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

Jörg Widmer, Catherine Boutremans, and Jean-Yves Le Boudec
Laboratory for Computer Communications and Applications
EPFL (Swiss Federal Institute of Technology), Lausanne, Switzerland

ABSTRACT

Current TCP-friendly congestion control mechanisms 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 size, resource sharing depends on packet size and is no longer fair. 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 resulting throughput bias. We investigate the design space in detail 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 packet rate, or applications that are forced to use a small packet size.

1. INTRODUCTION

Current congestion control as used by TCP or equation-based congestion control [1] aims to adjust the number of packets sent per time interval 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.

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 [2, 3]. However, there is little queuing in backbone networks, let alone congestion-related packet drops [4]. In contrast, access networks are often bandwidth limited, and this is exacerbated by wireless access. It is thus likely that many network paths are bandwidth limited.

Applications such as "Voice over IP" (VoIP) naturally prefer to trade off packet size for packet rate. Despite the low payload efficiency, many VoIP systems use small packets corresponding to 20 or 30 ms of audio, in order to keep the end-to-end delay below 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 [5, 6]. Other applications may be driven to adjust the packet size independently of congestion control (for example: a high bit error rate on a wireless link induces a small packet size). While it can be argued that the throughput of such flows is low, not using any congestion control at all in a potentially congested environment is not acceptable [7].

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 (RTT), the loss event rate, and the packet size [1]. 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 usually 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 when the actual packet size is smaller 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.

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. 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 post-processing produce similar results.

Depending on the queuing scheme used in the routers, the packet drop probability may either be stochastically independent of the packet size (e.g., with drop-tail queues measured in packets) or not (e.g., with RED in byte mode [8]). 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 see. 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 [9] under similar network conditions.[1] 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 further give a quantitative analysis of the bias introduced into the loss measurement process when sampling the bottleneck at different rates. In Section 3, we propose modifications to end-to-end congestion control. 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 well. In Section 5, we summarize the findings and list some open issues.

[1]The aim of the proposed mechanisms is to improve protocol performance in the face of different packet sizes and therefore, an obvious first step is to avoid the negative impact of fragmentation [10] by bounding the packet size to the MTU.

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 briefly recall some fundamentals of equation-based congestion control.

Our analysis is based on the unicast TCP-friendly equation-based congestion control protocol (TFRC) [1, 11] and also applies to its multicast counterpart TFMCC [12] since each calculates a fair sending rate in a similar manner.

2.1 Foundations of Equation-Based Congestion Control

For equation-based congestion control, the sending rate is commonly determined using a model for long-term TCP throughput such as the one described in [13].

$$R_{TCP} = \frac{S}{r\left(\sqrt{\frac{2l}{3}} + \left(12\sqrt{\frac{3l}{8}}\right) l\left(1 + 32l^2\right)\right)} \quad (1)$$

The expected throughput R_{TCP} is calculated as a function of the the round-trip time r, loss event rate l, and the packet size S. A receiver measures l and r, 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 measures the frequency of *loss events*. A loss event consists of one or more packets lost within the same RTT. Since TCP commonly halves the congestion window only once in response to several losses per RTT, the loss measurement process aggregates losses within the same RTT to a single loss event. Hence, 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, called the *Loss Insensitive Period* (LIP). On average, if a rate-controlled flow sends N packets per RTT, 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 $l = 1/(\sum_i w_i \theta_{n-i})$. Generally, older loss intervals are assigned a smaller weight than new loss intervals [1].

2.2 Proportional Scaling of Throughput

Let S be the packet size for bulk data transfer (i.e., the MTU if the source uses path MTU discovery or 576 bytes, the minimum MTU that has to be supported by an Internet router). Because of the linear dependency of throughput and packet size in Equation 1, a flow sending packets of smaller size s will only achieve a fraction of the throughput of a flow with packet size S.

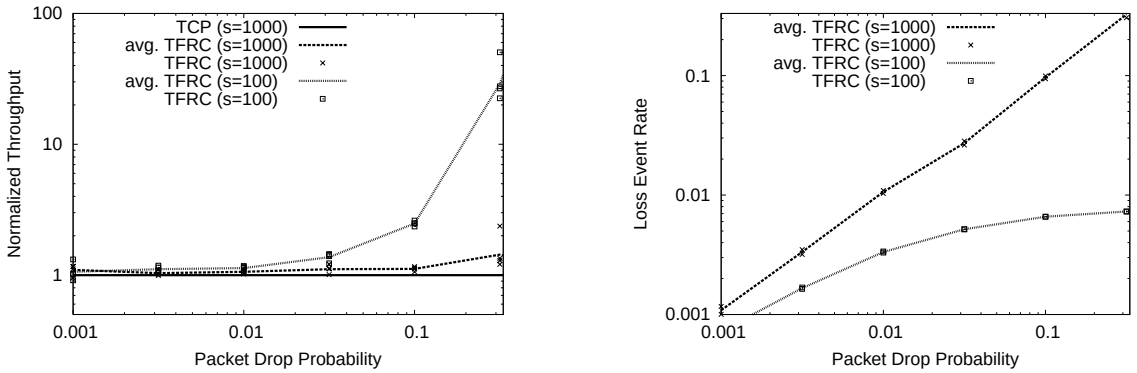


Figure 1: Normalized throughput and loss event rate for proportional scaling of throughput (per-packet drops)

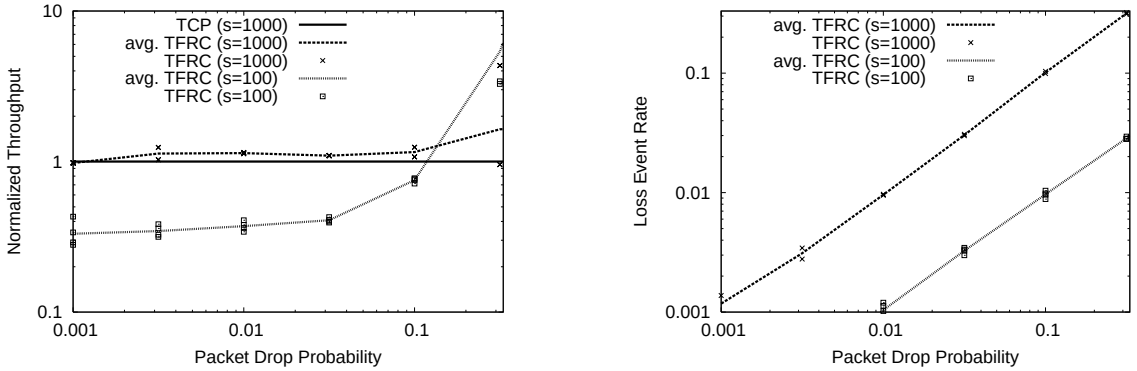


Figure 2: Normalized throughput and loss event rate for with small packets and per-byte drops

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 all other parameters remained the same, flows should achieve the same throughput independently of the packet size.

