End-to-end Congestion Control for Flows with Variable Packet Size

Jörg Widmer[1] Catherine Boutremans[2] Jean-Yves Le Boudec[2]

[1]Praktische Informatik IV
University of Mannheim
Mannheim, Germany

[2]Institute for Computer Communications and Applications
Swiss Federal Institute of Technology (EPFL)
Lausanne, Switzerland

Technical Report, ID: IC/2002/82

Abstract

Current TCP-friendly congestion control mechanisms such as those used in TFRC adjust the packet rate in order to adapt to network conditions and obtain a throughput not exceeding that of a TCP connection operating under the same conditions. In an environment where the bottleneck resource is packet processing, this is the correct behavior. However, if the bottleneck resource is bandwidth, and flows may use packets of different sizes, resource sharing then depends on packet size and is no longer fair. Now for some applications, such as Internet telephony, it is more natural to adjust the packet size, while keeping the packet rate as constant as possible. In this paper we study the impact of variations in packet size on equation-based congestion control and propose methods to remove the throughput bias resulting from the use of different packet sizes. We investigate in detail the design space of the approaches by means of mathematical analysis and propose a number of possible designs. We evaluate these designs through simulation and conclude with some concrete proposals. Our findings can be used to design a TCP-friendly congestion control mechanism for applications that adjust packet size rather than rate, or that are forced to use a small packet size. We base our analysis on the TFRC protocol, but similar considerations also hold for other congestion control mechanisms.

1 Introduction

The goal of end-to-end congestion control is to adapt the resource usage of a flow to the network's available resources, and to do so such that these are shared fairly between competing flows. Usually, the limited resource is either the packet rate or link capacity ("bandwidth"). In the former case, it is necessary to control the rate at which packets are sent, while in the latter either the packet rate or the packet size can be adjusted. A bandwidth-limited environment therefore allows to trade off packet size for packet rate (e.g., instead of reducing the packet rate to react to congestion, a flow may choose to maintain a high rate at the expense of a reduced packet size).

Current congestion control mechanisms such as the one used by TCP [1] or equation-based congestion control [13] aim to either reduce or increase the number of packets sent per time interval so as to adjust to the current level of congestion in the network. Reducing the packet rate in times of congestion is the correct behavior in a packet rate-limited environment. In an environment where the bottleneck is bandwidth limited and flows may use packets of different sizes, however, resources are not shared fairly among flows but governed instead by the packet size. A "TCP-friendly" congestion control mechanism using a smaller packet size than TCP would only achieve a fraction of the throughput justifiable according to its resource usage, if no corrective measure were taken. This fraction is, in first approximation, of the order of the ratio of the small packet size to the large packet size. One might argue that it is sufficient to account for the difference in packet size by some form of re-scaling, such as doubling the sending rate when packets are twice as small. As we explain below, such a simple scheme still has a large bias for high loss rates. Another scheme is to make active queue management such as RED [14] drop packets in direct proportion to their size. We show that this does not remove all bias either.

We expect the core network of the Internet to be mostly packet rate limited, as the packet processing rate of the routers is a greater bottleneck than the link capacity [22, 16]. However, there is little queuing in backbone networks, let alone congestion-related packet drops [20]. In contrast, access networks are often bandwidth limited, and this has been exacerbated by wireless access technologies. It is thus likely that many network paths are bandwidth limited.

Now there are applications that naturally prefer to trade off packet size for packet rate. This is true for "Voice over IP" (VoIP), where a high packet rate needs to be maintained in order to provide audio transmission with a sufficiently low delay [15]. Low-bit-rate codecs generate audio frames every 10, 20 or 30 ms, depending on the coding algorithm. There is a clear tradeoff between payload efficiency (payload/total packet size) and packetization delay (time to fill a packet), and one way to increase bandwidth efficiency would be to accumulate many audio frames within the same packet. However, many VoIP systems use 20 or 30-ms packets despite the low payload efficiency, in order to keep the end-to-end delay lower than 150 ms. Thus, audio sources typically generate packets at a constant rate and perform congestion control by switching codecs, which has the effect of varying their packet size [3, 2]. Other applications may be driven to adjust the packet size independently of congestion control (for example: a high bit error rate in a wireless environment induces a small packet size). Such applications have a variable packet size and simply adjusting their packet rate as though packets were path-MTU-sized is clearly not fair. While it can be argued that the throughput of such flows is comparatively low, not using any congestion control at all in a potentially congested environment is not acceptable [11].

In this paper we address the problem of how to make equation-based congestion control for an application that has a variable packet size compete fairly with TCP connections using path-MTU sized packets in a bandwidth-limited environment. In particular, these considerations apply to an application that modulates its rate by varying its packet size. Throughout the paper we will assume the bottleneck to be bandwidth limited. We thus have to consider modifications to congestion control mechanisms such that a flow sending small packets will not achieve a higher throughput than a flow sending larger packets. At the same time, a flow sending small packets should not achieve a much smaller throughput than what is justifiable given the actual resource usage of the flow.

Equation-based congestion control explicitly sets the sending rate using an equation that gives a TCP-friendly rate based on the current round-trip time, the loss event rate, and the packet size [17]. Therefore, using the packet size of a "reference" TCP flow (e.g., the MTU) in the equation rather than the flow's actual packet size seems to be a very simple way to achieve the same throughput as the reference TCP flow and therefore the same utilization of resources in a bandwidth-limited environment. This would allow a flow using equation-based congestion control to send smaller packets at a rate higher than TCP's. Unfortunately, such a simple approach does not work for the reasons we discuss now.

Similar to TCP, where commonly only one window reduction per congestion window is possible, a loss event is defined as one or more packets lost during one RTT (i.e., packet loss during an RTT is aggregated to a single loss event). Using the reference packet size in the equation results in a higher packet rate. The higher the number of packets per RTT, the more likely it is that multiple lost packets will be aggregated to a single loss event and the average number of loss events per packet will decrease, resulting in a strong bias *in favor* of sending small packets at a high rate. In Section 2.3, we give a quantitative analysis of the bias introduced into the loss measurement process when sampling the bottleneck at different rates. Our analysis is based on the unicast TFRC protocol [13] and also applies to its multicast counterpart TFMCC [24] since each calculates a fair sending rate in a similar manner.

To remove the bias, it is either possible to postprocess the measured loss event rate or to modify the loss measurement process itself. We present different algorithms for these purposes and discuss their characteristics in detail. The mechanisms are further evaluated through mathematical analysis and network simulation. As we will see later in the paper, a modified loss measurement is more robust under various network conditions and better able to produce a TCP-friendly rate. Only under very stable network conditions does postprocessing produce similar results.

Depending on the queuing scheme, the packet drop probability in the network is either related to the packet size (with drop-tail queues measured in bytes and RED in byte mode) or it is not (with drop-tail queues measured in packets and RED in packet mode).[1] Consequently, since the response from the network may differ significantly it is necessary to design both a *byte mode* and a *packet mode* version of the algorithms.

The results obtained above can also be applied when estimating a TCP-friendly rate without a rate control loop, as in equation-based congestion control. The TCP model used for equation-based congestion control relies on the packet rate being close to that of TCP so that the flow experiences congestion events similar to those a TCP flow would. For example, when using a constant bitrate flow to measure the network conditions, the modified loss measurement mechanisms can be used to calculate a valid TCP-friendly bitrate by removing the bias introduced by sampling the bottleneck at a different packet rate.

When the bottleneck is bandwidth limited, the presented mechanisms aim to achieve a sending rate comparable to that of a TCP flow with path-MTU discovery [18] under similar network conditions. Hence, for a fair sharing of resources, it is no longer necessary that competing flows have the same packet size. The modifications proposed in this paper are intended for applications that are forced to use a small packet size and would otherwise be tempted to use no congestion control at all. Their purpose is not to remove all incentives to use large packets instead of small ones. In addition to the less favorable ratio of the payload to the packet header for smaller packets (which is an incentive to send packets considerably larger than the header), it is possible to limit flows sending small packets to a throughput below the TCP-friendly rate (e.g., by mandating that the packet rate of equation-based congestion control must not exceed twice that of TCP). However, in this paper we will focus on how to determine this TCP-friendly rate. To the best of our knowledge, no prior work on the behavior of equation-based congestion control with variable packet sizes has been published.

The remainder of the paper is organized as follows. In Section 2 we show that the problem of fairly sharing resources with variable packet size cannot be solved using existing, or simple modifications to existing, end-to-end and queuing mechanisms. We then identify what specific problem needs to be addressed. In Section 3, we propose modifications to end-to-end congestion control. These solutions depend on whether the network path can be assumed to drop packets independent of their size or in proportion to it. We present several alternative solutions and tune their parameters by modeling their behavior over an erasure channel. In Section 4, we evaluate the different designs through extensive simulations under various network conditions. We find that only the solutions "virtual packets" and "random sampling" are robust enough if the network drops packets independent of packet size, while if the network implements the existing proposal for RED in byte mode, only "virtual packets" works. In Section 5, we summarize the findings and list some open issues.

2 Problem Definition

Before discussing in detail why simply using a "reference" packet size in the TCP model for equation-based congestion control as well as RED in byte mode are both insufficient for a fair sharing of resources, we will briefly introduce some fundamentals of equation-based congestion control that help in understanding the following sections.

[1]While one would expect the response to depend on the limited resource, this need not be the case in the Internet (e.g., with a drop-tail queue measured in packets in front of a bandwidth-limited bottleneck).

2.1 Foundations of Equation-Based Congestion Control

For TCP-friendly equation-based congestion control, the sending rate is commonly determined using a model for long-term TCP throughput such as the one described in [19].

$$\begin{aligned} R_{TCP} &= \frac{s}{RTT\,\left(\sqrt{\frac{2l}{3}} + \left(12\sqrt{\frac{3l}{8}}\right) l\,(1+32l^2)\right)} && (1) \\ &= f_{PFTK}(l) && (2) \end{aligned}$$

The expected throughput R_{TCP} of a TCP flow is calculated as a function of the loss event rate l, the round-trip time RTT, and the packet size s. A receiver measures its loss event rate and its RTT to the sender. It then uses the above equation to calculate R_{TCP} and feeds this rate back to the sender, which adjusts its sending rate accordingly.

The loss event rate l measures the frequency of *loss events*. A loss event consists of one or more packets lost within the same round-trip time. Since TCP should ideally halve the congestion window only once in response to several losses per round-trip time[2], the loss measurement process explicitly ignores losses within the same RTT and aggregates them to a single loss event. Consequently, a loss event represents the point in time where the TCP congestion window would be reduced in response to congestion. The initial packet loss that results in a loss event is followed by a period of time where all packet losses will be ignored by the loss measurement process. In this paper, we call this period the *Loss Insensitive Period* (LIP). On average, if a rate-controlled flow sends N packets per round-trip time, the LIP period consists of $N-1$ packets; since the initial packet loss is taken into account, it is not part of the LIP. A *loss interval* is defined as the number of packets between loss events. We denote the nth loss interval by θ_n. The loss event rate is then computed as the inverse of the weighted average loss interval calculated over a specific loss history.

$$l = \frac{1}{\sum_i w_i \theta_{n-i}}$$

Generally, older loss intervals are assigned a smaller weight than new loss intervals.

2.2 Network-Based Approach

One way to reduce the discrimination against flows sending small packets is to use RED gateways in byte mode [14]. With these gateways the fraction of packet drops for each connection sharing the bottleneck is roughly proportional to that connection's share of the bandwidth. Figure 1 shows the normalized throughput achieved by TCP and TFRC flows with large packets of 1000 bytes against the throughput of a TFRC flow sending packets of 100 bytes.[3] Instead of a factor of 10 (as with a per packet drop probability), the TFRC flow with small packets achieves a throughput only a factor of $\sqrt{10}$ worse than the throughput of the large flows. For high packet drop rates above 10%, the throughput of the flow with small packets even exceeds the throughput of "normal" TFRC and TCP because of the overproportional reduction of throughput of the TCP model given by Equation (1) in the high loss rate regime. Consequently, RED in byte mode does not suffice to ensure fairness between flows using different packet sizes.

2.3 End-to-End Approach

Instead of relying on the network to provide the appropriate packet drop rates, it is possible to modify the congestion control mechanism in the end systems.

[2]TCP Tahoe, NewReno and Sack generally halve the congestion window once in response to several losses per window while Reno reduces the congestion window twice in response to multiple losses in a window of data.

[3]The results were obtained by using a random dropper with drop probabilities proportional to the packet size in the *ns*-2 simulation environment.

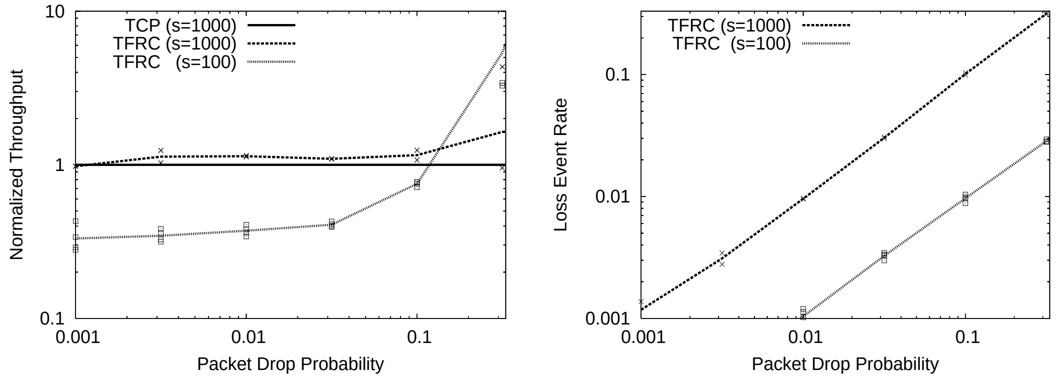


Figure 1: TFRC throughput and loss event rate with small packets and per-byte drops

Let S be the packet size for bulk data transfer, which is the MTU if the source uses path MTU discovery or 576 bytes, the minimum MTU that has to be supported by an Internet router. Given a bandwidth-limited bottleneck, a first step towards fairness is to use S instead of the actual packet size s of the flow in the loss-throughput formula. If the other parameters of the formula (i.e., loss event rate and RTT) remain unchanged, flows will achieve the same throughput and therefore use the same amount of resources at the bottleneck no matter what their packet size is.

Unfortunately, sending packets at a higher rate introduces a bias in favor of flows with small packets in the loss measurement process. The smaller the packet size (i.e., the higher the packet rate) and the higher the packet drop probability, the higher the probability to aggregate several packet drops within the same RTT to a single loss event. In this operating regime, the increase in the number of loss events is no longer proportional to the increase in the number of packets the loss events are sampled over and the measured loss event rate will decrease. A TFRC flow sending small packets at a high rate will therefore achieve a higher throughput than a TFRC flow with large packets or a TCP flow, since the size of the loss intervals is measured in terms of packets. This effect was also reported in [21].

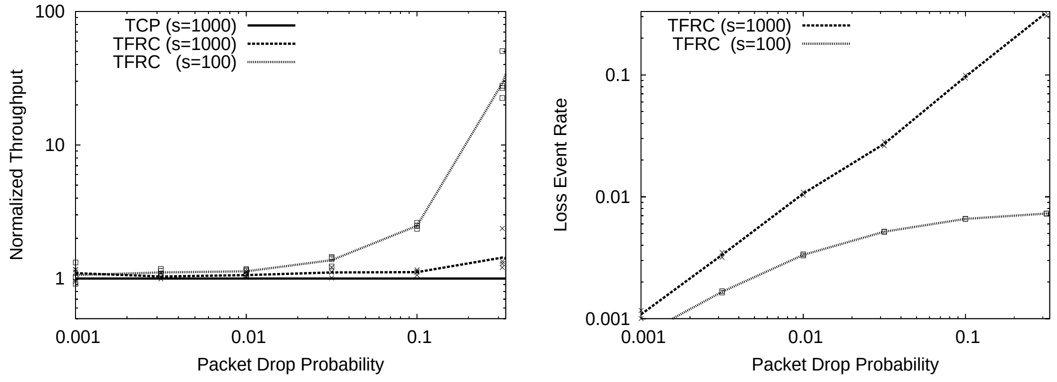


Figure 2: TFRC throughput and loss event rate with large packet size in the formula and per-packet drops

Figure 2 shows how the throughput for these types of flows varies for different packet drop rates. Here, a packet size of 1000 bytes for the large TFRC and the TCP flow and a packet size of 100 bytes for small TFRC flows were used. The TCP and the large TFRC flow achieve approximately the same rate under similar network conditions, with TFRC being slightly too aggressive in the regime of very high packet drop rates. In contrast, the TFRC flow with small packets is much more aggressive even in the regime of moderate packet drop rates between 1% and 10%, and for extremely high drop rates above 50% achieves more than 100 times the TCP throughput. If these flows were to compete against each other at the same bottleneck, the TFRC flow sending small packets would lock out the other flows. The higher the packet drop rate, the more pronounced is the bias in favor of small packet sizes.

