-DL: Congestion control for layered multicast,’’ IEEE J. on Selected Areas in Communications, Special Issue on Network Support for Multicast Communication, Vol. 20, No. 8, October 2002, pp. 1558-1570.

[4] 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.

[5] S. Deering, ‘‘Host Extensions for IP Multicasting,’’ RFC1112, August 1989.

[6] V. K Goyal, ‘‘On WEBRC Wave Design and Server Implementation,’’ Digital Fountain Technical Report no. DF2002-09-001, September 2002, available at http://www.digitalfountain.com/technology/.

[7] M. Handley, J. Padhye, S. Floyd, and J. Widmer, ‘‘TCP Friendly Rate Control (TFRC): Protocol Specification,’’ Internet Draft draft-ietf-tsvwg-tfrc-05, October 2002, a work in progress.

[8] H. W. Holbrook, ‘‘A Channel Model for Multicast,’’ Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, California, August 2001.

[9] R. Kermode and L. Vicisano, ‘‘Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents’’, RFC3269, April 2002.

[10] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft, ‘‘Asynchronous Layered Coding protocol instantiation,’’ Internet Draft draft-ietf-rmt-pi-alc-08, April 2002, a work in progress.

[11] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, ‘‘Layered Coding Transport building block,’’ Internet Draft draft-ietf-rmt-bb-lct-04.txt, February 2002, a work in progress.

[12] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, ‘‘The Use of

Forward Error Correction in Reliable Multicast,'' Internet Draft draft-ietf-rmt-info-fec-03.txt, September 2002, a work in progress.

[13] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, ''Forward Error Correction building block,'' Internet Draft draft-ietf-rmt-bb-fec-07.txt, September 2002, a work in progress.

[14] M. Luby and V. K Goyal, ''Wave and Equation Based Rate Control Using Multicast Round Trip Time: Extended Report,'' Digital Fountain Technical Report no. DF2002-07-001, September 2002, available at http://www.digitalfountain.com/technology/.

[15] M. Luby, V. K Goyal, S. Skaria, and G. B. Horn, ''Wave and Equation Based Rate Control Using Multicast Round Trip Time,'' Proc. ACM SIGCOMM 2002, Pittsburgh, PA, August 2002, pp. 191-214.

[16] A. Mankin, A. Romanow, S. Bradner, and V. Paxson, ''IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols,'' RFC2357, June 1998.

[17] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, ''Efficient and Secure Source Authentication for Multicast,'' Network and Distributed System Security Symposium, NDSS 2001, pp. 35-46, February 2001.

[18] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", Proc. IEEE Infocom '98, San Francisco, CA, March-April 1998, pp. 996-1003.

[19] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, and M. Luby, ''Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer,'' RFC3048, January 2001.

11. Authors' Addresses

Michael Luby
luby@digitalfountain.com
Digital Fountain
39141 Civic Center Drive, Suite 300
Fremont, CA, USA, 94538

Vivek K Goyal
vivek@digitalfountain.com
Digital Fountain
39141 Civic Center Drive, Suite 300
Fremont, CA, USA, 94538

12. Full Copyright Statement