Unfortunately, this is not the case. 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, as shown in Figure 3(a). This effect was also reported in [14].

Consider a rate-controlled flow sending N packets of size S per RTT. 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 $N-1$ packets within the LIP and the number of packets between the end of the LIP and the next packet loss (including the lost packet), which we denote as X. The random variable X has a geometric distribution with parameter p; thus $E(\theta_n) = N - 1 + \frac{1}{p}$ and $Var(\theta_n) = \frac{1-p}{p^2}$. We see that the higher the number of packets per RTT, 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 RTT. If the loss measurement process remains unchanged, this flow sees $E(\theta_n) = N\eta - 1 + \frac{1}{p}$ and thus overestimates the average loss interval compared to a flow sending large packets, leading to an unfair distribution of bandwidth.

Figure 1 shows how the throughput for these types of flows varies for different packet drop rates.[2] Here, we use a packet size of 1000 bytes for the large TFRC and the TCP flow, and a packet size of 100 bytes for the small TFRC flow. The TCP flow is not depicted in the graph but achieves exactly the same throughput as the TFRC flow with large packets. As can bee seen from the graph, 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. The reason for this aggressiveness can easily be seen from the loss event rates. In contrast to the flows with large packets, the loss event rate of the flow with small packets levels off and stays below 1% even for packet drop rates close to 100%.

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.

[2]The simulation results were obtained using *ns*-2 with a random dropper with drop probabilities independent of the packet size and otherwise the same simulation parameters as in Section 4.1.1.

2.3 Network-Based Approach

Another way to reduce the discrimination against flows sending small packets is to use RED gateways in byte mode [8] instead of modifying the packet size in the throughput formula. With these gateways the packet drop probability is proportional to the packet size, so that the average number of bytes between packet drops remains the same, independently of the packet size. Figure 2 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. As expected, the loss event rates for the flows with the two packet sizes differ by a factor of 10. Due to the square-root dependency on the loss event rate, the TFRC flow with small packets achieves a throughput only a factor of $\sqrt{10}$ worse than the throughput of the large flows (whereas using RED in packet mode or drop-tail queues would result in a factor of 10). 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.

Neither a simple scaling of the sending rate nor RED in byte mode suffice to ensure fairness between flows using different packet sizes.

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. 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 real packet size s need not be fixed but can vary from one packet to the next.

The solutions depend on whether the network drops packets irrespective of their size ("Packet Mode"), or whether a scheme such as RED in byte mode is already deployed ("Byte Mode"). They are designed such that the bias due to variable packet size is removed exactly when packet drops on the end-to-end path are independent ("Bernoulli" loss model). In Section 4, we evaluate the robustness to other channel conditions by means of simulation.

3.1 Loss Measurement Postprocessing

A first naive method to improve fairness is to measure the loss interval as before, compute the bias, and then remove this bias from the measured loss interval.

3.1.1 Unbiasing (Packet Mode)

A connection sending $N\eta$ packets of size $s = S/\eta$ per RTT measures an average loss interval (in packets) $E(\theta_n) = N\eta - 1 + \frac{1}{p}$ which differs from a correct measure of the loss interval by $(\eta - 1)N$. The simplest correction is to subtract this from the measured interval. Since both sender and receiver know η and N, the correction can be carried out by either of them.

3.1.2 Unbiasing (Byte Mode)

If the packet drop probability is proportional to the packet size, the previous section is modified as follows. Assume that the drop probability for a packet of size $s = S/\eta$ is $p_s = p_S/\eta$, where p_S is the probability for a large packet to be dropped. A flow sending $N\eta$ packets of size s per RTT measures an average loss interval $E(\theta_n) = N\eta - 1 + \frac{1}{p_s}$, thus the correction is now $(\eta - 1)N + \frac{\eta - 1}{p_S}$. 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 the equation for $E(\theta_n)$ and compute the bias as $\frac{(\eta-1)(\theta_n+1)}{\eta}$. Note that here the bias depends on θ_n instead of N.

3.1.3 Discussion

Modifying the measured loss event rate is very challenging when network loss conditions are not perfectly stable. 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. 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.

3.2 Loss Measurement Modifications (Packet Mode)

In the following, we present three alternative modifications to TFRC's loss measurement mechanism. Their aim is to estimate the process θ_n that would be measured by a flow sending packets of the reference size S. In contrast to loss measurement postprocessing, these mechanisms do not require to estimate an average ratio of actual packet size to reference size but directly use the packet size of the current packet. They are therefore independent of the pattern of variation of the packet size.

For gateways in packet mode, the packet drop probability p is the same for small and for large packets. Under the assumption of Bernoulli loss, we show analytically that these modified loss measurement algorithms are unbiased estimators of θ_n. We also evaluate the variance of the different estimators to better be able to compare the various solutions and analyze their conservativeness. A more detailed analysis as well as explicit computations of the expected loss event intervals can be found in [15].