Consider a rate-controlled flow sending N packets of size S per round-trip time. Assume that packets are dropped according to a Bernoulli packet loss process and let p be the packet drop probability. The nth loss interval θ_n is composed of the sum of (1) the $N-1$ packets within the LIP and (2) the number of packets between the end of the LIP and the next packet loss (including the lost packet), which we denote as X. Under the Bernoulli loss assumption, the random variable X follows a Geometric distribution $G(p)$ whose probability mass function is given as:

$$P(X=m) = (1-p)^{m-1}p\ ,\ m \in \mathbb{N}, m \geq 1$$

θ_n is thus defined as follows:

$$\theta_n = N - 1 + X \quad \text{where} \quad X \sim G(p)$$

The expected value of θ_n is given as:

$$E(\theta_n) = N - 1 + \frac{1}{p} \tag{3}$$

and its variance, $Var(\theta_n)$, is

$$Var(\theta_n) = \frac{1-p}{p^2} \tag{4}$$

Equation (3) shows that the higher the number of packets per round-trip time, the larger the expected loss interval and hence, the smaller the loss event rate. Consider now a flow sending $N\eta$ ($\eta \geq 1$) smaller packets of size $s = S/\eta$ per round-trip time. If the loss measurement process remains unchanged, a flow sending many more smaller packets will overestimate the average loss interval (i.e., $E(\theta_n) = N\eta - 1 + \frac{1}{p}$) compared to a flow sending large packets, leading to an unfair distribution of bandwidth.

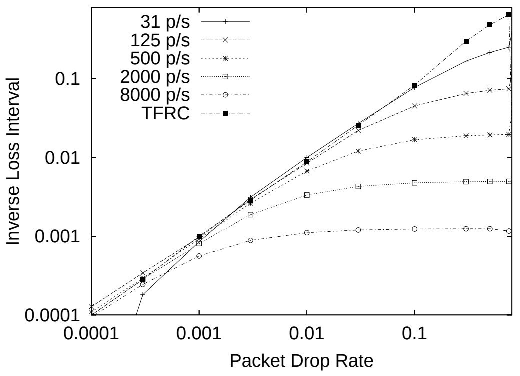


Figure 3: Loss Event Rate measured by different CBR flows

The bias introduced in the estimate of the loss event rate is illustrated in Figure 3. The figure shows the loss event rate measured by different constant bit rate (CBR) flows and by a TFRC flow for different packet drop rates. The TFRC flow constantly adjusts its sending rate to the measured loss event rate, whereas the CBR flows maintain a constant packet rate. Consequently, when packet drop rates are high, the loss event rate measured by the CBR flows depends almost exclusively on the (fixed) number of packets per RTT, since the flows experience a loss event very soon after the LIP. The relationship of packet loss rate and loss event rate shown in Figure 3 coincides perfectly with the results given as Equation (3).

3 Modifications to Equation-Based Congestion Control

In this section we propose methods of removing the bias introduced in the loss measurement process when using equation-based congestion control at a packet rate different from that of a TCP connection, as discussed in Section 2.3. For all of the methods, it is necessary to use the reference packet size S instead of the real packet size of the flow in the equation that determines throughput (Equation (1)). The solutions depend on whether the network drops packets irrespective of size ("Packet Mode"), or whether a scheme such as the existing proposal for RED in byte mode is already deployed ("Byte Mode").

The solutions are designed as follows. We tune their parameters such that the bias due to variable packet size is removed exactly when the end-to-end path drops packets as an erasure channel with independent drop events ("Bernoulli" loss model). Then, in Section 4, we evaluate by means of simulation the robustness to other channel conditions.

A more detailed analysis of the algorithms and the respective expected values of the resulting loss interval estimates can be found in Appendix B. There, we also discuss a different version of byte mode where the packet drop probability is derived from a byte loss probability (Appendix B.3).

3.1 Unbiasing (Packet Mode)

A first naive method to remove this bias in favor of small flows is to measure the loss interval as before, compute the bias, and then remove this bias from the measured loss interval. A connection sending $N\eta$ packets of size $s = S/\eta$ per round-trip time measures an average loss interval (in packets) of

$$\begin{aligned} E(\theta_n) &= N\eta - 1 + \frac{1}{p} \\ &= [N - 1 + \frac{1}{p}] + (\eta - 1)N \end{aligned}$$

instead of $E(\theta_n) = N - 1 + \frac{1}{p}$. The simplest way to obtain a correct measure of the loss interval is to subtract $(\eta - 1)N$ from the measured interval. If the receiver reports the measured loss event rate to the sender, the sender can adjust the measurement accordingly before calculating a TCP-friendly sending rate. Since both the sender and the receiver know η and N, the correction can be carried out by either of them but as we will see later, other modifications to the receiver lead to much better results. Therefore, removing the bias only makes sense at the sender if the receiver cannot be modified.

The correction depends critically on the use of correct values for η and N, which is very difficult since N and possibly also η vary over time. Directly using the current values can lead to very large variations in the corrected loss interval size. However, it is equally difficult to properly smooth the values of N and η used for the correction without introducing additional artifacts.

3.2 Unbiasing (Byte Mode)

In the event that the packet drop probability is proportional to the packet size, it is possible to derive a method similar to the one discussed in Section 3.1. Consider a packet loss process where the packet drop probability depends on the packet size as follows:

$$p_S = \eta\, p_s \tag{5}$$

where p_S and p_s respectively denote the probabilities of a packet of size S and s $(= S/\eta)$ to be dropped.

In that case, a flow sending $N\eta$ packets of size s per round-trip time measures an average loss interval of:

$$\begin{aligned} E(\theta_n) &= N\eta - 1 + \frac{1}{p_s} = N\eta - 1 + \frac{\eta}{p_S} && (6) \\ &= [N - 1 + \frac{1}{p_S}] + (\eta - 1)N + \frac{\eta - 1}{p_S} && (7) \end{aligned}$$

while a flow sending N packets of size S per round-trip time sees an average loss interval $E(\theta_n) = N - 1 + \frac{1}{p_S}$. In this case, the way to obtain a correct estimate of the loss interval is to subtract $(\eta - 1)N + \frac{\eta-1}{p_S}$ from the measured loss interval. The main difficulty at this point is that the end system does not know p_S and thus has to estimate it from the measured loss interval. Since the end system knows η and N, it can easily deduce the value of p_S from the measured interval using Equation (6) and then compute the bias as

$$\frac{(\eta - 1)(\theta_n + 1)}{\eta}$$

Like its packet-mode counterpart, this method depends on the use of a correct value for η. Instead of N, the bias now depends on θ_n and again it is necessary to properly smooth the value to avoid large variations in the bias.

Discussion. Modifying the measured loss event rate is very challenging when network loss conditions are not perfectly stable. In order to more reliably remove the bias introduced in the measure of the loss event rate over a certain time span, it is necessary to exactly follow the dynamics of the loss process during this time interval. Since the sender does not have access to this information, it will not be able to accurately estimate the bias under dynamic loss conditions. It is therefore preferable to modify the loss measurement process itself, instead of trying to modify its outcome. This is the essence of the solutions proposed in the rest of this section and we show in Section 4 that they are indeed more robust.

3.3 Modifi cations to the Loss Measurement Mechanism (Packet Mode)

In the following, we will present three alternative modifications to the TFRC's loss measurement mechanism. The aim of these modifications is to estimate the process θ_n that would be measured by a flow using path MTU discovery (i.e., sending packets of size S). For gateways in packet mode, the packet drop probability p for flows sending large packets and for flows sending small packets is the same.

Under the assumption of Bernoulli loss, we show analytically that these modified loss measurement algorithms are unbiased estimators of θ_n (i.e., a flow sending $N\,\eta$ ($\eta \geq 1$) smaller packets of size $s = S/\eta$ per round-trip time will estimate the same average loss interval as a flow with packets of size S, regardless of the value of η). In addition, we evaluate the variance of the different estimators. This is useful for comparing the various solutions and understanding their conservativeness.

3.3.1 Virtual Packets

The main idea of a loss measurement mechanism based on virtual packets is to combine small packets of size s into packets of size S. Whenever a receiver receives S or more bytes (in packets of size s), it records the arrival of a virtual packet. Similarly, a virtual packet is marked as lost when the amount of bytes lost exceeds S. Figure 4(b) gives an overview of how virtual packets are formed.

To apply this method, it is necessary to modify the definition of loss event and loss interval. The duration of the LIP remains unchanged (i.e., on average it comprises $N\eta - \eta$ packets of size s).

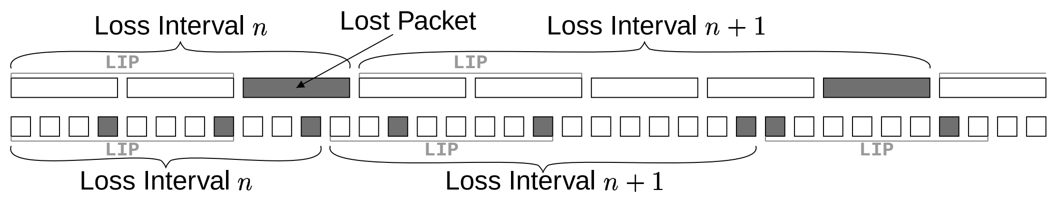
Defi nition 1 *A packet loss constitutes a* loss event *if (a) the LIP following the last loss event ended and (b) at least S bytes were lost since the end of the LIP.*

Defi nition 2 *A* loss interval *is measured as the number of virtual packets received between two successive loss events, including the lost packet that ends the loss interval.*

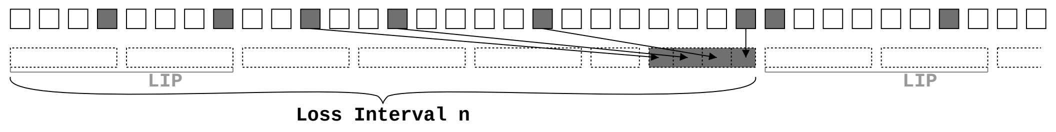
The mechanism aims at reducing the number of packets that form a loss interval by "normalizing" the number of packets by means of the concept of virtual packets. Note that the size of a loss interval need not be whole-numbered.

From Definition 1, a flow sending packets of size s experiences a loss event as soon as η packets have been lost since the end of the last LIP. Let θ_n^{VP} be the nth loss interval measured by a flow using the virtual-packets method. From Definition 2, under the Bernoulli loss assumption θ_n^{VP} is defined over the set $\{N + \frac{i}{\eta}, i \in I\!N\}$ as follows:

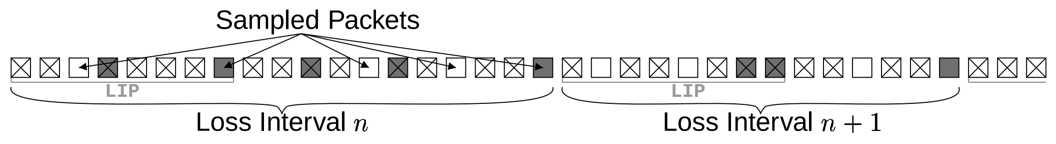
$$\theta_n^{VP} = N - 1 + \frac{1}{\eta}\sum_{i=1}^{\eta} X_i \quad \text{with} \quad X_i \sim G(p)\ ,\ i = 1, \ldots, n \tag{8}$$



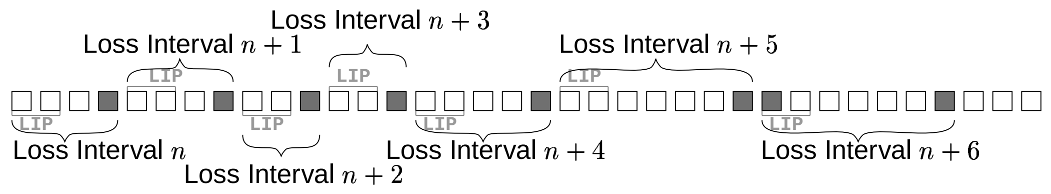
(a) Unmodified Measurement Mechanism



(b) Virtual Packets



(c) Random Sampling



(d) LIP Scaling

Figure 4: Schematic illustration of the three solutions presented in Section 3.3.

where the random variables X_i , $i = 1, \ldots, n$ are independent and identically distributed (they follow a Geometric distribution of parameter p). The expected loss interval resulting from Equation (8) is the same as the one defined in Equation (3).

The variance of the loss intervals measured using the virtual-packets method is given as follows:

$$Var(\theta_n^{VP}) = \frac{1}{\eta^2} \sum_{i=1}^{\eta} Var(X_i) = \frac{1}{\eta} \frac{1-p}{p^2} \qquad (9)$$

Equation (9) shows that the virtual-packet algorithm smoothes out the measure of θ_n since the variance of θ_n^{VP} is η times smaller than the the variance of θ_n. As shown in [23], a high variability of the loss event estimator will result in a more conservative congestion control. The reduced variation of the measured loss interval therefore causes the virtual-packet method to achieve a slightly higher sending rate than that of the unmodified TFRC.

3.3.2 Random Sampling

The same normalization effect can be achieved in a different way. Instead of aggregating small packets into large ones, the loss interval can be normalized by excluding excess packets (and excess packet losses) from the loss measurement process. Consider a flow that sends packets of size s at the same bitrate and thus at a higher packet rate than that of a flow with packets of size S. Upon packet arrival (or the detection of a packet loss), the receiver performs a random experiment that succeeds with the probability $\frac{s}{S}$. Only when the random experiment succeeds is the packet arrival or packet loss taken into account in the loss measurement process. To apply this mechanism, the same notion of LIP, loss event and loss interval as the one for the original loss measurement process can be used. An example of random sampling is given in Figure 4(c).

Let θ_n^{RS} be the nth loss interval measured using the random sampling method. In practice, each packet of size $s = S/\eta$ has a probability $p_\eta = 1/\eta$ to be sampled by the loss measurement process. Thus, the loss interval θ_n^{RS} is the sum of:

- the number of packets sampled during the LIP, which we denote as Y and
- the number of correctly received sampled packets between the end of the LIP and the next sampled packet that is lost, which we denote as X.

Since packets are sampled independently with probability p_η, Y represents the number of successes in a sequence of $(N-1)\eta$ Bernoulli trials and therefore has a Binomial distribution $Bin((N-1)\eta, p_\eta)$, where p_η is the probability of success. Since packets are sampled independently with the same probability (and thus independently from the loss process), the number of correctly received sampled packets between the end of the LIP and the next sampled packet lost, X, is a Geometric random variable of parameter p, $G(p)$. Finally, as a result of the independence of the sampling and loss processes, the random variables X and Y are independent. θ_n^{RS} is thus defined as follows:

$$\theta_n^{RS} = Y + X \quad \text{where} \quad \begin{aligned} Y &\sim Bin((N-1)\eta, p_\eta) \\ X &\sim G(p) \end{aligned} \tag{10}$$

From Equation (10), the expected loss interval is given as:

$$E(\theta_n^{RS}) = (N-1)\eta\frac{1}{\eta} + \frac{1}{p} = N - 1 + \frac{1}{p} \tag{11}$$

Equation (11) shows that θ_n^{RS} is an unbiased estimator of θ_n. The variance of θ_n^{RS} is computed as follows:

$$\begin{aligned} Var(\theta_n^{RS}) &= Var(Y) + Var(X) \ \text{(X and Y are independent)} \\ &= (N-1)\eta\frac{1}{\eta}(1 - \frac{1}{\eta}) + \frac{1-p}{p^2} \\ &= (N-1)(1 - \frac{1}{\eta}) + \frac{1-p}{p^2} \end{aligned} \tag{12}$$

The first term of the variance as expressed in Equation (12) can be removed by taking $Y = N-1$. In this case, we would measure $N-1$ (by aggregating small packets during the LIP) instead of estimating it by random sampling packets during the LIP. This would amount to mixing the virtual-packet and the random-sampling methods and would result in the same variance for θ_n^{RS} as the one of θ_n.

3.3.3 LIP Scaling

A third method to reduce the number of packets contained in a loss interval is to reduce the duration of the LIP. When the loss insensitive period is scaled in proportion to the factor $\frac{s}{S}$, then the LIP of a flow sending small packets should on average contain the same number of packets as the LIP of a flow sending larger packets.

In particular, a flow sending N packets of size S per round-trip time and a flow sending $N\eta$ packets of size s per round-trip time both send $N-1$ packets per LIP if the LIP of the small flow is η times smaller than that of the large flow. As a consequence, both flows will calculate roughly the same loss event rate, given that they experience the same packet drop rate. The LIP scaling mechanism is illustrated in Figure 4(d).

A side effect of the LIP scaling method is that it changes the responsiveness of the loss measurement process. Reducing the LIP increases the responsiveness of the flow, which can give rise to undesirable oscillations in the measure of the loss interval. To reduce this effect and make sure that the network conditions are measured over the same timescale as a flow sending large packets, we increase the size of the loss history proportionally to the factor $\frac{S}{s}$. This way, the loss history should comprise roughly the same time interval as that of flows with large packets, while the loss event rate would be calculated over many more samples. The impact of these changes of timescale on the dynamics of the algorithm will be analyzed in more detail in the simulation section.