3.2.1 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

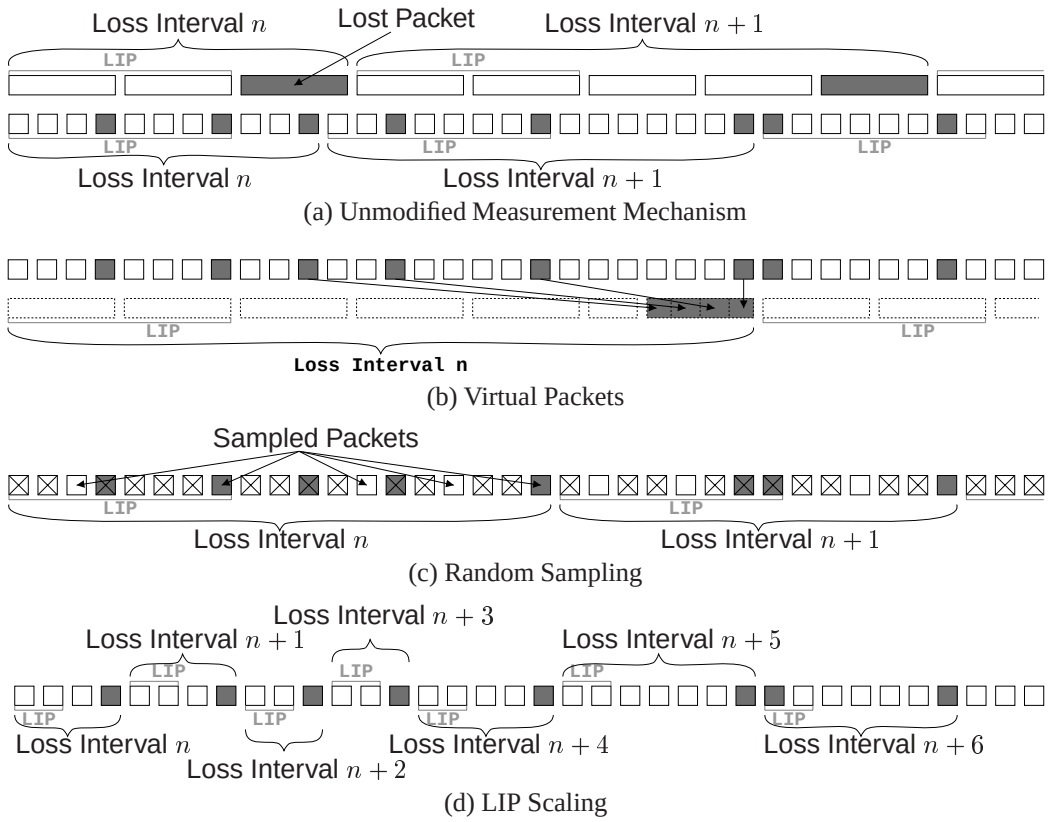


Figure 3: Schematic illustration of the modified loss measurement mechanisms.

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 (see Figure 3(b)). 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).

DEFINITION 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.*

DEFINITION 2. *A* loss interval *is measured as the number of virtual packets between two successive loss events, including the lost packet that ends the loss interval. The size of a loss interval need not be an integer.*

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 using the virtual-packets method. θ_n^{VP} can be expressed as $\theta_n^{VP} = N - 1 + \frac{1}{\eta}\sum_{i=1}^{\eta} X_i$, where the X_i are iid geometric random variables with parameter p. It follows that $E(\theta_n^{VP}) = E(\theta_n)$, thus the bias is removed in average. However, the variance is $\mathrm{var}(\theta_n) = \frac{1-p}{p^2}$ for the original mechanism and $\mathrm{var}(\theta_n^{VP}) = \frac{1}{\eta}\frac{1-p}{p^2}$ for virtual packets. Thus the virtual-packet algorithm smoothes out the measure of θ_n (since $\eta \geq 1$). As shown in [16], a high variability of the loss event estimator results 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 an unmodified TFRC.

3.2.2 Random Sampling

Instead of aggregating small packets into large ones, the loss interval can also 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 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 (see Figure 3(c)). The same notion of LIP, loss event and loss interval as the one for the original loss measurement process is used.

Let θ_n^{RS} be the nth loss interval measured using the random sampling method. Each packet 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 (Y) and the number of sampled packets between the end of the LIP and the next sampled packet that is lost (X). It follows from our assumptions that X is geometric with parameter p and Y is binomial with parameters $((N-1)\eta, p_\eta)$. We derive that $E(\theta_n^{RS}) = E(\theta_n)$ and the bias is removed in average. Further, X and Y are independent and thus $\mathrm{var}(\theta_n^{RS}) = (N-1)(1-\frac{1}{\eta}) + \frac{1-p}{p^2}$.

The first term of the variance 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 amounts 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. This way, random sampling can be used to achieve the same responsiveness to congestion as the unmodified TFRC, whereas the virtual-packet method results in a smoother response.

3.2.3 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 $\frac{s}{S}$, a flow sending N packets of size S per RTT and a flow sending $N\eta$ packets of size s per RTT both send $N - 1$ packets per LIP. 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 3(d).

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 need to increase the size of the loss history (i.e., the number of loss intervals used for the calculation of the loss event rate) 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.

3.3 Loss Measurement Modifications (Byte Mode)

With RED in byte mode, the probability of dropping a packet of size S is η times greater than p_s, the probability of dropping a packet of size s. Therefore, the average number of packets between two packet losses is η times smaller for the flow sending large packets while the number of bytes between two packet losses is roughly the same for all the flows. Again, we present three different methods. However, in contrast to packet-mode methods, they are not equally well suited to remove of the bias in the loss measurement process.

3.3.1 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 the number of virtual packets within the LIP, $N - 1$, and the number of virtual packets between the end of the LIP and the next packet loss (including the lost packet), X, which is geometric with parameter p_s. The random variable θ_n^{VPb} is now $\theta_n^{VPb} = N - 1 + \frac{1}{\eta}X$, the expected loss interval size is $E(\theta_n^{VPb}) = N - 1 + \frac{1}{p_S}$, and is thus also free of bias.

3.3.2 Random Sampling

Remember that in byte mode, the number of bytes between loss events is roughly the same for a flow sending small packets and a flow sending large packets. The flow with small packets will therefore see loss intervals that are too large by a factor of η. Correcting the loss interval size amounts to a random sampling of arrived data packets. Upon packet arrival, a random experiment is performed and with probability $1/\eta$ the packet is included in the loss history. In contrast, lost data packet always have to be taken into account.