3.4 Modifications to the Loss Measurement Mechanism (Byte Mode)

With RED gateways operating in byte mode, the probability of dropping a packet of size S is η times greater than the probability of dropping a packet of size s (i.e., $p_S = \eta\, p_s$). As a consequence, the average interval (in terms of packets) between two packet losses is roughly η times smaller for the flow sending packets of size S than for the flow sending small packets of size s. The average number of bytes between two packet losses is roughly the same for all the flows, independent of the packet size.

Again, we present three different methods. However, in contrast to packet-mode methods, they are not equally well equipped to remove of the bias in the loss measurement process.

3.4.1 Virtual Packets

While for the virtual-packet algorithm operating in byte mode the packets arriving in between loss events are still aggregated to virtual packets of size S, a loss event is declared as soon as a packet of s bytes is lost after the LIP. Thus, we have to relax Definition 1 and introduce a new definition of a loss event:

Definition 3 *A packet loss constitutes a* loss event *if the LIP following the last loss event ended.*

The definition of a loss interval remains the same as the one given in Definition 2. Let θ_n^{VPb} be the nth loss interval measured by a flow using the virtual-packet method in byte mode. In Definitions 2 and 3, θ_n^{VPb} is defined as the sum of (1) the number of virtual packets within the LIP, $N-1$, and (2) the number of virtual packets between the end of the LIP and the next packet loss (including the lost packet), X, which is a Geometric random variable of parameter p_s. The random variable θ_n^{VPb} is thus defined as follows:

$$\theta_n^{VPb} = N - 1 + \frac{1}{\eta}X \quad \text{where} \quad X \sim G(p_s) \tag{13}$$

and the expected loss interval size is given as:

$$E(\theta_n^{VPb}) = N - 1 + \frac{1}{\eta p_s} = N - 1 + \frac{1}{p_S} \tag{14}$$

3.4.2 Random Sampling

It is possible to construct an algorithm for RED in byte mode that is based on random sampling. This amounts to a random sampling of arrived data packets, whereas lost data packet always need to be taken into account. While the virtual-packets and the random-sampling mechanisms for packet mode are each a valid mechanism on their own with slightly different properties, random sampling in byte mode merely ignores information that is available to the receiver. The number of packets between packet drops is estimated instead of being measured directly as in the virtual-packet approach. Generally, we expect random sampling in byte mode to be inferior to virtual packets in byte mode.

3.4.3 LIP Scaling

Adjusting LIP scaling to byte mode results in an even less favorable mechanism. Here, the expected number of lost packets within the LIP does not increase with a decrease in packet size; therefore, reducing the duration of the LIP would require introducing artificial packet losses into later intervals. We do not recommend the use of LIP scaling in combination with a bottleneck in byte mode.

4 Evaluation by Simulation

In order to evaluate our solutions, we implemented them in the *ns*-2 network simulator [5] and measured their performance under various conditions. This allows us to check the feasibility and investigate the behavior under more realistic settings than the ones used in the design phase in Section 3. In a first set of simulations, the channel is artificial; this allows us to isolate the effect of the channel properties. In a second set of simulations, we use realistic network settings with RED and drop-tail queues.

For the simulation topology, we use the well-known dumbbell topology, with senders and receivers on either side of a single bottleneck link, as depicted in Figure 5. The access links to the routers are provisioned with 100 MBit/s, while the bottleneck link between the routers either has a lower capacity, or a loss module is inserted at the bottleneck router so that the total bandwidth consumed by all flows is well below 100 MBit/s.

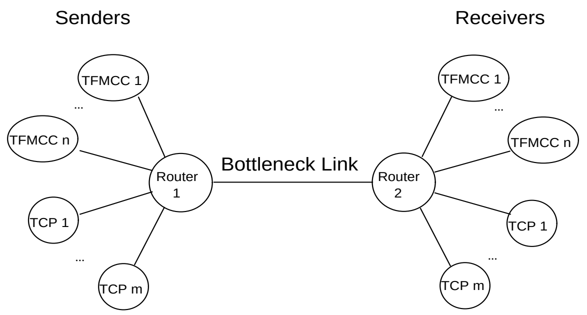


Figure 5: Dumbbell topology

When discussing the scenarios, we will denote TFRC flows which use the same packet size as TCP as “TFRC” and TFRC flows with a different packet size or with a fixed packet rate as “VP-TFRC”. The standard *ns*-2 implementation of TCP Sack is used.[4] For all simulations, the reference TCP packet size is 1000 bytes.

4.1 Artifi cial Channel

Before investigating more complex scenarios where flows interact and compete for resources at a common bottleneck, we will analyze whether VP-TFRC, TFRC, and TCP flows achieve a similar sending rate when the packet drop rate is independent of the sending rate of the flows. Since the model for TCP (Equation (1)) is based on this assumption, equation-based congestion control should work best in such an environment.

We use two different parameter settings for the VP-TFRC flows: one where the VP-TFRC packet size is fixed and the sending rate is modified by changing the packet rate, and another where the packet rate is a fixed number of packets per second while the packet size varies so as to achieve the desired sending rate. Furthermore, we use three different loss models for our analysis: Bernoulli loss, Bernoulli loss with a drop rate varying over time, and a Gilbert loss model. The Bernoulli dropper discards each incoming packet with the same probability. The second loss model provides more variable network conditions where the packet drop probability alternates between high and low. While the average drop probabilities are the same as those in the first loss model, the congestion control protocols frequently have to adjust their sending rate, putting more emphasis on the transient behavior of the protocols. In the third, the Gilbert loss model, packet losses are no longer independent but highly correlated. Our time-based model[5] alternates between the no-loss state “0” and the loss state “1” with a transition probability p to switch from the no-loss state to the loss state and a transition probability q to remain in the loss state (Figure 6). The model remains in the current state for a fixed amount of time, τ, after which a random experiment is performed to see whether a state change

[4]In particular, the parameters that determine TCP’s timeout behavior remain unchanged with `timestamps_=false`, `tcpTick_=0.01`, and `minrto_=1`

[5]A packet-based model would have the disadvantage, that the timescale over which loss bursts occur depends to a large degree on the packet rate. Furthermore, if fbws with different packet rates are run concurrently under such conditions, they are no longer independent from each other but would experience very different loss patterns.

should occur. For the Gilbert model, the average loss rate is $p/(p+q)$ and the average length of a loss burst is $1/q$ time units. To arrive at the same average packet drop probability $\bar{p}$ as in the previous two models, we set $p = a\bar{p}$ and $q = a(1-\bar{p})$. The higher the average packet drop probability, the higher the average burst length. The parameter a can be used to modify the burst length while keeping the same average packet drop probability. In the following simulations we use $a = 0.8$ and $\tau = 10$ ms.

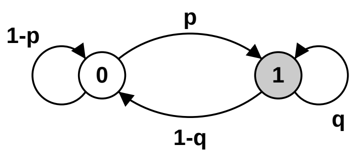


Figure 6: Gilbert loss model

With our mechanisms we aim to emulate a TFRC flow with large packets, so the best we can hope for is a throughput similar to the throughput of such a flow. To assess the proposed mechanisms, we therefore normalize the throughput graphs by dividing by the throughput of a TFRC flow with large packets. To allow a comparison with TCP, Figure 7 depicts the ratio of throughput of plain TFRC (with large packets) to TCP throughput. This allows us to separate the differences in throughput introduced by using equation-based congestion control in general from those stemming from the use of a different packet size.

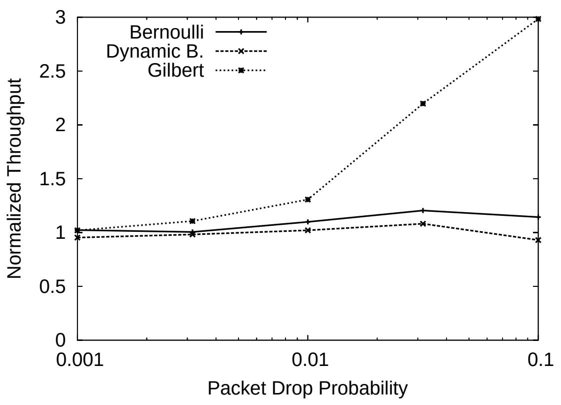


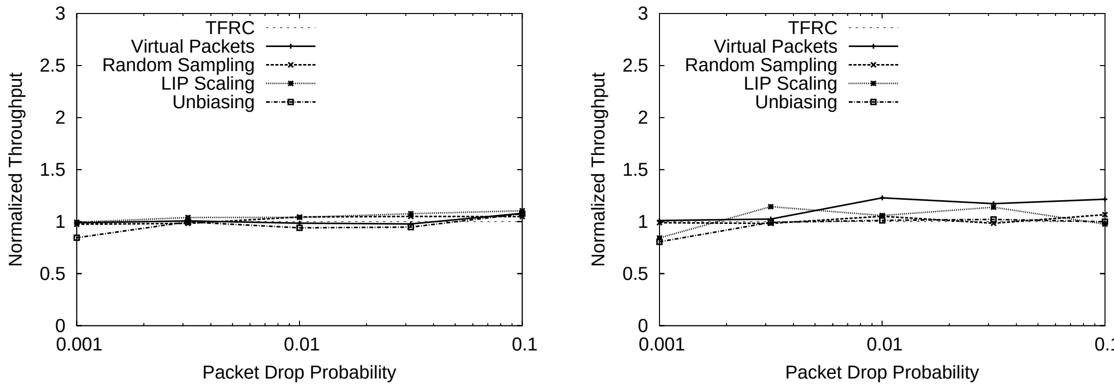
Figure 7: Fairness of plain TFRC with different loss models

For the Bernoulli loss model, TFRC throughput coincides well with TCP throughput and becomes slightly too aggressive in the regime of high loss rates. When the packet drop rate changes over time, TFRC is a bit more conservative, but in general, equation-based congestion control works very well for both loss models. The results are quite different for the third loss model, where TFRC is significantly more aggressive than TCP when drop rates are high.

The reason for this discrepancy in TFRC and TCP throughput results from the correlation of packet drops. Since TCP sends packets back-to-back, it is likely that a number of them arrive during the loss state of the Gilbert model. For higher loss rates, the congestion window comprises only a few packets and the number of consecutive packet losses is large, leaving too few packets in the pipe to allow fast recovery. Instead of a triple duplicate ACK for one or more segments lost in a congestion window, TCP frequently experiences a timeout, many more than are to be expected with the Bernoulli dropper given the same loss rate. Since the TCP model used for TFRC is based on different assumptions about the packet loss pattern, TFRC's sending rate will be too aggressive.

4.1.1 Bernoulli Loss Model

The most basic scenario to investigate is based on a loss module inserted at the bottleneck router, that drops incoming packets with a fixed probability. With a delay for the bottleneck link of 30 ms and 10 ms for the access links, the RTT for all flows is almost constant at 100 ms. Since the achieved sending rates are below the bottleneck bandwidth, no queuing occurs and the queuing strategy has no impact on the simulation. Furthermore, flows running concurrently will not influence each other.



(a) VP-TFRC packet size 100 bytes (b) VP-TFRC packet rate 160 packet/second

Figure 8: Bernoulli loss

All different mechanisms (i.e., the sender-based unbiasing, as well as the three receiver-based loss measurement modifications) achieve a throughput very close to the throughput of a TFRC flow with large packets and thus very close to TCP throughput. When the packet size of the VP-TFRC flow is set to 100 bytes and the packet rate varies, we obtain a normalized throughput of VP-TFRC as depicted in Figure 8(a). Here, the deviation in throughput is nearly always less than 10%. Direct unbiasing is slightly more conservative than TFRC and the other methods are slightly more aggressive, with virtual packets resulting in the throughput most closely resembling that of TFRC.

In contrast, when the packet rate is fixed at 160 packets/second and the packet size varies (see Figure 8(b)), the virtual-packets method has the highest deviation from TFRC throughput, while unbiasing as well as random sampling achieve almost exactly the TFRC rate. Nevertheless, at less than 20% the difference is relatively slight. As all of the mechanisms are based on the assumption of a Bernoulli loss process, these good results are to be expected.

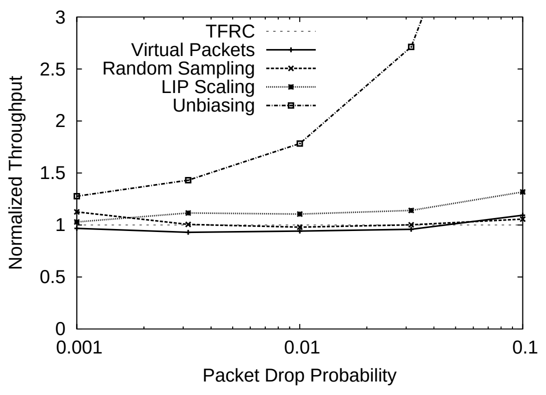
4.1.2 Dynamic Bernoulli Loss Model

To analyze the behavior under more dynamic network conditions, we use a Bernoulli packet dropper, where the drop rate alternates between 0.5 times and 1.5 times the average drop rate. The packet drop rate changes every 10 seconds (i.e, 24 times over the entire simulation of 250 seconds).

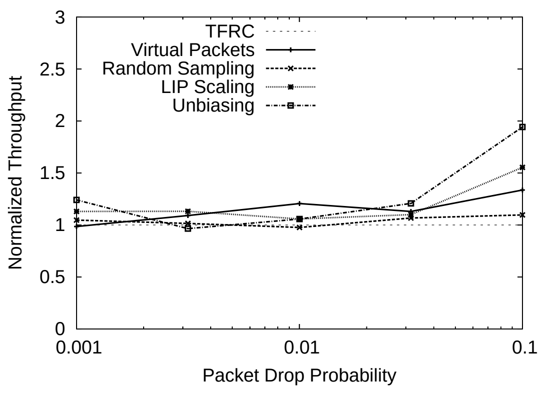
As soon as network conditions become more dynamic, the shortcomings of the simplest of the mechanisms, the unbiasing of the loss interval, become obvious. Unbiasing is much more aggressive than TFRC for both a fixed packet size and a fixed packet rate, but this effect is much more pronounced for a fixed packet size. For packet drop rates of more than a few percent, the throughput is a multiple of the rate achieved by either TCP or TFRC. The methods of virtual packets and random sampling behave quite well, while LIP scaling is somewhat too aggressive in the case of fixed packet size, as shown in Figure 9(a). All mechanisms tend to become more aggressive than TFRC when the packet rate is fixed and the drop probability is high. Under such conditions, random sampling performs best, with only a marginal increase in sending rate compared to plain TFRC (Figure 9(b)).

4.1.3 Gilbert Loss Model

The parameters of the time-based Gilbert model specified above ($\tau = 10$ ms and $a = 0.8$) are not intended to closely model network conditions we expect to find in the Internet but are merely used to analyze how the



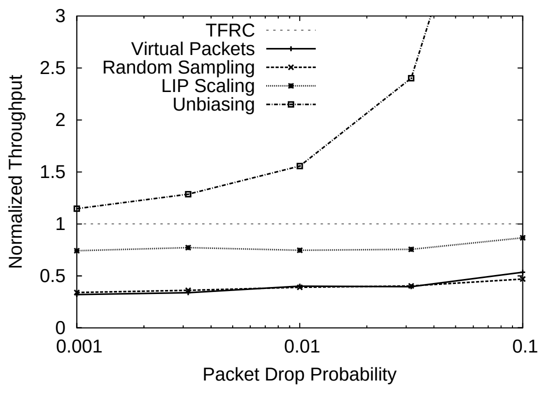
(a) VP-TFRC packet size 100 bytes



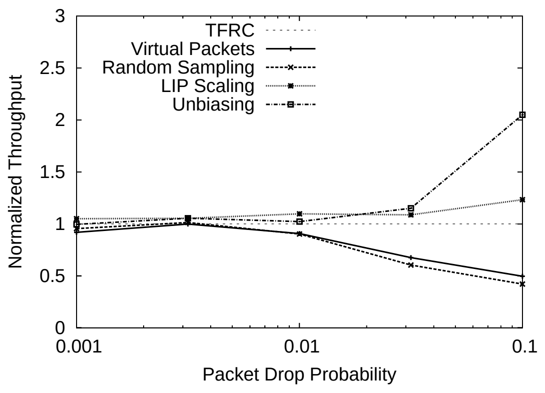
(b) VP-TFRC packet rate 160 packets/second

Figure 9: Dynamic Bernoulli loss

mechanisms perform when the assumption of packet loss independence is not met.



(a) VP-TFRC packet size 100 bytes



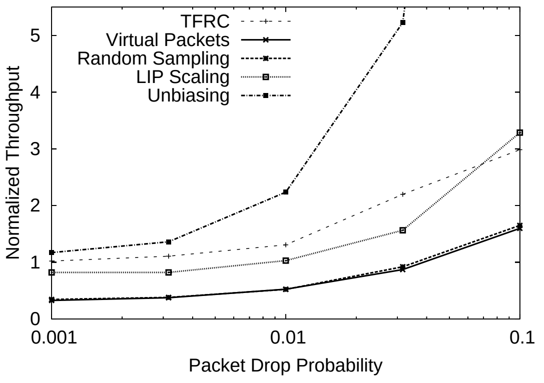
(b) VP-TFRC packet rate 160 packets/second

Figure 10: Gilbert loss model

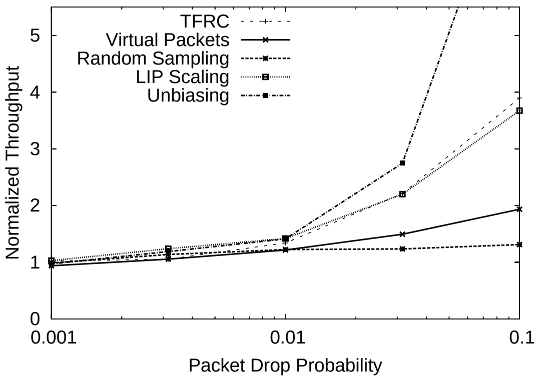
As far as the throughput of the different mechanisms is concerned, there is a striking difference from the Bernoulli based experiments. All of the methods are far from fair (for all packet drop rates in the case of a fixed packet size and only for higher packet drop rates in the case of a fixed packet rate). As in the dynamic Bernoulli experiments, unbiasing is by far the most aggressive scheme, making it unsuitable for all but very simple static network conditions. LIP scaling achieves a somewhat lower sending rate and becomes a bit more aggressive than TFRC for high loss rates in the fixed packet rate case. Given that TFRC itself is much more aggressive than TCP under such circumstances (see Figure 7), we do not recommend LIP scaling for such network environments.

Virtual packets and random sampling perform alike, with a throughput of less than 50% of that of TFRC when the interval between packets varies (Figure 10(a)), and going from fair to around 50% of TFRC throughput when the interval between packets is fixed (Figure 10(b)). Therefore, these mechanisms achieve a throughput much closer to TCP throughput than plain TFRC, as shown in Figure 11.

We note that this improvement in fairness is caused by two effects that counterbalance each other (equation-based congestion control in general being too aggressive under such network conditions and the modified loss measurement mechanisms resulting in an overestimation of the loss event rate), not because the mechanisms themselves better model TCP performance. Nevertheless, with these mechanisms we achieve roughly the same performance under normal circumstances and they are more conservative than TFRC under unfavorable



(a) VP-TFRC packet size 100 bytes



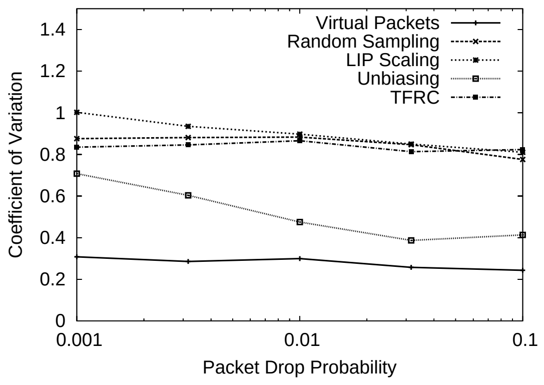
(b) VP-TFRC packet rate 160 packets/second

Figure 11: Gilbert loss model with throughput relative to TCP

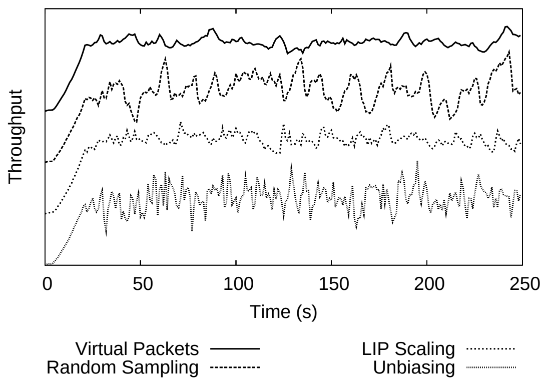
network conditions where TFRC is too aggressive. This is the behavior we would like to see in a modified TFRC protocol and we therefore recommend the methods of virtual packets and random sampling rather than unbiasing and LIP scaling (unless with known favorable network conditions and a high level of statistical multiplexing).

4.1.4 Throughput Stability

As discussed in Section 3.3, the proposed loss measurement mechanisms have a different variance. Figure 12(a) shows the coefficient of variation (CoV) of the measured loss interval size from the previous experiments based on the Bernoulli loss model. The variations of the loss interval size have an impact on the stability of the throughput, which is depicted in Figure 12(b).[6]



(a) Coeffi cient of variation of the loss interval size



(b) Throughput over time

Figure 12: Throughput variance

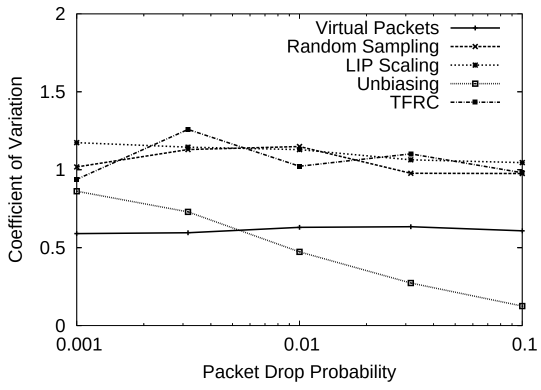
As expected, LIP scaling and random sampling have roughly the same CoV as an unmodified TFRC, while the CoV of virtual packets is significantly lower. Consequently, using virtual packets results in the most stable sending rate. A similarly stable throughput is achieved with LIP scaling. While with a Bernoulli loss model the loss intervals of LIP scaling have the same CoV as those of plain TFRC, the large loss history smoothes out throughput variations. Random sampling results in roughly the same smoothness of throughput

[6]All mechanisms have the same average throughput; the lines have been shifted for better readability. The plot shows the throughput of a simulation with a drop probability of 0.01.

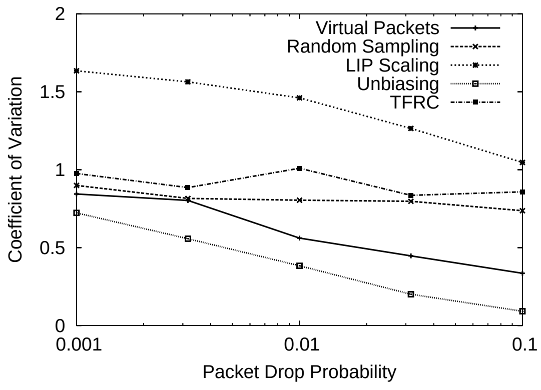
as an unmodified TFRC. Unbiasing has an intermediate CoV with variation which decreases at higher loss rates. Despite a loss interval CoV lower than those for LIP scaling and random sampling, unbiasing delivers the most bursty sending rate. The loss interval is unbiased after the measurement and this unbiasing process depends on the current packet sending rate as well as on the current packet size. The noise inherent in the measurement of these parameters causes the burstiness in the sending rate of the unbiasing method.

Hence, for applications that require a smooth sending rate, virtual packets is the method of choice, while random sampling can be used to achieve the same responsiveness as with an unmodified TFRC.

The CoV for the dynamic network scenarios is shown in Figures 13(a) and 13(b). Analyzing the dynamic Bernoulli loss model shows the CoV is generally higher, but the results themselves are comparable. With the Gilbert loss model however, the CoV of LIP scaling is significantly higher than that of the other methods. Since packet loss is correlated, patterns of packet loss are very different on the timescale of the LIP scaling method, resulting in a much higher variation of throughput.



(a) Coefficient of variation for dynamic Bernoulli loss



(b) Coefficient of variation for Gilbert loss

Figure 13: Throughput variance under dynamic network conditions

4.2 Bandwidth Limited Bottlenecks

After gaining a first insight into the performance of the proposed algorithms, in this section we will analyze their performance under more realistic network conditions where flows with small and large packets compete directly at a bandwidth-limited bottleneck.

4.2.1 Simulation Setup

In the simulations with bandwidth-limited bottlenecks, we use a propagation delay of 10 ms for the access links and 80 ms for the bottleneck link (in addition to the serialization delay and a possible queuing delay). The bottleneck capacity is scaled to the number of flows. Usually, only flows with a low bandwidth requirement are forced to using a packet size smaller than the MTU, therefore, we mainly investigate settings with a low per-flow bandwidth. For all the simulations, we investigate three different settings:

- The packet size is fixed at 100 bytes and the fair bandwidth is 96 KBit/s per flow.
- The packet rate is fixed at 160 packets/s, the maximum packet size is 100 bytes (resulting in a maximum sending rate of 128 KBit/s) and the fair bandwidth is 96 KBit/s per flow.
- The packet rate is fixed at 50 packets/s, the maximum packet size is 200 bytes (resulting in a maximum sending rate of 80 KBit/s) and the fair bandwidth is 64 KBit/s per flow.

Setting a reasonable queue size for the simulations is not an easy task. Generally, TCP performs better when there is a large amount of buffer space available (so that there is enough space to accommodate TCP's packet bursts), while TFRC is relatively insensitive to the queue size and therefore outperforms TCP when the queue size is small. Particularly if the queue size is measured in packets, with a potentially large number of small packets in the queue the queue size available to TCP may vary significantly. When the queue is measured in bytes, a large TCP packet occupies the space of many small VP-TFRC packets and when TCP sends a burst of packets, it may occupy a large fraction of the queue space. However, due to TFRC's insensitivity to the available queue size, this has only a relatively small impact.

We chose to set the queue parameters as follows:

- If the queue is measured in bytes, we set the queue size to twice the bandwidth-delay product, assuming a RTT (including buffer delay) of 500 ms.
- If the queue is measured in packets, we set the average packet size to $\frac{2S}{1.0+S/s}$ where S is the TCP packet size and s is the size of the VP-TFRC packets in the case of a fixed packet size or the size of a VP-TFRC packet if the flow were sending at exactly the fair rate in the case of a fixed packet rate. The queue size in packets is then set to twice the bandwidth-delay product divided by the average packet size.
- For RED queues, we further set the minimum threshold min_{th} to 5% of the queue size, the maximum threshold max_{th} to 50% of the queue size, and the maximum packet drop probability max_p to one drop per 22.5% of the queue size in packets (the average of min_{th} and max_{th}), which are the recommended parameter settings from [6]. The $gentle\_$ option of RED is enabled.

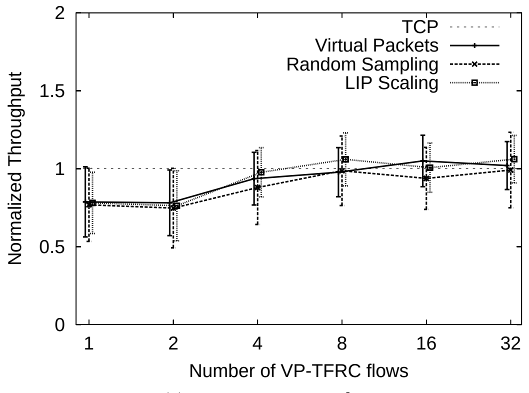
The same number of VP-TFRC flows and TCP flows was used for all the simulations (i.e., 1vs1, 2vs2, etc.). The simulation results were averaged over six runs for each parameter setting (together with slight variations in the available bandwidth and the starting times of the flows so as to provide some degree of randomness). All the experiments were also conducted with half the queue size (i.e., using the equivalent of one bandwidth-delay product) with the expected results of a decrease in the level of fairness in favor of TFRC and VP-TFRC. For reasons of brevity, we only present a small selection of the simulation results.

4.2.2 Packet Mode

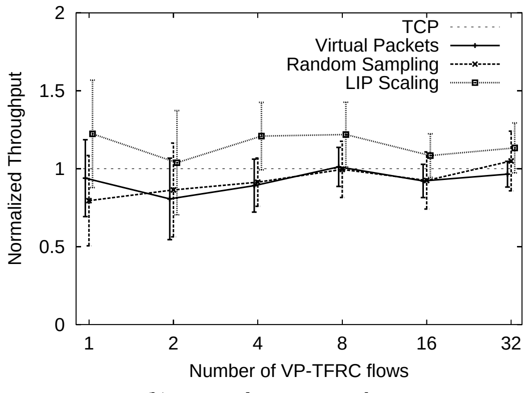
In Figures 14 to 16, we show the throughput of the different VP-TFRC variants, normalized to TCP throughput, together with the standard deviation. The decision to drop a packet at the bottleneck is based only on the number of packets and not on the packet size (packet mode). As is to be expected, the fairness of the VP-TFRC variants improves when RED queuing rather drop-tail queuing is used. In the simulations, LIP scaling is usually the most aggressive of the different variants. Particularly when packet loss is correlated as in the drop-tail queue, LIP scaling is significantly too aggressive (as evidenced in the previous simulations with the Gilbert loss model). Random sampling and virtual packets perform very similar, with random sampling being somewhat more conservative in most of the simulations. As mentioned before, the slightly higher sending rate achieved by virtual packets stems from its lower variance of the loss interval estimator and the convexity of Equation (1). The increase in VP-TFRC's aggressiveness in the simulations with a packet rate of 50 packets/s compared to the ones with 160 packets/s is largely due to the reduced buffer size at the congested router[7] and TCP's burstiness when sending packets, which increases the likelihood of TCP experiencing a packet drop.

Surprisingly, in contrast to the other loss measurement variants or plain TFRC, LIP scaling becomes more aggressive as the buffer size increases. When we compare the simulation results for a bottleneck with drop-tail queue and one bandwidth-delay product worth of buffering to simulations with twice the amount of buffering, random sampling and virtual packets behave like plain TFRC and become more aggressive buffer space as decreases. In contrast, LIP scaling is less aggressive (i.e., relatively fair to TCP) with a buffer size of one

[7]A fair sending rate of only 64 KBit/s for the 50 packet/s simulations results in a smaller bandwidth delay product and hence the size of the buffer is set to a smaller value.

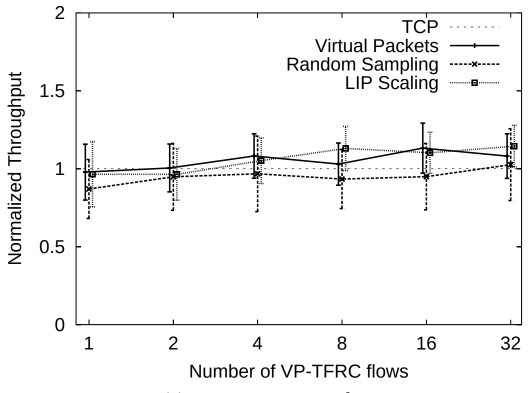


(a) RED queue in packets

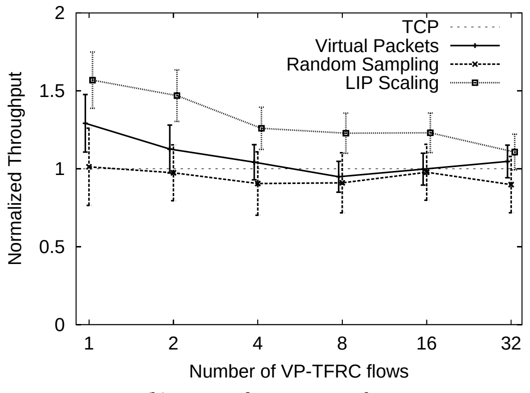


(b) Drop-tail queue in packets

Figure 14: VP-TFRC packet size 100 bytes

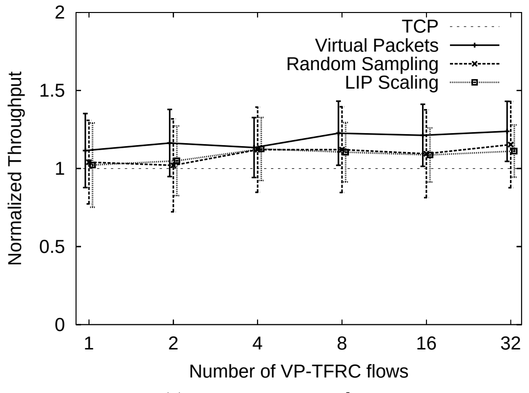


(a) RED queue in packets

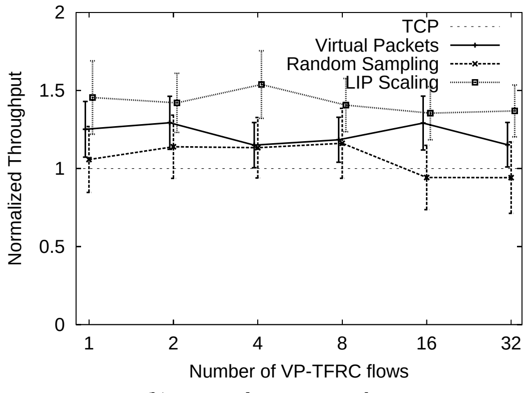


(b) Drop-tail queue in packets

Figure 15: VP-TFRC packet rate 160 packets/second

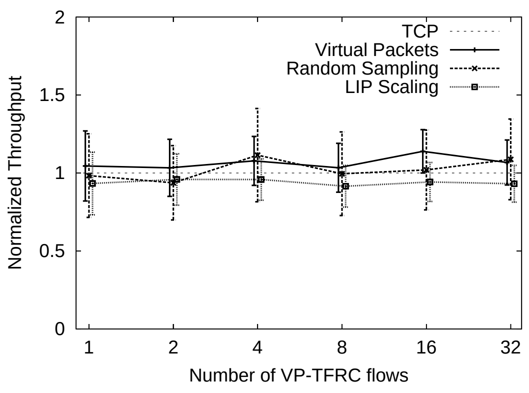


(a) RED queue in packets

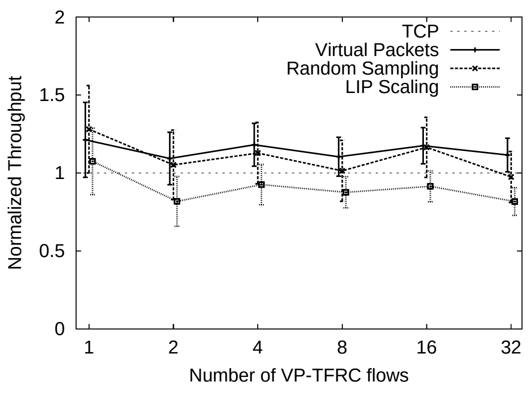


(b) Drop-tail queue in packets

Figure 16: VP-TFRC packet rate 50 packets/second



(a) RED queue in packets



(b) Drop-tail queue in packets

Figure 17: VP-TFRC packet size 100 bytes, one bandwidth delay product of queuing

bandwidth-delay product as shown in Figure 17, but consistently too aggressive for larger buffer sizes. The same effect can be seen with RED queuing, although on a smaller scale.