While the virtual-packets and the random-sampling mechanisms for packet mode are each 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 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.

3.3.3 LIP Scaling

Adjusting LIP scaling to byte mode results is an even less favorable mechanism. In packet mode, the expected number of lost packets within the LIP increases with a reduction in packet size. Consequently, reducing the LIP causes some lost packets that would have otherwise been ignored to be counted as loss events.

In contrast, in byte mode the expected number of lost packets within the LIP does not increase with a decrease in packet size. Reducing the LIP is therefore not sufficient to compensate for the larger loss intervals seen by a flow sending small packets. In theory, it is possible to introduce artificial loss events to split the large intervals into η segments. While the expected loss event rate would be the same as for a flow with large packets, introducing artificial packet loss does not capture the dynamics of the underlying loss process. 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 [17] 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.

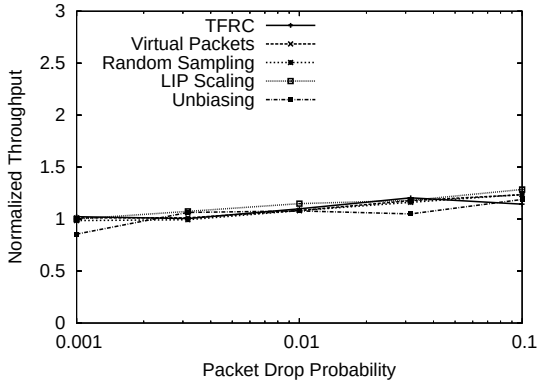


Figure 4: Bernoulli loss (VP-TFRC packet size 100 bytes)

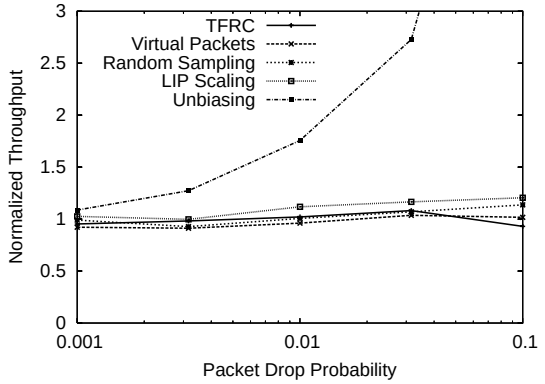


Figure 5: Dynamic Bernoulli loss (VP-TFRC packet size 100 bytes)

For the simulation topology, we use the well-known dumbbell topology, with senders and receivers on either side of a single bottleneck link. 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.

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. For all simulations, the reference TCP packet size is 1000 bytes.

4.1 Artificial Channel

Before investigating more complex scenarios where flows 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. For reasons of brevity, we only show graphs for the first setting and discuss whenever results for the second setting differ.

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 same average drop probability is maintained. The congestion control protocols frequently have to adjust their sending rate, putting more emphasis on the transient behavior of the protocols. Lastly, a time-based variant of the two state Gilbert loss model is used, where packet losses are no longer independent but highly correlated. It is not intended to closely model realistic network conditions but is merely used to analyze how the mechanisms perform when the assumption of packet loss independence is not met.

In these simulations, a loss module is inserted at the bottleneck router so that the total bandwidth consumed by all flows is well below the link capacity. All packet drops are caused by the loss module. 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.

4.1.1 Bernoulli Loss Model

We insert a loss module at the bottleneck router which drops packets with a fixed probability. 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 do not influence each other.

The throughput of the mechanisms shown in Figure 4 is normalized to TCP throughput. Their performance is extremely good with deviations from TCP throughput of 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. We obtain almost the same results when the packet rate is fixed at 160 packets/second and the packet size varies, although here the virtual-packets method has the highest deviation from TFRC throughput.

4.1.2 Dynamic Bernoulli Loss Model

The dynamic Bernoulli packet dropper alternates the drop rate between 0.5 times and 1.5 times the average. The 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 TCP 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 as shown in Figure 5. All mechanisms tend to become more aggressive than TCP when the packet rate is fixed and the drop probability is high.

4.1.3 Gilbert Loss Model

We use a single time-based Gilbert loss model at the bottleneck with a time constant of 10 ms. The drop pattern differs from that of a setup with a Gilbert loss model per flow and depends on the number of flows sharing the bottleneck and the packet rate of the flows. Nevertheless it introduces significant correlation in the loss process. As far as throughput is concerned, all of the methods are far from fair, as can be seen from the normalized throughput graph, Figure 6. Again, 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 than TCP for loss rates but becomes as aggressive as plain TFRC for high loss rates. Virtual packets and random sampling perform alike, with a throughput of less than 50% of that of TFRC. With TFRC itself being too aggressive, they achieve a throughput much closer to TCP throughput than plain TFRC.

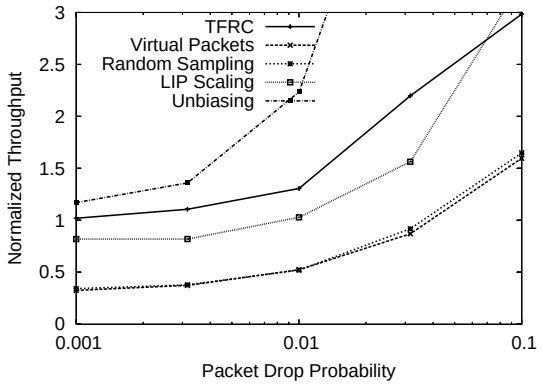


Figure 6: Gilbert loss model (VP-TFRC packet size 100 bytes)

We note that this improvement in fairness is caused by two effects that counterbalance each other, not because the mechanisms themselves better model TCP performance (equation-based congestion control in general being too aggressive because model assumptions are violated and the modified loss measurement mechanisms resulting in an overestimation of the loss event rate). Nevertheless, with these mechanisms we achieve roughly the same performance under normal circumstances and they are more conservative than TFRC under unfavorable network conditions where TFRC is too aggressive. This is the behavior we would like to see in a modified TFRC protocol and thus at this point we eliminate unbiasing and LIP scaling and keep Virtual Packets and Random Sampling.

Performance is slightly better when a fixed packet rate is used instead of a fixed packet size. All of the mechanisms achieve a throughput very close to TCP's for low packet loss rates, but results more and more resemble the unfair behavior shown in Figure 6 for higher loss rates.

4.1.4 Throughput Stability

From the analysis in Section 3.2 we know that the proposed loss measurement mechanisms have a different variance. Figure 7 shows the coefficient of variation (CoV) of the measured loss interval size from the previous experiments based on the Bernoulli loss model.

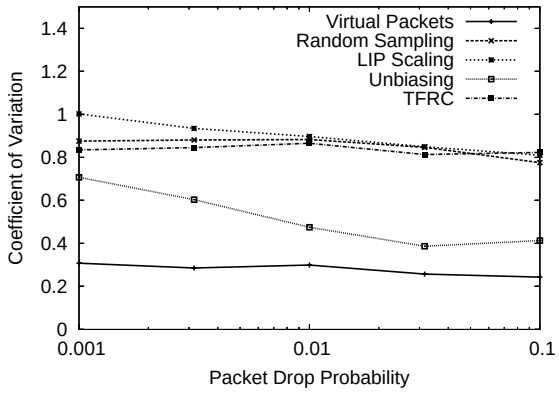


Figure 7: Coefficient of variation of the loss interval size

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 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, as shown in Figure 8.[3] 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. Analyzing the dynamic Bernoulli loss model shows a CoV that is generally higher, but the results themselves are comparable. In contrast, with the Gilbert loss model the CoV of LIP scaling is significantly higher than that of the other methods.

As a consequence, 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.

4.2 Bandwidth Limited Bottleneck

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. Usually, only flows with a low

[3] All mechanisms have the same average throughput; the lines are shifted vertically for better readability and hence the y-axis is omitted.

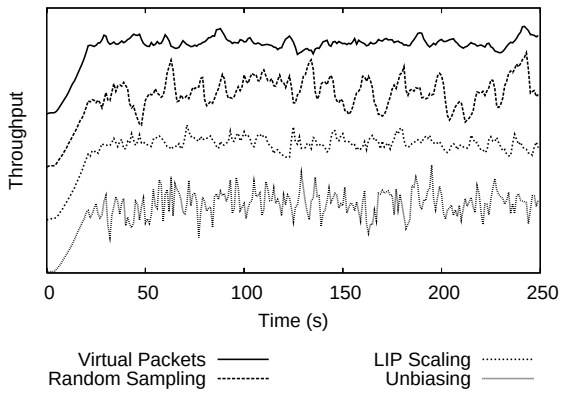


Figure 8: Throughput over time

bandwidth requirement are forced to use a packet size smaller than the MTU and therefore, we mainly investigate settings with a low per-flow bandwidth.

We use four different settings for the application requirements, the first with a fixed packet size, two settings with different but fixed packet rates, and lastly a setting where the packet size and the packet rate vary.

1. The packet size is fixed at 100 bytes and the fair bandwidth is 96 KBit/s per flow.

2. The packet rate is fixed at 50 packets/second, 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.

3. The packet rate is fixed at 160 packets/second, 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.[4]

4. The packet size is uniformly distributed between 50 bytes and 350 bytes and the fair bandwidth is 96 KBit/s per flow (i.e., packet rate and packet size are variable).

The simulation results are 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).

Unless stated otherwise, the same number of VP-TFRC flows and TCP flows is used for all the simulations (i.e., 1vs1, 2vs2, etc.). The bottleneck capacity is scaled to the number of flows. All of the simulations were conducted once with RED and once with a drop-tail queue. The bottleneck buffer is set to the equivalent of twice the bandwidth delay product (see Appendix A for more details). The experiments were also conducted with half the queue size with the expected result of a decrease in the level of fairness in favor of TFRC and VP-

[4]The main reason for using a packet size smaller than the MTU is to allow delay sensitive applications to send before accumulating a full MTU worth of data. With a typical maximum inter-packet delay for audio packets of ∼20ms, we expect flows to have packet rate requirements on the order of 50 packets/second and flows with 160 packets/second are clearly border line cases.

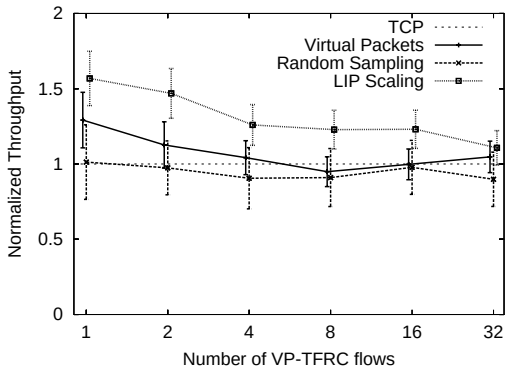


Figure 11: VP-TFRC packet rate 160 packets/second (drop-tail queue in packets)

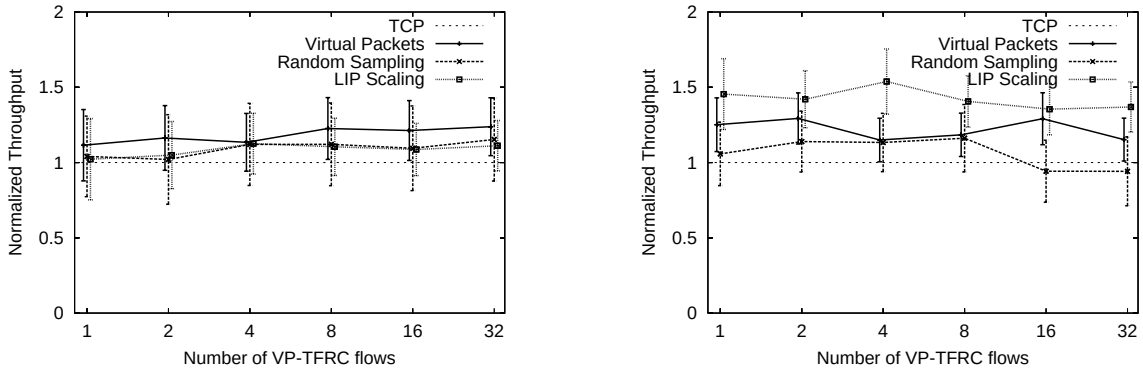
TFRC. More details on the simulation setup can be found in the appendix. For reasons of brevity we only present a small selection of the simulation results. Further simulations can be found in the technical report [15].