The root of this discrepancy lies in the different timescales over which the loss measurement mechanisms operate. When the buffer size is large, the TCP flows in the simulation tend to synchronize, so that the buffer occupancy oscillates and periods of no packet loss alternate with periods of very high packet loss. We can observe two effects that counterbalance each other:

- For random sampling and virtual packets, the number of packets within the loss interval that comprises the non-congested period is comparable to that of TCP. During the same time interval, LIP scaling will experience a loss interval that is η times larger.
- Since TCP backs off within the time frame of one RTT, virtual packets and random sampling will experience (at most) one loss event per congested period. During the same time frame, a flow with LIP scaling may experience up to η loss events, given a sufficiently high packet drop rate.

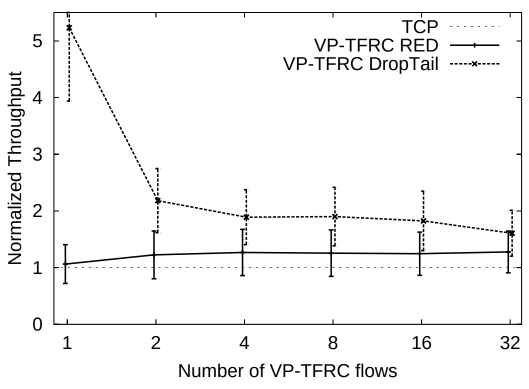
Under such circumstances, under LIP scaling the size of the loss intervals is no longer independent of η and although the two effects tend to counterbalance each other, they will usually not cancel each other out. This phenomenon can be observed whenever the loss process is time-driven instead of packet-driven.

4.2.3 Byte Mode

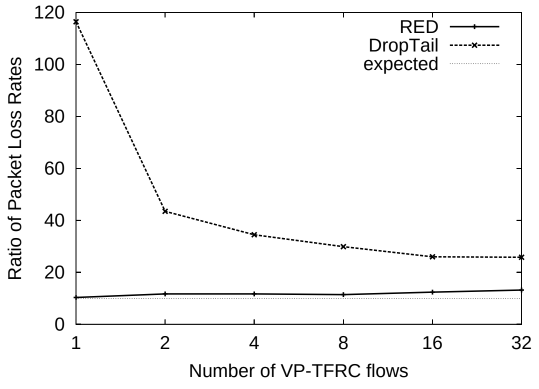
Only the virtual-packet method also works in an environment where the packet drop probability is proportional to the packet size. In Figure 18 we show how the virtual-packet method performs for RED gateways in byte mode as well as for drop-tail gateways with a queue measured in bytes. As for the previous simulations, we depict VP-TFRC throughput normalized to TCP throughput for both types of gateways. Furthermore, we show the ratio of packet drop rates experienced by flows with small and large packets (1) as measured during the simulations and (2) as would be expected if the drop probability were proportional to the packet size.

As in the previous experiments, we observe that fairness towards TCP is significantly higher for RED queues in byte mode than for drop-tail queues, but here the discrepancy is much larger. While with RED queues the packet drop probability is explicitly set proportional to the packet size, for drop-tail queues the ratio of packet drop probabilities for flows with large and small packets depends on the distribution of queue occupancy. Only if the probability of a small packet's fitting into the queue is a factor of η larger than the probability of a large packet's fitting in, will the primary assumption of proportional packet drop rates made in the design of the algorithm be met.

As can be seen from Figure 18(b), particularly at low levels of statistical multiplexing small packets have a higher-than-proportional probability of fitting into the drop-tail queue. Consequently, VP-TFRC achieves a

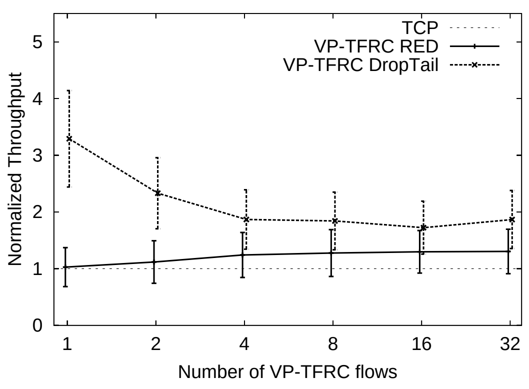


(a) Normalized Throughput

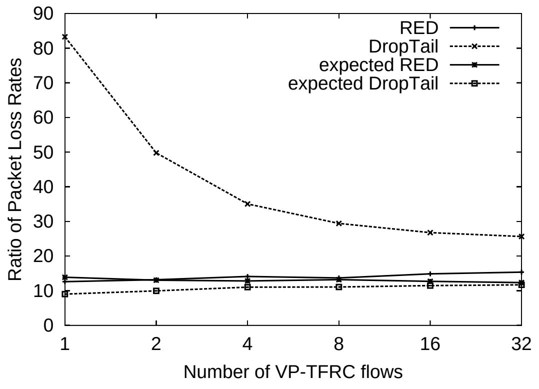


(b) Ratio of Packet Loss Rates (p_S/p_s)

Figure 18: Fairness with byte mode (VP-TFRC packet size 100 bytes)

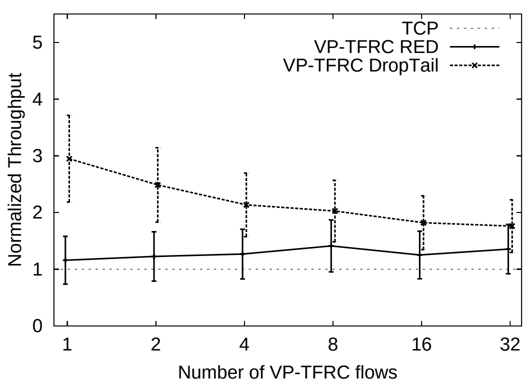


(a) Normalized Throughput

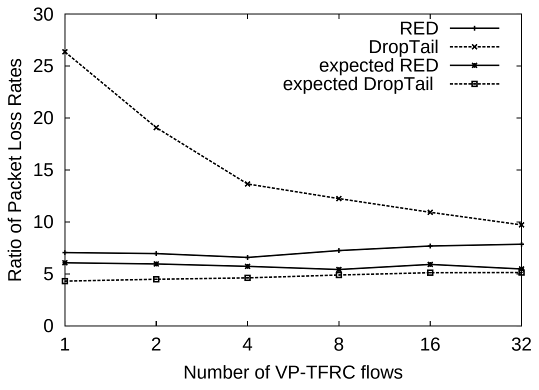


(b) Ratio of Packet Loss Rates (p_S/p_s)

Figure 19: Fairness byte mode (VP-TFRC packet rate 160 packets/second)



(a) Normalized Throughput



(b) Ratio of Packet Loss Rates (p_S/p_s)

Figure 20: Fairness byte mode (VP-TFRC packet rate 50 packets/second)

throughput of roughly twice the TCP throughput (except at very low levels of statistical multiplexing, where this ratio is even worse). In contrast, with RED in byte mode, a high level of fairness is achieved. While VP-TFRC is not exactly sending at the same rate as TCP, at around 25% the deviation is comparable to the packet-mode case. The simulations with a packet rate of 160 packets per second and 50 packets per second and a variable packet size lead to comparable results.

To further analyze the relationship of packet size and drop probability in drop-tail queues, we performed a number of simulations with constant-bitrate flows and TFRC flows with different packet sizes and compared the packet drop rates. From these simulations we take that independent of the absolute value of the packet drop rate and the exact simulation setup, the ratio of packet drop rates is close to $p_S = 2\eta\, p_s$, rather than to $p_S = \eta\, p_s$ as for RED in byte mode (see Appendix A.1). A variant of virtual packets in byte mode based on this ratio performs much better in a drop-tail environment, as shown in Figure 21.

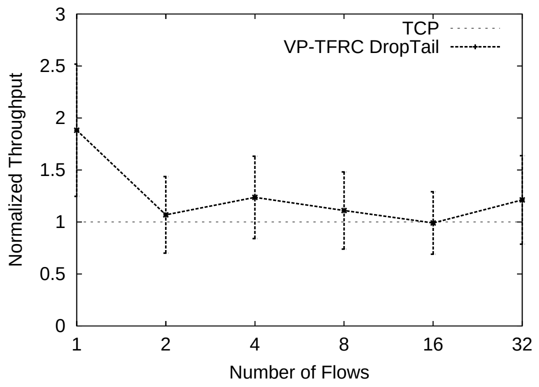


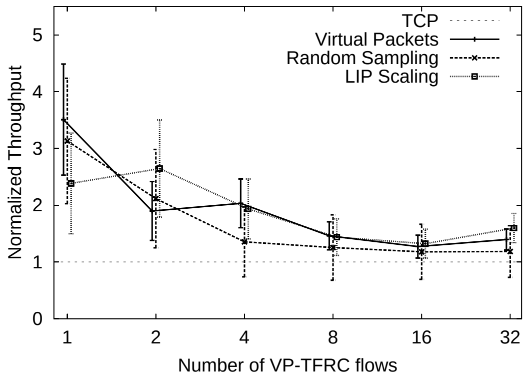
Figure 21: Modified byte mode for $p_S = 2\eta\, p_s$ (VP-TFRC packet size 100 bytes)

Hence, with the proper dependency of packet drop probability and packet size in drop-tail queues in bytes, the virtual-packet method can be adjusted so that even with such queues VP-TFRC flows and TCP flows share bandwidth in a fair manner. Nevertheless, the correction by a factor of 2 presented here is based only on simulation results; a more solid understanding, perhaps with mathematical modeling, is required before making any final conclusion.

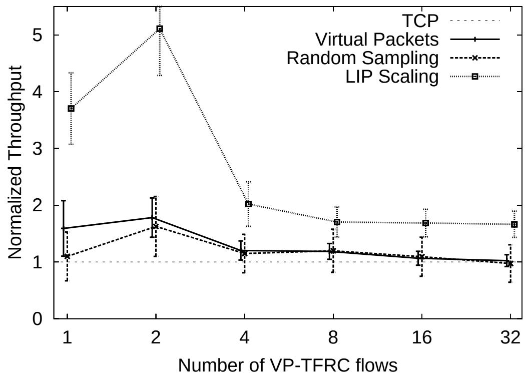
4.2.4 Further Simulations

Similar simulations with a bandwidth-limited bottleneck were carried out with a fixed number of 32 TCP flows and 1 to 32 VP-TFRC flows. Generally, these simulations (not shown here) lead to results comparable to the simulations with 16 or 32 TCP and VP-TFRC flows. The degree of fairness is more related to the level of statistical multiplexing than to the exact traffic mix at the bottleneck.

Simulations with a lower RTT lead to a decrease in fairness when only few flows compete at the bottleneck. This lower RTT is achieved by reducing the delay of the access links to 5 ms and that of the bottleneck link to 10 ms. Since we set the queue size at the bottleneck router proportional to the bandwidth-delay product, the size of the queue decreases. At the same time, the packet drop rate has to increase to compensate for the drop in RTT, since the bandwidth available for each flow remains the same. In contrast to the previous simulations, flows experience fairly high drop rates around 10% to 20%. Under these conditions, TCP performance is slightly worse and for low levels of statistical multiplexing we obtain a lower fairness ratio for VP-TFRC. This effect is shown in Figures 22 to 24. While the byte mode simulations with a fixed packet size are comparable to those with a higher RTT, the simulations for byte mode with a fixed packet rate of 160 packets/s (Figure 24(b)) show a significantly too aggressive VP-TFRC. Depending on the level of statistical multiplexing, VP-TFRC achieves a throughput of 2 times to 6 times the TCP throughput with RED queues. As is to be expected, results are significantly worse for drop-tail queues (the corresponding line is outside the y-range of the graph). There, VP-TFRC achieve a throughput of 20 to 70 times TCP throughput for

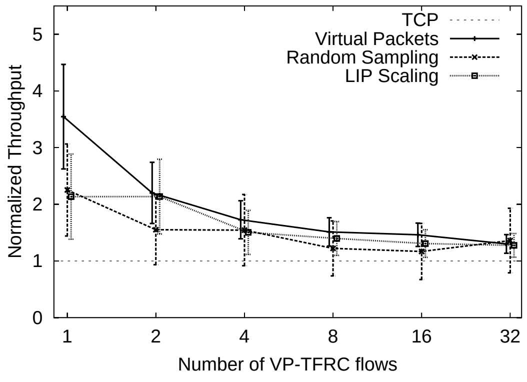


(a) RED queue in packets

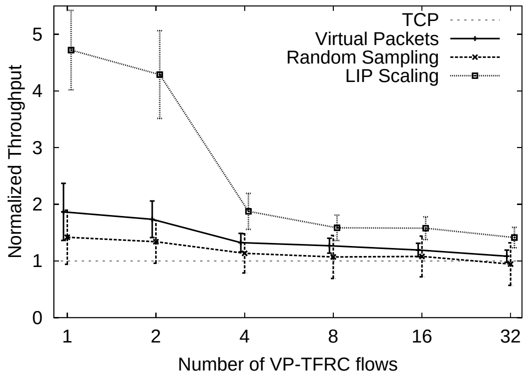


(b) Drop-tail queue in packets

Figure 22: Low RTT (VP-TFRC packet size 100 bytes)

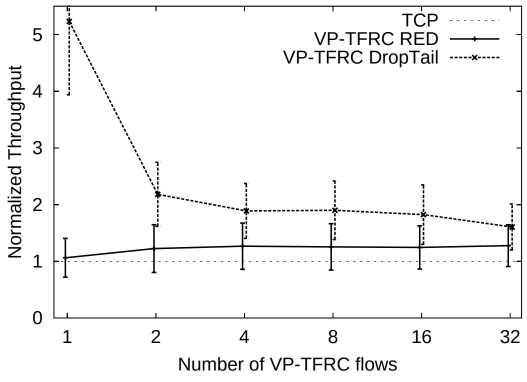


(a) RED queue in packets

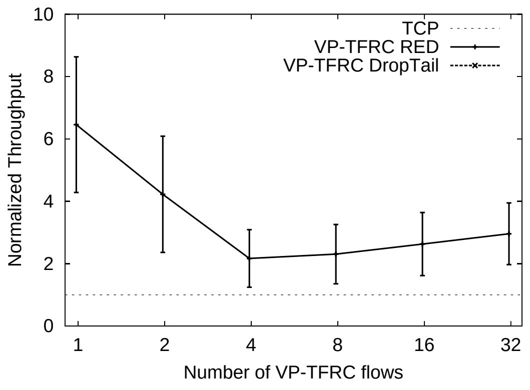


(b) Drop-tail queue in packets

Figure 23: Low RTT (VP-TFRC packet rate 160 packets/second)

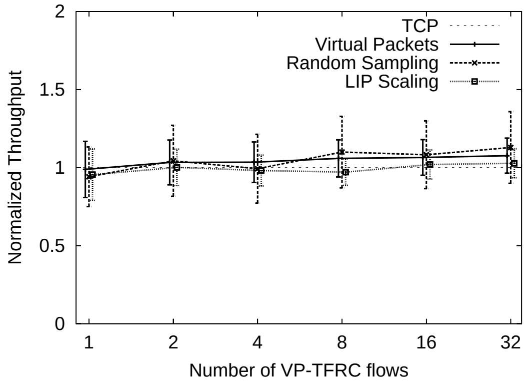


(a) VP-TFRC packet size 100 bytes

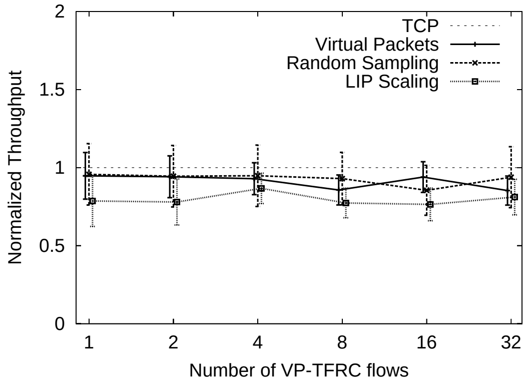


(b) VP-TFRC packet rate 160 packets/second

Figure 24: Low RTT (with VP-TFRC in byte mode)