The access links to the routers are provisioned with 100 MBit/s. Unless stated otherwise, 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). For all simulations, the reference TCP packet size is 1000 bytes.

4.2.1 Packet Mode

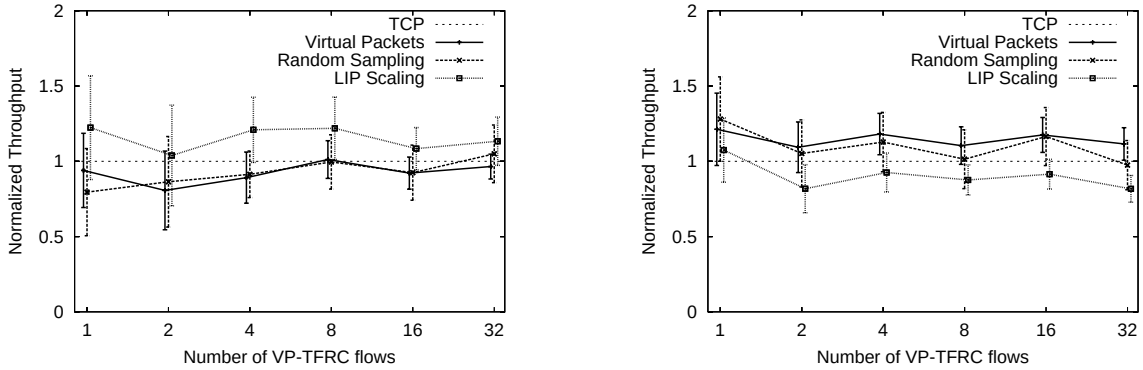
In packet mode, the decision to drop a packet at the bottleneck is based only on the packet rate and not on the packet size. In Figures 9 and 10, we show the throughput of the different VP-TFRC variants, normalized to TCP throughput, together with the standard deviation of the original time series. As is to be expected, the fairness of the VP-TFRC variants improves when RED queuing rather than drop-tail queuing is used. In the simulations, LIP scaling is the most aggressive of the different variants. Particularly when packet loss is correlated when using the drop-tail queue, LIP scaling is significantly too aggressive (as evidenced in the 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). When comparing Figures 9(b) and 10(a), we observe that using a fixed packet rate results in slightly higher VP-TFRC throughput but overall, whether the packet size or packet rate is fixed has little impact on the fairness of the algorithms. The same holds for the corresponding experiments with a RED queue. For reasons of brevity, we only give the results of the drop-tail simulations with 160 packets/second in Figure 11. These higher packet rate simulations generally achieve slightly better fairness than the 50 packets/second simulations.

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



(a) RED queue in packets (b) Drop-tail queue in packets

Figure 9: VP-TFRC packet rate 50 packets/second (buffer size 2x bandwidth delay product)



(a) Buffer size 2x bandwidth delay product (b) Buffer size 1x bandwidth delay product

Figure 10: VP-TFRC packet size 100 bytes (drop-tail queue in packets)

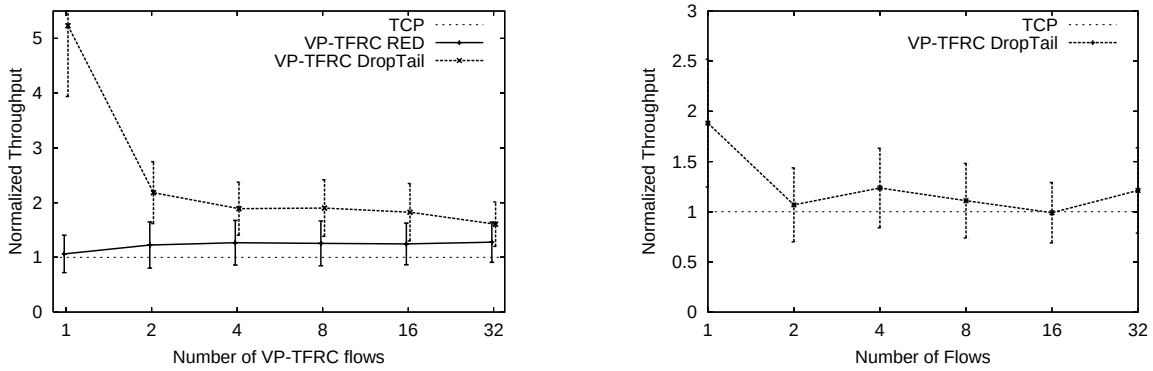
simulation results for a bottleneck with a drop-tail queue and two bandwidth-delay products worth of buffering (Figure 10(a)) to simulations with half the amount of buffering (Figure 10(b)), random sampling and virtual packets behave like plain TFRC and become more aggressive as buffer space decreases. In contrast, LIP scaling is less aggressive (i.e., relatively fair to TCP) with a buffer size of one bandwidth-delay product but consistently too aggressive for larger buffer sizes. The same effect can be seen with RED queues (not shown here), 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 is present whenever the loss process is time-driven instead of packet-driven.

In the simulations presented so far, either the packet size or the packet rate was fixed. To verify that the mechanisms also work when both vary simultaneously, we perform the same simulations as discussed before with a packet size that is uniformly distributed between 50 bytes and 350 bytes. VP-TFRC should then adjust the packet rate to obtain a fair share of resources. Comparing Figure 12 to the throughput obtained with a fixed packet rate shown in Figure 9(a), we can see that the more variable characteristics of the VP-TFRC flows with a random packet size do not lead to a decrease in fairness. The same can be observed for simulations with a drop-tail queue which give results similar to the ones shown in Figures 9(b) and 10(a).