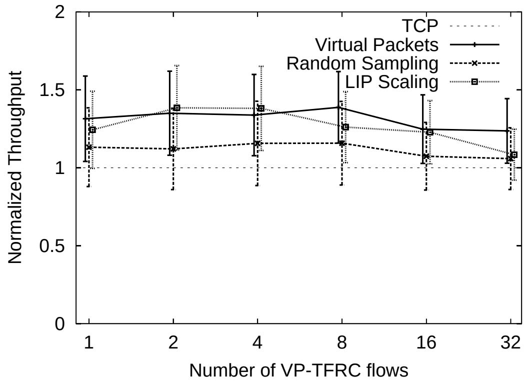


(a) RED queue in packets

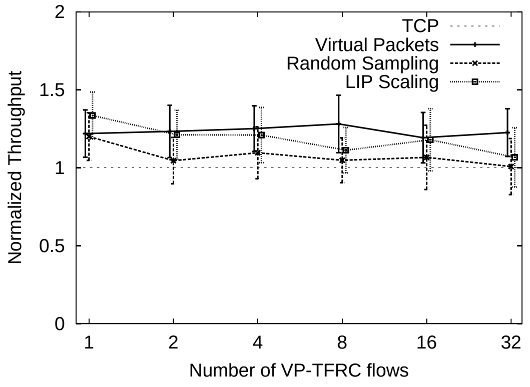


(b) Drop-tail queue in packets

Figure 25: Low RTT and high capacity (VP-TFRC packet size 100 bytes)

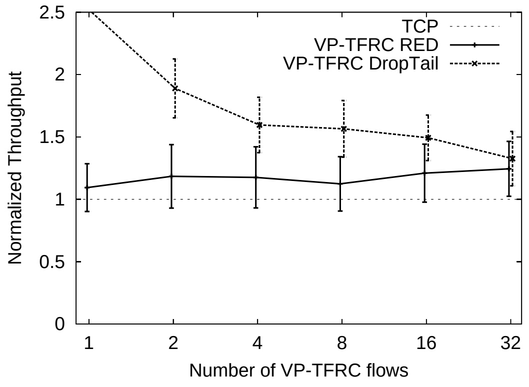


(a) RED queue in packets

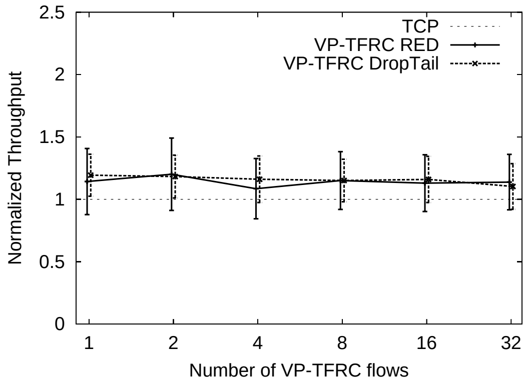


(b) Drop-tail queue in packets

Figure 26: Low RTT and high capacity (VP-TFRC packet rate 160 packets/second)



(a) VP-TFRC packet size 100 bytes



(b) VP-TFRC packet rate 160 packets/second

Figure 27: Low RTT and high capacity (with VP-TFRC in byte mode)

high levels of statistical multiplexing and and effectively shuts out the TCP flows for low levels of statistical multiplexing. The reason lies again in the combination of a relatively small buffer available to TCP and VP-TFRC's high packet rate of 160 packets/second. When the same simulations are run with a buffer size of 4 times the bandwidth delay product, the byte mode results are comparable to the relatively good results obtained with a higher RTT.

When we also increase the available bandwidth per flow in conjunction with a decrease of the RTT, the simulations give very good fairness results, as shown in Figures 25 to 27. In these simulations, the fair bandwidth is set to 800 KBit/s per flow. Particularyl the simulations with a fixed packet size of 100 byte achieve almost perfect fairness to TCP. Also the fixed packet rate and byte mode simulations show very good results. Here, the large number of packets per RTT smoothes loss rate measurement for VP-TFRC and due to the relatively large congestion window of TCP, there are usually enough packets in flight to prevent TCP timeouts, leading to reatively stable network conditions.

4.3 Failure Cases

There are a number of different failure cases when the method used for the loss measurement does not match the type of bottleneck. We need to differentiate between two different cases.

1. The byte mode version of the mechanisms is used with a bandwidth limited bottleneck in packet mode or vice versa.
2. The assumption that the bottleneck is bandwidth limited does not hold.

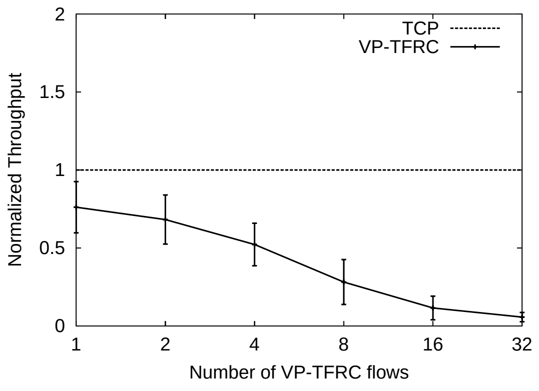
In the first case it is easy to predict the expected behavior. When a packet mode mechanism is used with a byte mode bottleneck, packet loss that should be treated as separate loss events will be aggregated, leading to a too low estimate of the loss event rate. On average (ignoring the effect of aggregation of packet loss within a RTT)), the loss intervals will be too large by a factor of S/s, leading to a sending rate that exceeds the rate of TCP roughly by a factor of $\sqrt{S/s}$. Conversely, a byte mode mechanisms in conjunction with a packet mode bottleneck results in much too conservative protocol behavior, with a sending rate too low by about the same factor. While the latter may render the congestion control mechanism useless, the former is much more dangerous since it can harm the network itself.

The methods put forward in this paper assume that the bottleneck is bandwidth limited, since otherwise a tradeoff between packet size and packet rate is not possible. In case the bottleneck is in fact packet rate limited, the proposed modifications to the loss measurement mechanisms will result in the VP-TFRC flows consuming more than their fair share of the limited resources.

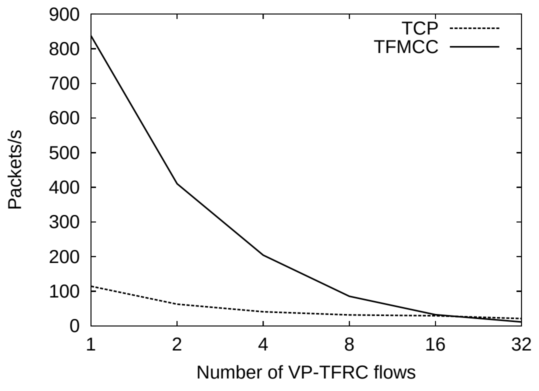
In case the packet mode version of the mechanisms is used the expected results (not depicted here) are fairly straightforward. VP-TFRC flows using a small packet size s will achieve the roughly same throughput as a TCP flow with large packets of size S and consequently, VP-TFRC's packet rate will be a factor of S/s higher than TCP's packet rate. Bottleneck resources will be shared in a very unfair manner. It is therefore not recommended to use the packet mode version of the mechanisms unless it is known that the bottleneck is bandwidth limited *and* packets are dropped irrespective of their size.

An analysis of the behavior of the byte mode version in conjunction with a packet rate limited bottleneck is more interesting. A packet rate limited bottleneck does not discriminate between packets of different size when dropping packets. Therefore, due to the aggregation of packets, a flow sending small packets will see much smaller loss intervals than a flow with large packets. This counterbalances the effect of an overproportional packet rate at the rate limited bottleneck.

When simulating a VP-TFRC flow with a fixed packet size of 100 bytes and a TCP flow with the setting discussed above, the VP-TFRC packet rate greatly exceeds TCP's packet rate for levels of statistical multiplexing, as shown in Figure 28. Since the queue is measured in packets, the high packet rate of the VP-TFRC flows leave to little buffer space for the TCP flows to arrive at the proper sending rate. However, for higher levels of statistical multiplexing, this effect is mitigated and for 16 flows and more, the VP-TFRC packet rate is less or equal to TCP's packet rate. (In these simulations, VP-TFRC flows always consume less bandwidth than TCP flows but in any case, bandwidth is not the limited resource).



(a) Throughput



(b) Packetrate

Figure 28: Packet rate limited bottleneck (VP-TFRC in byte mode with packet size 100 bytes)

Of course, if VP-TFRC modifies the packet size but maintains a fixed packet rate, it is unresponsive to congestion in case of a packet rate limited bottleneck.

5 Conclusions

In this paper we analyzed the impact of variations in packet size on equation-based congestion control. We presented four methods to allow flows sending at a packet rate different from that of TCP to estimate a TCP-friendly rate. Without these modifications, the loss event rate measured by equation-based congestion control mechanisms depends to a large degree on the packet-sending rate. Protocol behavior is either much too conservative with an unmodified TFRC or too aggressive if flows simply compensate a packet size smaller than TCP's by a proportional increase in their packet-sending rates. Two of the proposed mechanisms, random sampling and virtual packets, perform well over a wide range of different network conditions if the network drops packets independent of their size. While random sampling has the same responsiveness as an unmodified TFRC, virtual packets achieves a smoother sending rate that may be particularly desirable for the type of application requiring this modified TFRC congestion control. The smoothness of the sending rate results in a slightly higher throughput compared to plain TFRC in environments where the network conditions vary a lot. Furthermore, virtual packets also works in environments where the packet drop probability is proportional to the packet size, as in RED in byte mode. We expect these mechanisms to behave well in real-world environments.

To be able to deploy the mechanisms proposed in this paper, it is important to know the types of bottlenecks to be expected in the network. To gain an insight into which of the different mechanisms is appropriate for the Internet, measurements that analyze the relationship of packet drop rates and packet sizes are necessary. The type of bottleneck (and therefore the aforementioned relationship) is likely to differ depending on where in the network the bottleneck is located (i.e., in the backbone or close to the edge). Getting reasonably accurate information about the characteristics of bottlenecks is the object of parallel, ongoing work. This information depends largely on the type of routers and switches likely to be deployed at network bottlenecks. Once this information is available, the performance of the proposed mechanisms will have to be tested in a real-world environment.

A number of issues remain. The current implementation of virtual packets for byte mode works well together with RED. We have studied the difference in packet drop rates for RED in byte mode and drop-tail queues measured in bytes and from these results can design a byte-mode variant of virtual packets which provides a fair sharing of bandwidth in the case of drop-tail queues. However, the results should be confirmed by an analytical model of the drop probabilities experienced in such a queue to be better able to adjust the virtual-packet mechanism.

So far, the simulations are all based on the common dumbbell topology. Deeper insight into the behavior of the different mechanisms can be gained by also analyzing more complex simulation topologies such as the parking lot scenario, scenarios with more than one bottleneck, scenarios with bottlenecks that are (partly) packet-rate limited, etc.

References

[1] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, Internet Engineering Task Force, Apr. 1999.

[2] J.-C. Bolot and A. V. Garcia. Control mechanisms for packet audio in the internet. In *IEEE Infocom'96*, Mar. 1996.

[3] J.-C. Bolot, S. F. Parisis, and D. Towsley. Adaptive FEC-based error control for internet telephony. In *Infocom'99*, Mar. 1999.

[4] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, Internet Engineering Task Force, 1998.

[5] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[6] J. Byers, G. Horn, M. Handley, M. Luby, W. Shaver, and L. Vicisano. More thoughts on reference simulations for reliable multicast congestion control schemes. Notes from a meeting at Digital Fountain, Aug. 2000.

[7] S. D. Cnodder, O. Elloumi, and K. Pauwels. RED behavior with different packet sizes. In *Proc. Fifth IEEE Symposium on Computers and Communications*, Antibes-Juan les Pins, France, July 2000.

[8] V. Firoiu and M. Borden. A study of active queue management for congestion control. In *Proc. IEEE Infocom*, Tel-Aviv, Israel, Mar. 2000.

[9] S. Floyd. RED: Discussions of setting parameters. http://www.icir.org/floyd/REDparameters.txt, Nov. 1997.

[10] S. Floyd. RED: Discussions of byte and packet modes. http://www.icir.org/floyd/REDaveraging.txt, Mar. 1997 (with additional comments from January 1998 and October 2000).

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

[12] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED. *under submission*, Aug. 2001.

[13] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. ACM SIGCOMM*, pages 43 – 56, Stockholm, Sweden, Aug. 2000.

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

[15] B. Goode. Voice over internet protocol (VoIP). *Proceedings of the IEEE*, 90(9):1495–1517, Sept. 2002.

[16] P. Gupta. *Algorithms for Routing Lookups and Packet Classification*. PhD thesis, Stanford University, Dec. 2000.

[17] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Note sent to the end2end-interest mailing list, Jan. 1997.

[18] J. C. Mogul and S. E. Deering. Path MTU discovery. RFC 1191, Internet Engineering Task Force, Nov. 1990.

[19] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, Apr. 2000.

[20] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. In *Proc. IEEE Infocom*, New York, June 2002.

[21] S. Ramesh and I. Rhee. Issues in TCP model-based flow control. Technical Report TR-99-15, Department of Computer Science, NCSU, 1999.

[22] S. Sikka and G. Varghese. Memory-efficient state lookups with fast updates. In *Proc. ACM SIGCOMM*, Sept. 2000.

[23] M. Vojnovic and J. Y. Le Boudec. On the long-run behavior of equation-based rate control. In *Proc. ACM SIGCOMM*, Pittsburgh, Aug. 2002.

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

Appendix

A Queuing

The congestion signals flows receive depend to a large degree on the queuing scheme that is utilized at the bottleneck. The predominant queuing scheme in the Internet is drop-tail queuing. Packets are enqueued at the tail of the router queue, as long as buffer space is available. If the maximum queue length is reached, arriving packets are dropped, until packets from the queue are transmitted and buffer space is freed. Depending on the specific buffer management used, the size of the queue can be

- a certain number of packets (if a packet occupies one bin in the queue independently of its actual size),
- a certain number of bytes (if packets are queued in bins of exactly the packet size), or
- a combination of both (if there is a fixed number of bin sizes and packets are placed the smallest free bin that can hold the packet, or if packets are spread over a number of small bins of fixed size).

Despite its wide deployment, drop-tail queuing has significant drawbacks. Drop-tail queues signal congestion only when the queue is full and under some circumstances, queue space can be distributed very unequally among competing flows [4].

Through active queue management, routers are better able to control the queue by proactively dropping packets before the buffer overflows, the most prominent example being Random Early Detection (RED) [14]. RED maintains an exponentially weighted moving average of the queue size and drops packets probabilistically, based on the average queue size being between certain thresholds. Below a minimum threshold, no packets are dropped, between the minimum threshold and a maximum threshold the drop probability varies between 0 and a configurable maximum drop probability, and above the maximum threshold all packets are dropped.[8]

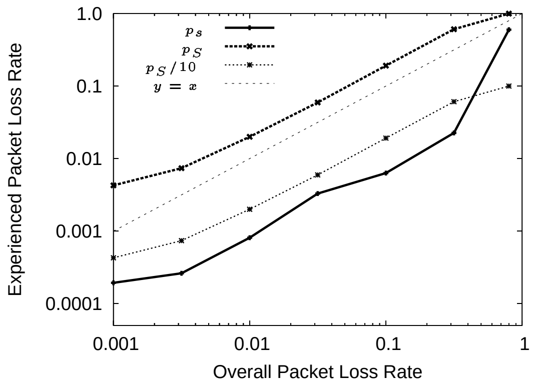
As with drop-tail queues, the queue size can be measured either in bytes or in packets. Furthermore, it is possible to base the drop probability on the size of the packet to be dropped.

[8]A more "gentle" RED variant which drops packets less aggressively when the average queue size exceeds the maximum threshold is described in [8].

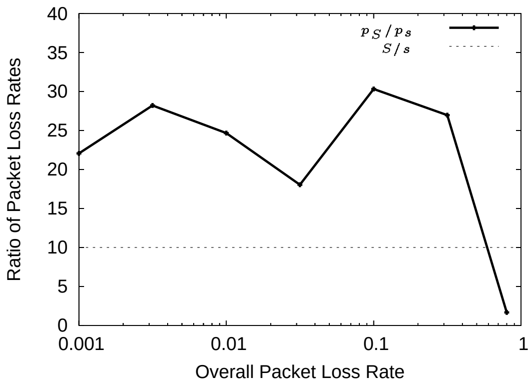
A.1 Drop-Tail Packet Drop Probabilities

When only a few flows compete at a bottleneck with a drop-tail queue in packets, the queue occupancy (and thus the packet drop probability) depends to a large degree on the recent "drop history" and the consequent changes in the sending rates of the flows. However, if the number of flows sharing the bottleneck is high, the packet drop probability is relatively independent from the drop history and flows will experience a full queue with roughly the same probability. The number of packet drops a flow can expect to see only depends on the rate at which it samples the bottleneck (i.e., the packet rate) and is independent from the packet size. Also the packet drop pattern depends on the level of statistical multiplexing. With few flows sharing the bottleneck, flows experience bursts of packet drops when the queue is full until the flows react to the congestion (usually about one RTT later). No packets are lost inbetween those intervals of very high packet drop rates. For higher numbers of flows, the bottleneck queue tends to be close to full most of the time and the distinct pattern of packet drops is lost (although also large numbers of flows may synchronize).

When the queue size is measured in bytes rather than packets, small packets have a higher probability of fitting into a nearly full queue. This reduces the number of packet drops for flows with small packets, compared to a queue measured in packets. Under the assumption that each queue occupancy is equally likely, a small packet of size s is S/s times less likely to be dropped than a large packet of size S. However, as discussed before, the bottleneck queue tends to be full rather than empty. If small packets also arrive at a higher rate than large packets (to achieve the same throughput), the queue occupancy distribution is much less favorable for flows with large packets.



(a) Experienced packet loss rates



(b) Ratio of packet loss of large packet CBR flow to small packet CBR flow

Figure 29: CBR flows with drop-tail queue in bytes

While a mathematical analysis of the drop probabilities for packets with different sizes in a queue measured in bytes is complex, simulations already provide some insight into the bias introduced by the different packet sizes. For the graph shown in Figure 29(a), two CBR flows with the same bitrate and packet sizes of 1000 bytes and 100 bytes respectively were run over the same bottleneck. The combined sending rates are slightly higher than the bottleneck bandwidth so as to produce the average packet drop rate indicated on the x-axis. On the y-axis, we plot the packet drop rate experienced by the different flows. In case the queue occupancy were uniformly distributed, we would expect to see a ten times lower drop rate ($p_S/10$) for the flow with small packets than for the flow with large packets (p_S). However, the drop rate for the small flow (p_s) is much lower than that. From Figure 29(b) we can see that p_s is in fact too low by a factor of roughly 2.5, unless the drop rate is close to 100%.

The way the simulation is set up, the queue is close to full all the time, which results in a distribution of queue occupancy that is exceedingly favorable for flows with small packets. When the buffer utilization varies over a larger range of values (as is to be expected when the queue occupancy is driven by TCP's AIMD), we expect the bias to be somewhat less pronounced.

A.2 RED Packet Drop Probabilities

As specified in [14], the RED's average queue size avg is calculated from the current queue occupancy q using an exponentially weighted moving average

$$avg \leftarrow (1 - w_q)\, avg + w_q\, q$$

The current packet drop probability p_b varies linearly with the average queue size

$$p_b \leftarrow max_p\, (avg - min_{th})/(max_{th} - min_{th})$$

where max_p is the maximum value for the drop probability and max_{th} and min_{th} are the maximum and minimum queue threshold, respectively. This drop probability is not used directly but transformed in order to provide uniformly distributed packet drops (or more specific a uniform distribution of the number of packets between packet drops).

$$p_a \leftarrow p_b/(1 - count \cdot p_b)$$

Here, $count$ is the number of transmitted packets (i.e., packets that were not dropped) since the last dropped packet. Arriving packets are then dropped with the probability p_a, if the average queue size is between the two thresholds. The drop probability is independent of the size of the incoming packet. A discussion of reasonable values for the RED parameters used above can be found in [9, 12].

The queue thresholds and the queue occupancy can either be measured in packets or in bytes. Furthermore, the packet drop probabilities can be modified accordingly such that large packets are more likely to be dropped than small packets. The author of [10] recommends to use RED in byte mode when bottlenecks are bandwidth limited. For RED in byte mode, the final packet drop probability has to be weighted by the ratio of the size of the current packet s_i to a maximum packet size S

$$p_a \leftarrow \frac{p_b}{(1 - count \cdot p_b)} \frac{s_i}{S}$$

and $count$ needs to be increased by the appropriate fraction of a large packet

$$count \leftarrow count + \frac{s_i}{S}$$

as specified in [7]. In this case, the probability distribution of the number of packets between packet drops is

$$P(L = m) = \begin{cases} 0 & \text{if } \sum_{i=1}^{m} s_i/S > 1/p_b \\ p_b \dfrac{s_m}{S} & \text{if } \sum_{i=1}^{m} s_i/S \leq 1/p_b \end{cases}$$

where s_i is the size of the i^{th} incoming packet after a drop. While this is the distribution seen at the router itself, a single flow will usually experience a different distribution of the number of packets between drops. Only if there is a single flow on the bottleneck link will it experience the same uniform distribution. For a sufficiently high level of statistical multiplexing, the drop probability for a packet of a flow is independent from the packet loss the flow experienced earlier. Therefore, the distribution will be resemble more and more a geometric distribution as the level of statistical multiplexing increases.

B Analysis

In the following, we will derive expected values for the loss intervals measured by the virtual packet and the random sampling methods. We also briefly analyze how to modify the loss measurement mechanisms in case of a channel where the packet drop probability is derived from a per-byte drop probability (as wouold be the case with a wireless channel). LIP scaling is not discussed since it essentially follows the original TFRC lossmeasurement mechanism.

B.1 Modifi cations to the Loss Measurement Mechanism (Packet Mode)

The aim of the modifications to the loss measurement mechanisms presented in this section is to estimate $E(\theta_n) = N - 1 + \frac{1}{p}$, given that the flow actually sends $N\eta$ packets per RTT. For gateways in packet mode, the packet drop probability p is the same for flows sending large packets and flows sending small packets.

Under the assumption of Bernoulli losses, we will show analytically that with these modified loss measurement algorithms, a flow sending $N\,\eta$ ($\eta \geq 1$) smaller packets of size $s = S/\eta$ per round-trip time will estimate the same average loss interval as a flow with packets of size S, no matter the value of η.

The expected loss interval obtained with the unmodified loss measurement mechanism is derived as follows:

$$\begin{aligned}
E(\theta_n) &= \sum_{m=0}^{\infty} m\,P(\theta_n = m) \\
&= \sum_{i=0}^{\infty}(N+i)(1-p)^i p \\
&= Np\sum_{i=0}^{\infty}(1-p)^i + p\sum_{i=0}^{\infty} i(1-p)^i \\
&= N + \frac{1-p}{p} \\
&= N - 1 + \frac{1}{p}
\end{aligned}$$

B.1.1 Virtual Packets

The main idea of a loss measurement mechanism based on virtual packets is to combine small packets of size s to packets of size S. Whenever a receiver receives S or more bytes (in packets of size s), it records the arrival of a virtual packet. Similarly, a virtual packet is marked as lost, when the amount of bytes lost exceeds S.

A flow sending packets of size s experiences a loss event as soon as η packets are lost since the end of the last LIP. θ_n is defined over the set $\{N + \frac{i}{\eta}, i \in \mathbb{N}_0\}$ and its probability mass function is given by:

$$P(\theta_n = m) = \begin{cases} 0 & \text{if } m < N \\ \binom{i+\eta-1}{\eta-1}(1-p)^i p^\eta & \text{if } m = N + \frac{i}{\eta}\ ,\ i \in \mathbb{N}_0 \end{cases} \qquad (15)$$

Based on Equation 15, the expected loss interval measured with the virtual packets algorithm is given by:

$$\begin{aligned}
E(\theta_n) &= \sum_{i=0}^{\infty}(N + \frac{i}{\eta})P(\theta_n = N + \frac{i}{\eta}) \\
&= N + \sum_{i=0}^{\infty} \frac{i}{\eta}\, P(\theta_n = N + \frac{i}{\eta}) \\
&\qquad \text{because} \sum_{i=0}^{\infty} P(\theta_n = N + \frac{i}{\eta}) = \frac{1}{p^\eta}\, p^\eta = 1 \\
&= N + \sum_{i=0}^{\infty} \frac{i}{\eta} \binom{i+\eta-1}{\eta-1}(1-p)^i p^\eta
\end{aligned}$$

$$
\begin{aligned}
&= N + \frac{p^\eta}{\eta}(1-p)\sum_{i=0}^{\infty} i \binom{i+\eta-1}{\eta-1}(1-p)^{i-1} \\
&= N + \frac{p^\eta}{\eta}(1-p)\,\eta\frac{1}{p^{\eta+1}} \\
&= N - 1 + \frac{1}{p}
\end{aligned}
\tag{16}
$$

B.1.2 Random Sampling

The same effect of normalization can be achieved in a different way. Instead of aggregating small packets into large packets, the loss interval can be normalized by preventing excess packets (and excess packet losses) to be included in the loss measurement process. Consider a flow that sends packets of size s at the same bitrate and thus at a higher packet rate than a flow with packets of size S would. Upon packet arrival (or the detection of a packet loss), the receiver performs a random experiment that succeeds with the probability $\frac{s}{S}$. Only when the random experiment succeeds is the packet arrival or packet loss taken into account in the loss measurement process. For this mechanism, the same notion of LIP, loss event and loss interval as the one for the original loss measurement process can be used.

In practice, each packet of size $s = S/\eta$ has a probability $p_\eta = 1/\eta$ to be sampled by the loss measurement process. Thus, a loss interval θ_n of size m, $(m \in I\!N)$ is measured if:

- j packets are sampled within the $N\,\eta - \eta$ packets following a loss, with $0 \le j \le \min(m-1, N\eta - \eta)$
- the following $m - j - 1$ sampled packets are not lost
- the last sampled packet is lost

which translates to:

$$
\begin{aligned}
P(\theta_n = m) \;=\; & \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} {p_\eta}^j (1-p_\eta)^{N\eta-\eta-j} \\
& \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1}(1-p_\eta)^{k-(m-j-1)}(1-p)^{m-j-1} p_\eta p
\end{aligned}
\tag{17}
$$

and leads to the expected loss interval given in Equation (3).

With this method, the nth loss interval, θ_n, is defined as follows (see Section 3.3.2):

$$
\begin{aligned}
P(\theta_n = m) \;=\; & \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} {p_\eta}^j (1-p_\eta)^{N\eta-\eta-j} \\
& \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1}(1-p_\eta)^{k-(m-j-1)}(1-p)^{m-j-1} p_\eta p \\
=\; & \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p)}\right)^j \\
& \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} (1-p_\eta)^{k-(m-j-1)}(1-p)^{m-1}(1-p_\eta)^{N\eta-\eta}\, {p_\eta}^m\, p \\
=\; & (1-p_\eta)^{N\eta-\eta}\, p\,(1-p)^{m-1} \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p)}\right)^j \left(\frac{1}{p_\eta}\right)^{m-j} \quad {p_\eta}^m
\end{aligned}
$$

$$= (1-p_\eta)^{N\eta-\eta}\, p\,(1-p)^{m-1} \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \tag{18}$$

The expected loss interval is then given by:

$$\begin{aligned}
E(\theta_n) &= (1-p_\eta)^{N\eta-\eta}\, p \sum_{m=1}^{\infty} m\,(1-p)^{m-1} \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right.\\
&\qquad \left. + \sum_{m=N\eta-\eta}^{\infty} m\,(1-p)^{m-1} \sum_{j=0}^{N\eta-\eta+1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right.\\
&\qquad \left. + \left(\sum_{m=0}^{\infty} m\,(1-p)^{m-1} - \sum_{m=0}^{N\eta-\eta} m\,(1-p)^{m-1}\right) \left(1+\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right.\\
&\qquad \left. + \frac{1}{p^2}\left(1-(N\eta-\eta)(1-p)^{N\eta-\eta+1} + (N\eta-\eta+1)(1-p)^{N\eta-\eta} - 1\right) \left(1+\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right.\\
&\qquad \left. + \frac{(1-p)^{N\eta-\eta}}{p^2}(1-(N\eta-\eta)\,p) \left(1+\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta} \right] \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{m=1}^{N\eta-\eta} m\,(1-p)^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right] \\
&\qquad \underbrace{+ \frac{1+(N\eta-\eta)\,p}{p}\left[(1-p_\eta)(1-p)+p_\eta\right]^{N\eta-\eta}}_{A} \\
&= (1-p_\eta)^{N\eta-\eta}\, p \left[\sum_{j=0}^{N\eta-\eta-1} \underbrace{\sum_{m=j+1}^{N\eta-\eta} m\,(1-p)^{m-1}}_{B} \binom{N\eta-\eta}{j} \left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \right] + A
\end{aligned}$$

where B is given by:

$$B = \sum_{m=0}^{N\eta-\eta} m\,(1-p)^{m-1} - \sum_{m=0}^{j} m\,(1-p)^{m-1}$$

$$
\begin{aligned}
&= \frac{(N\eta-\eta)(1-p)^{N\eta-\eta+1}-(N\eta-\eta+1)(1-p)^{N\eta-\eta}+1}{p^2} \\
&\quad -\frac{j\,(1-p)^{j+1}-(j+1)(1-p)^j+1}{p^2} \\
&= \frac{(1-p)^{N\eta-\eta}}{p^2}[(N\eta-\eta)(1-p)-(N\eta-\eta+1)]-\frac{(1-p)^j}{p^2}[\,j\,(1-p)-(j+1)] \\
&= -\frac{(1-p)^{N\eta-\eta}}{p^2}[(N\eta-\eta)p+1)]+\frac{(1-p)^j}{p^2}(jp+1)
\end{aligned}
$$

thus

$$
\begin{aligned}
E(\theta_n) &= A-\frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)]\sum_{j=0}^{N\eta-\eta-1}\binom{N\eta-\eta}{j}\left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j \\
&\quad +(1-p_\eta)^{N\eta-\eta}\sum_{j=0}^{N\eta-\eta-1}\binom{N\eta-\eta}{j}j\left(\frac{p_\eta}{1-p_\eta}\right)^j \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p}\sum_{j=0}^{N\eta-\eta-1}\binom{N\eta-\eta}{j}\left(\frac{p_\eta}{1-p_\eta}\right)^j \\
&= A-\frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)] \\
&\qquad \left[\sum_{j=0}^{N\eta-\eta}\binom{N\eta-\eta}{j}\left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^j-\left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta}\right] \\
&\quad +(1-p_\eta)^{N\eta-\eta}\left[\sum_{j=0}^{N\eta-\eta}\binom{N\eta-\eta}{j}j\left(\frac{p_\eta}{1-p_\eta}\right)^j-(N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p}\left[\sum_{j=0}^{N\eta-\eta}\binom{N\eta-\eta}{j}\left(\frac{p_\eta}{1-p_\eta}\right)^j-\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= A-\frac{[(1-p_\eta)(1-p)]^{N\eta-\eta}}{p}[(N\eta-\eta)p+1)] \\
&\qquad \left[\left(1+\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta}-\left(\frac{p_\eta}{(1-p_\eta)(1-p)}\right)^{N\eta-\eta}\right] \\
&\quad +(1-p_\eta)^{N\eta-\eta}\left[\frac{p_\eta}{1-p_\eta}(N\eta-\eta)\left(1+\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta-1}-(N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p}\left[\left(1+\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}-\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= A-\frac{(N\eta-\eta)p+1}{p}\left\{[(1-p_\eta)(1-p)+p_\eta]^{N\eta-\eta}-{p_\eta}^{N\eta-\eta}\right\} \\
&\quad +(1-p_\eta)^{N\eta-\eta}\left[p_\eta(N\eta-\eta)\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}-(N\eta-\eta)\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&\quad +\frac{(1-p_\eta)^{N\eta-\eta}}{p}\left[\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}-\left(\frac{p_\eta}{1-p_\eta}\right)^{N\eta-\eta}\right] \\
&= \frac{(N\eta-\eta)p+1}{p}{p_\eta}^{N\eta-\eta}+p_\eta(N\eta-\eta)-(N\eta-\eta){p_\eta}^{N\eta-\eta}+\frac{1}{p}-\frac{p_\eta^{N\eta-\eta}}{p}
\end{aligned}
$$

$$
\begin{aligned}
&= p_\eta(N\eta - \eta) + \frac{1}{p} \\
&= (N-1) + \frac{1}{p}
\end{aligned}
$$

B.2 Modifications to the Loss Measurement Mechanism (Byte Mode)

Consider now a packet loss process where the packet drop probability depends on the packet size as follows:

$$p_S = \eta\, p_s \tag{19}$$

where p_S and p_s respectively denote the probabilities for a packet of size S and s $(= S/\eta)$ to be dropped.

With RED gateways operating in byte mode the probability to drop a packet of size S is η times larger than the probability to drop a packet of size s. As a consequence, the average interval (in terms of packets) between two packet losses is roughly η times smaller for the flow sending packets of size S than for the flow sending small packets of size s. The average number of bytes between two packet losses is roughly the same for all the flows, independently of the packet size.