(a) Normalized Throughput (b) Modified byte mode for $p_S = 2\eta\, p_s$

Figure 13: Fairness with byte mode (VP-TFRC packet size 100 bytes)

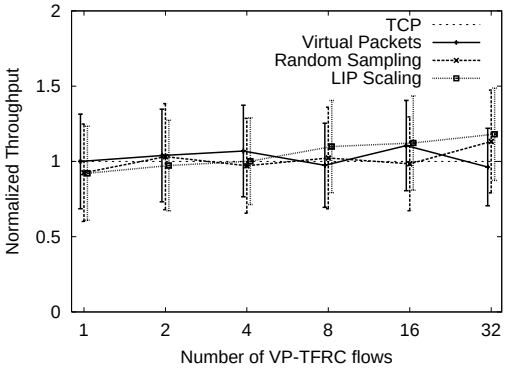


Figure 12: VP-TFRC packet size uniformly distributed between 50 bytes and 350 bytes (RED queue in packets)

4.2.2 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 13 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 in the previous experiments, we observe that fairness towards TCP is significantly higher for RED queues 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 fitting into the queue is a factor of η larger than the probability of a large packet to fit in, the primary assumption of proportional packet drop rates made in the design of the algorithm is met.

When analyzing the packet loss rates of TCP and VP-TFRC, we observe that particularly at low levels of statistical multiplexing small packets have a higher-than-proportional probability of fitting into the drop-tail queue. Instead of having a 10 times smaller packet drop rate than TCP, VP-TFRC sees a packet drop rate 25 to 100 times lower than TCP. As a consequence, VP-TFRC achieves a 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 50 packets/second, with 160 packets/second, and with 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. In these simulations, independent of the absolute value of the packet drop rate and the exact simulation setup, the ratio of packet drop rates was always close to $p_S = 2\eta\, p_s$, rather than $p_S = \eta\, p_s$ as for RED in byte mode. A variant of virtual packets in byte mode based on this ratio performs much better in a drop-tail environment, as shown in Figure 13(b).

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.

4.2.3 Further Simulations

Similar simulations were also carried out with a fixed number of 32 TCP flows and 1 to 32 VP-TFRC flows. Generally, those simulations 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. 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. 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 Figure 14 for a 5 ms delay at the access links and a delay at the bottleneck link of 10 ms, thus reducing the minimum RTT by a factor of 5.

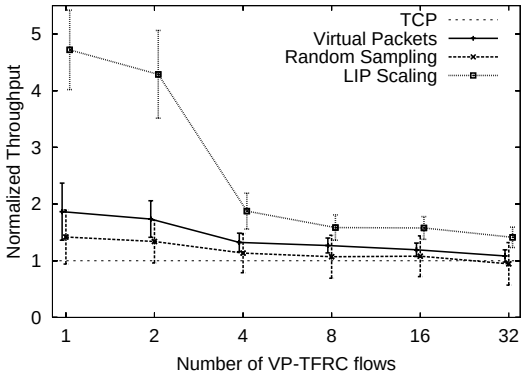


Figure 14: Fairness with a low RTT (drop-tail queue in packet mode)

The graph shows VP-TFRC flows with a fixed packet rate of 160 packets per second over a drop-tail queue. On these small timescales, LIP scaling becomes particularly aggressive. In contrast, if we also increase the available bandwidth per flow in conjunction with a decrease of the RTT, the simulations give very good fairness results.

We further performed simulations with a mix of different RTTs. Half of the TCP and VP-TFRC flows have the low RTT used above, while the remaining flows experience a much higher RTT. In Figure 15 we show the resulting throughput normalized to the throughput of the high-RTT TCP.

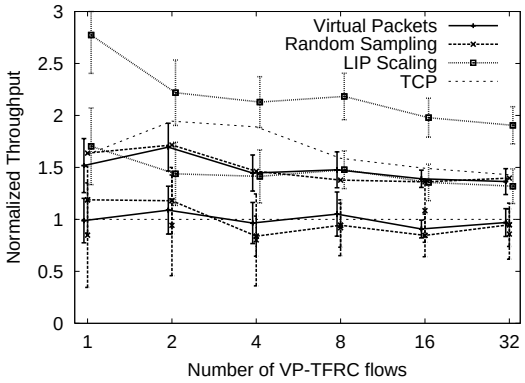


Figure 15: Fairness with mixed RTTs (drop-tail queue in packet mode)

We obtain similar fairness results as before for flows with the same RTT. In fact, a mix of RTTs improves fairness within the class of low RTT flows. The virtual packet mechanism and random sampling are now less aggressive than the TCP flow with the same RTT and also the aggressiveness of LIP scaling is reduced. As expected, the flows with a low RTT achieve a higher throughput than the flows with the high RTT. However, the difference is much less than the theoretical factor of 5 since a large contribution to the RTT comes from the queuing delay, which is the same for all flows. All in all, the proposed mechanisms seem to be rather insensitive to the specific RTT values.

4.3 Failure Cases

There are different possibilities where the method used for the loss measurement does not match the type of bottleneck: (1) the byte mode version of the mechanisms is used with a bandwidth limited bottleneck in packet mode or vice versa, or (2) the assumption that the bottleneck is bandwidth limited and not packet rate limited does not hold.

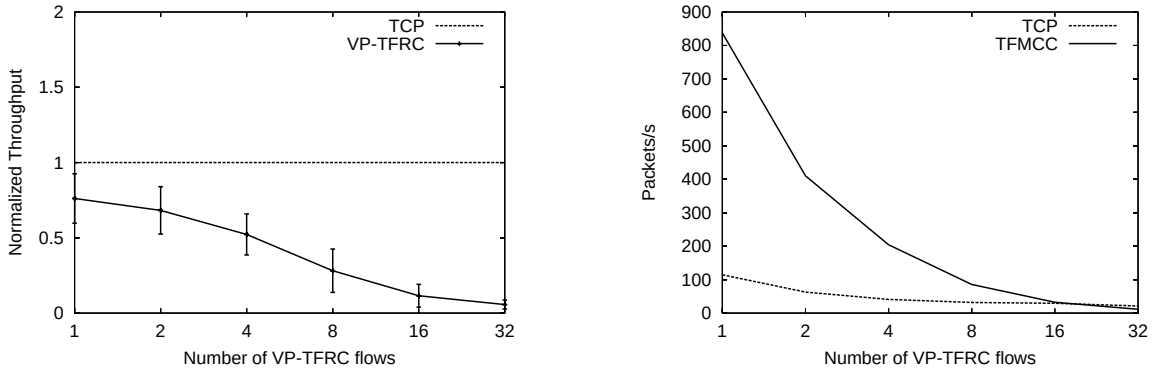
In the first case it is easy to predict the expected behavior. When a packet mode mechanism is used with a byte mode bottleneck, packet loss that should be treated as separate loss events will be aggregated, leading to a too low estimate of the loss event rate. On average (ignoring the effect of aggregation 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, VP-TFRC flows using a small packet size s will achieve roughly the same throughput as a TCP flow with large packets of size S. However, VP-TFRC's packet rate will be a factor of S/s higher than TCP's packet rate and 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 bottleneck.