B.2.1 Virtual Packets

For the virtual packet algorithm operating in byte mode, a loss event is declared as soon as a packet of s bytes was lost after the LIP. The probability mass function of the random variable θ_n representing the loss interval is thus:

$$P(\theta_n = m) \quad = \quad \begin{cases} 0 & \text{if } m < N \\ (1-p_s)^{i-1} p_s & \text{if } m = N - 1 + \frac{i}{\eta}, i \in I\!N \end{cases} \tag{20}$$

and the expected loss event interval is given by:

$$E(\theta_n) = N - 1 + \frac{1}{p_S} \tag{21}$$

It is possible to construct an algorithm for RED in byte mode based on random sampling. This amounts to a random sampling of arrived data packets, whereas lost data packet always need to be taken into account. While the virtual packets and the random sampling mechanism for packet mode are both valid mechanisms of their own with slightly different properties, random sampling in byte mode merely ignores information that is available to the receiver. The number of packets between packet drops is estimated instead of being directly measured as in the virtual packet approach. Generally, we expect random sampling in byte mode to be inferior to virtual packets in byte mode. For this reason we only discuss the approach briefly in Appendix B.2.2.

Adjusting LIP scaling to byte mode results in an even less favorable mechanism. Here, the expected number of lost packets within the LIP does not increase with a decrease in packet size and therefore reducing the duration of the LIP would require introducing artificial packet loss in later intervals. We do not recommend to use LIP scaling in combination with a bottleneck in byte mode.

Using Equation 20, we compute the expected loss event interval as follows:

$$\begin{aligned}
E(\theta_n) &= \sum_{m=0}^{\infty} m\, P(\theta_n = m) \\
&= \sum_{i=1}^{\infty} (N - 1 + \frac{i}{\eta})\, P(\theta_n = N - 1 + \frac{i}{\eta}) \\
&= N - 1 + \frac{1}{\eta} \sum_{i=1}^{\infty} i\, (1 - p_s)^{i-1} p_s \\
&= N - 1 + \frac{1}{\eta}\, \frac{1}{{p_s}^2}\, p_s \\
&\quad \text{(From Equation 19)} \\
&= N - 1 + \frac{1}{p_S} \qquad (22)
\end{aligned}$$

B.2.2 Random Sampling

With this method, each packet of size $s = S/\eta$ will be sampled by the loss measurement process with a probability of:

- $p_\eta = 1/\eta$ if it is received,
- 1 if it is lost

Thus, a loss interval θ_n of size m, ($m \in I\!N$) is measured if:

- j ($0 \leq j \leq \min(m-1, N\eta - \eta)$) packets are sampled within the $N\,\eta - \eta$ packets following a loss,
- the $m - j - 1$ packets are sampled among the k packets received between the end of the LIP period and the next packet loss,
- the $k + 1$st packet following the LIP period is lost packet.

which is translated as follows:

$$\begin{aligned}
P(\theta_n = m) &= \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta - \eta}{j} {p_\eta}^j (1 - p_\eta)^{N\eta-\eta-j} \\
&\quad \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} {p_\eta}^{m-j-1} (1 - p_\eta)^{k-(m-j-1)} (1 - p_s)^k p_s \\
&= \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta - \eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^j \\
&\quad \sum_{k=m-j-1}^{\infty} \binom{k}{m-j-1} ((1-p_\eta)(1-p_s))^{k-(m-j-1)}\, (1-p_s)^{m-1} (1-p_\eta)^{N\eta-\eta}\, {p_\eta}^{m-1}\, p_s \\
&= (1-p_\eta)^{N\eta-\eta}\, {p_\eta}^{m-1}\, p_s\, (1-p_s)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta - \eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^j \left(\frac{1}{1-(1-p_\eta)(1-p_s)}\right)^{m-j} \\
&= \frac{p_s\, (1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)} \left(\frac{p_\eta(1-p_s)}{1-(1-p_\eta)(1-p_s)}\right)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta - \eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j
\end{aligned}$$

The expected loss interval is then given by:

$$\begin{aligned}
E(\theta_n) &= \underbrace{\frac{p_s\,(1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)}}_{A} \sum_{m=1}^{\infty} m \left(\underbrace{\frac{p_\eta(1-p_s)}{1-(1-p_\eta)(1-p_s)}}_{Z} \right)^{m-1} \\
&\quad \sum_{j=0}^{\min(m-1,N\eta-\eta)} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&= A \sum_{m=1}^{N\eta-\eta} m\, Z^{m-1} \sum_{j=0}^{m-1} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&\quad + A \sum_{m=N\eta-\eta+1}^{\infty} m\, Z^{m-1} \sum_{j=0}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&= \underbrace{A \sum_{j=0}^{N\eta-\eta-1} \underbrace{\sum_{m=j+1}^{N\eta-\eta} m\, Z^{m-1}}_{B} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j}_{C} \\
&\quad + A \underbrace{\sum_{m=N\eta-\eta+1}^{\infty} m\, Z^{m-1}}_{D} \left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}
\end{aligned} \tag{23}$$

where B, C and D are computed as follows:

$$\begin{aligned}
B &= \sum_{m=0}^{N\eta-\eta} m\, Z^{m-1} - \sum_{m=0}^{j} m\, Z^{m-1} \\
&= \frac{(N\eta-\eta)Z^{N\eta-\eta+1} - (N\eta-\eta+1)Z^{N\eta-\eta} + 1}{(1-Z)^2} - \frac{j\, Z^{j+1} - (j+1)Z^j + 1}{(1-Z)^2} \\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)Z - (N\eta-\eta+1)\right] - \frac{Z^j}{(1-Z)^2}\left[\, j\, Z - (j+1)\right] \\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] + \frac{Z^j}{(1-Z)^2}\left[j(1-Z)+1\right]
\end{aligned}$$

$$\begin{aligned}
C &= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right] \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)^j \\
&\quad + \frac{1}{1-Z} \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} j \left[Z\left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)\right]^j \\
&\quad + \frac{1}{(1-Z)^2} \sum_{m=j+1}^{N\eta-\eta} \binom{N\eta-\eta}{j} \left[Z\left(\frac{1}{(1-p_\eta)(1-p_s)} - 1\right)\right]^j
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left[\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}-\left(\frac{1}{(1-p_\eta)(1-p_s)}-1\right)^{N\eta-\eta}\right]\\
&\quad +\frac{1}{1-Z}\left[Y(N\eta-\eta)(1+Y)^{N\eta-\eta-1}-(N\eta-\eta)Y^{N\eta-\eta}\right]\\
&\quad +\frac{1}{(1-Z)^2}\left[(1+Y)^{N\eta-\eta}-Y^{N\eta-\eta}\right]\\
&\qquad \text{with}\quad Y=Z\left(\frac{1}{(1-p_\eta)(1-p_s)}-1\right)=\frac{1}{1-p_\eta}-1\\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left[\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}-\left(\frac{Y}{Z}\right)^{N\eta-\eta}\right]\\
&\quad -\frac{Y^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1\right]\\
&\quad +\frac{(1+Y)^{N\eta-\eta-1}}{(1-Z)^2}\left[Y(N\eta-\eta)(1-Z)+1+Y\right]\\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}\\
&\quad +\frac{(1+Y)^{N\eta-\eta-1}}{(1-Z)^2}\left[Y(N\eta-\eta)(1-Z)+1+Y\right]\\
&= -\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}\\
&\quad +\frac{1}{(1-Z)^2}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-Z)(1-p_\eta)+1\right]
\end{aligned}
$$

$$
\begin{aligned}
D &= \sum_{m=0}^{\infty} m\, Z^{m-1}-\sum_{m=0}^{N\eta-\eta} m\, Z^{m-1}\\
&= \frac{1}{(1-Z)^2}-\frac{(N\eta-\eta)Z^{N\eta-\eta+1}-(N\eta-\eta+1)Z^{N\eta-\eta}+1}{(1-Z)^2}\\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[-(N\eta-\eta)Z+(N\eta-\eta+1)\right]\\
&= \frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]
\end{aligned}
$$

thus

$$
\begin{aligned}
E(\theta_n) &= A\,C+A\,D\,\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}\\
&= -A\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}\\
&\quad +A\frac{1}{(1-Z)^2}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-Z)(1-p_\eta)+1\right]\\
&\quad +A\frac{Z^{N\eta-\eta}}{(1-Z)^2}\left[(N\eta-\eta)(1-Z)+1)\right]\left(\frac{1}{(1-p_\eta)(1-p_s)}\right)^{N\eta-\eta}
\end{aligned}
$$

$$\begin{aligned}
&= A\frac{1}{(1-Z)}\left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[Y(N\eta-\eta)(1-p_\eta)+\frac{1}{1-Z}\right] \\
&= \frac{p_s\,(1-p_\eta)^{N\eta-\eta}}{1-(1-p_\eta)(1-p_s)}\;\frac{1-(1-p_\eta)(1-p_s)}{p_s} \\
&\quad \left(\frac{1}{1-p_\eta}\right)^{N\eta-\eta}\left[\frac{p_\eta}{1-p_\eta}(N\eta-\eta)(1-p_\eta)+\frac{1-(1-p_\eta)(1-p_s)}{p_s}\right] \\
&= N-1+\frac{p_\eta}{p_s}+(1-p_\eta) \\
&= N-1+\frac{1}{\eta p_s}+\frac{\eta-1}{\eta} \\
&= N-1+\frac{1}{p_S}+\frac{\eta-1}{\eta} \qquad (24)
\end{aligned}$$

While the above loss measurement mechanism introduces a bias of $\frac{\eta-1}{\eta}$, this can easily be compensated for since η is known to the receiver.

B.3 Hypothetical RED (Channel with Byte Errors)

Consider a packet loss process where the packet drop probability is derived from a byte loss probability. In this case, a packet is lost if one or more bytes of the packet are lost. Thus, if p_S and p_s respectively denote the probabilities for a packet of size S and $s(=S/\eta)$ to be dropped, both probabilities are related as follows:

$$p_S = 1-(1-p_s)^\eta \qquad (25)$$

In the following, we discuss a loss measurement mechanism based on virtual packets that is designed for such a hypothetical packet marking or packet drop scheme.

B.3.1 Virtual Packets

When the virtual packet algorithm is used with the hypothetical RED dropping strategy, the definition of loss event remains the same as for RED in byte mode (Definition 3) but the definition of loss event interval has to be modified as follows.

Defi nition 4 *A* loss interval *is measured as the number of* entire *virtual packets received between two successive loss events, including the lost packet that ends the loss interval; the lost packet being counted as an* entire *virtual packet*[9].

Even though there is no intuitive justification for Definition 4, we show later in this section that defining the loss interval in this way allows to ensure that all flows measure the same loss event rate, no matter their packet size.

Based on Definition 4, the random variable θ_n is thus characterized as follows:

$$P(\theta_n = m) \quad = \quad \begin{cases} 0 & \text{if } m < N \\ \sum_{l=0}^{\eta-1}(1-p_s)^{\eta i+l}p_s & \text{if } m = N+i,\, i \in I\!N_0 \end{cases} \qquad (26)$$

[9]Note that counting the lost packet as an *entire* virtual packet and then rounding down (by taking the entire number of virtual packets between loss events) is equivalent to counting the lost packet as a small packet and then rounding up.

And the expected loss event interval is given by:

$$\begin{aligned}
E(\theta_n) &= \sum_{m=0}^{\infty} m\, P(\theta_n = m) \\
&= \sum_{i=0}^{\infty} (N+i) \sum_{l=0}^{\eta-1} (1-p_s)^{\eta i + l} p_s \\
&= \sum_{i=0}^{\infty} (N+i) p_s (1-p_s)^{\eta i} \sum_{l=0}^{\eta-1} (1-p_s)^{l} \\
&= \sum_{i=0}^{\infty} (N+i) p_s (1-p_s)^{\eta i} \frac{1-(1-p_s)^{\eta}}{p_s} \\
&= \sum_{i=0}^{\infty} (N+i) (1-p_S)^{i} p_S \\
&\quad \text{(From Equation 25)} \\
&= N - 1 + \frac{1}{p_S} \qquad (27)
\end{aligned}$$

In practice, θ_n as defined in Equation 26 represents the sum of (1) the number of virtual packets[10] that are ignored because they are part of the nth lip period i.e. $(N\,\eta - \eta)/\eta$ and (2) the number of 'entire' virtual packets received between the end of the LIP period and the next packet loss (including the lost packet, which, alone counts for a complete virtual packet). At first, it may seem surprising that the expected value of the loss interval defined in this way (using a truncated part of the interval until the next loss) is exactly equal to the desired expected value given in Equation 27. But this can be explained by taking a closer look at the expected number of 'entire' virtual packets received between the end of a LIP period and the next packet loss.

Let υ_n be number of virtual packets received between the end of the nth LIP period and the next packet loss (including the lost packet that counts for an entire virtual packet). $\lfloor \upsilon_n \rfloor$ is the number of 'entire' such packets and thus $\delta_n = \upsilon_n - \lfloor \upsilon_n \rfloor$ represents the truncated part of the loss interval.

We have:

$$P(\upsilon_n = 1 + \frac{m}{\eta}) = (1-p_s)^m p_s \;\;,\; m \in \mathbb{N}_0 \qquad (28)$$

$$\begin{aligned}
E(\upsilon_n) &= \sum_{m=0}^{\infty} (1 + \frac{m}{\eta})\; P(\upsilon_n = 1 + \frac{m}{\eta}) \\
&= 1 + \sum_{m=0}^{\infty} \frac{m}{\eta}\; P(\upsilon_n = 1 + \frac{m}{\eta}) \\
&= 1 + \frac{1}{\eta}\;(1-p_s) \sum_{m=0}^{\infty}\; m\;(1-p_s)^{m-1} p_s \\
&= 1 + \frac{1}{\eta}\;(1-p_s) \frac{1}{{p_s}^2} p_s \\
&= 1 + \frac{1-p_s}{\eta\, p_s} \\
&= \frac{1}{\eta\, p_s} + \frac{\eta - 1}{\eta} \qquad (29)
\end{aligned}$$

[10]Note that the virtual packets are made up of small packets.

Equation (29) shows that the fact that we count the lost packet as an entire virtual packet introduces a bias of $\frac{\eta-1}{\eta}$ in the measurement of $E(\upsilon_n)$.

Now, δ_n is defined as follows:

$$\begin{aligned}
P(\delta_n = \frac{j}{\eta}) &= \sum_{i=1}^{\infty} P(\upsilon_n = \frac{j+\eta\, i}{\eta})\ ,\ j = 0 \ldots \eta - 1 \\
&= \sum_{i=1}^{\infty} (1-p_s)^{j+\eta(i-1)} p_s \\
&= \sum_{i=0}^{\infty} (1-p_s)^{j+\eta i} p_s \\
&= p_s(1-p_s)^j \frac{1}{1-(1-p_s)^{\eta}} \\
&\quad \text{(From Equation 25)} \\
&= p_s\,(1-p_s)^j\,\frac{1}{p_S} \qquad (30)
\end{aligned}$$

The bias introduced by the truncation of the loss interval, δ_n, is thus given by:

$$\begin{aligned}
E(\delta_n) &= \sum_{j=0}^{\eta-1} \frac{j}{\eta}\, P(\delta_n = \frac{j}{\eta}) \\
&= \sum_{j=0}^{\eta-1} \frac{j}{\eta}\, p_s\,(1-p_s)^j\, \frac{1}{p_S} \\
&= \frac{p_s}{\eta p_S} \sum_{j=0}^{\eta-1} j\ (1-p_s)^j \\
&= \frac{p_s}{\eta\, p_S}\ \frac{1}{{p_s}^2}\ \{(\eta-1)(1-p_s)^{\eta+1} - \eta(1-p_s)^{\eta} + (1-p_s)\} \\
&= \frac{1}{\eta\, p_S\, p_s}\{(\eta-1)(1-p_s)(1-p_S) - \eta(1-p_S) + (1-p_s)\} \\
&= \frac{1}{\eta\, p_S\, p_s}\{p_S + (\eta-1)\, p_S\, p_s - \eta\, p_s)\} \\
&= \frac{1}{\eta\, p_s} - \frac{1}{p_S} + \frac{\eta-1}{\eta}
\end{aligned} \qquad (31)$$

Equation (31) shows that the bias introduced by the the truncation of the loss interval exactly compensates for (1) the bias introduced by the fact that the lost packet is counted as a virtual packet and (2) the fact that the number of virtual packets received between losses is smaller when sending small packets, i.e. $1/(\eta p_s)$ than when sending large packets, i.e. $1/p_S$. So that finally, we can write:

$$\begin{aligned}
E(\theta_n) &= \frac{N\,\eta - \eta}{\eta} + E(\lfloor \upsilon_n \rfloor) \\
&= N - 1 + E(\upsilon_n) - E(\delta_n) \\
&= N - 1 + \frac{1}{p_S} \qquad (32)
\end{aligned}$$