For the simulations with a packet rate limited bottleneck, a rate limiter is inserted at the bottleneck router. The limit on the packet rate applies to all incoming links of the router (i.e., TCP acknowledgements and TFRC receiver reports add to the packet rate). The buffer size is set to 100 packets and all other parameters are set as in the simulations with a bandwidth limited bottleneck.



(a) Throughput (b) Packet rate

Figure 16: Packet rate limited bottleneck (VP-TFRC in byte mode with packet size 100 bytes)

When simulating a VP-TFRC flow in byte mode with a fixed packet size of 100 bytes and a TCP flow with the settings discussed above, the VP-TFRC packet rate greatly exceeds TCP's packet rate for low levels of statistical multiplexing. When only two flows compete, TCP's packet rate is around 100 packets/s while VP-TFRC's packet rate is 8 times as high, as shown in Figure 16. In these simulations, VP-TFRC flows always achieve less throughput than TCP flows but in any case, bandwidth is not the limited resource. Since the queue is measured in packets, the high packet rates of the VP-TFRC flows leave to little buffer space for the TCP flows to get to the proper sending rate. Interestingly, for higher levels of statistical multiplexing, this effect is mitigated and for 16 flows and more, the VP-TFRC packet rate is even less than TCP's packet rate. Since at a packet rate limited bottleneck the drop probability is independent of the packet size, the VP-TFRC flow in byte mode will see a roughly 10 times higher loss event rate. When this drop probability becomes significant because many flows share the link, the VP-TFRC flow may end up in the high loss rate regime of the throughput equation with an overproportional reduction in throughput. 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.

During transition periods, a flow may be limited by two different types of bottlenecks. This is the case if network conditions change and the sending rate of the flow is now too high for a bandwidth limited bottleneck and at the same time the packet rate is too high for a packet rate limited bottleneck on the same path. As soon as the congestion control mechanism has adapted the rate to the new conditions, the flow will only be limited by one of the two bottlenecks. However, the type and location of the limiting bottleneck may change over time. Since such cases seem to be of less importance, we leave them for future work.

5. CONCLUSIONS

We analyzed the impact of variations in packet size on equation-based congestion control (as used for example by TFRC) and presented four methods to cope with it. Without our 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. As can be seen from the variance of the loss interval estimator, random sampling has the same responsiveness as an unmodified TFRC, whereas virtual packets achieves a smoother sending rate which 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, the virtual packet method 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 that end, 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, etc.

Finally, we would like to briefly mention our experience with an alternative solution. Both RED in byte mode and the virtual-packet mechanism in byte mode partly compensate for the bias against flows with small packets. Instead of having part of the removal of the bias done by the RED mechanism and the other part done by the loss measurement mechanism, RED's byte mode could be altered so that (theoretically) a fair sharing of bandwidth among flows with different packet sizes can be achieved without any modifications to the loss measurement process. The fair sending rate depends on the RTT and the loss event rate, which in turn depends on the number of packets sent per RTT and the packet drop probability. We can numerically solve this equation offline to obtain the packet rate as a function of the packet drop probability. Using a function table, RED could then modify the packet drop probability so as to equalize the throughput of competing flows.

Great care has to be taken with the design of the averaging process at the RED gateway, since the instantaneous packet drop probability of RED used for the modification can be very different from the drop probabilities experienced at the end systems when there is a high level of statistical multiplexing. We found that the method works well when the network is a Bernoulli dropper, but we encountered compatibility problems with the drop uniformization procedure of RED that still remain as an open challenge.

Acknowledgements

We would like to thank Mark Handley and Sally Floyd for initiating the work on variable packet size congestion control and for many helpful discussions.

6. REFERENCES

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

[2] Sandeep Sikka and George Varghese, "Memory-efficient state lookups with fast updates," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, Aug. 2000.

[3] Pankaj Gupta, *Algorithms for Routing Lookups and Packet Classification*, Ph.D. thesis, Stanford University, Dec. 2000.

[4] 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.

[5] J-C. Bolot, S. Fosse Parisis, and D. Towsley, "Adaptive FEC-based error control for internet telephony," in *Proc. IEEE Infocom*, Mar. 1999.

[6] J-C. Bolot and A. Vega Garcia, "Control mechanisms for packet audio in the internet," in *Proc. IEEE Infocom*, Mar. 1996.

[7] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

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

[10] Colleen Shannon, David Moore, and k claffy, "Characteristics of fragmented IP traffic on internet links," in *Proc. First ACM SIGCOMM Workshop on Internet Measurement*, San Francisco, CA, Nov. 2001.

[11] Mark Handley, Jitendra Padhye, Sally Floyd, and Jörg Widmer, "TCP friendly rate control (TFRC): Protocol specification," RFC 3448, Jan. 2003.

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

[13] Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

[14] S. Ramesh and I. Rhee, "Issues in TCP model-based flow control," Tech. Rep. TR-99-15, Department of Computer Science, NCSU, 1999.

[15] J. Widmer, C. Boutremans, and J. Y. Le Boudec, "End-to-end congestion control for flows with variable packet size," Tech. Rep. EPFL-DI-ICA SSC/2002/82, EPFL, Switzerland, December 2002.

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

[17] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

APPENDIX

A. SETTING THE QUEUE SIZE IN THE SIMULATIONS

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, 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. We 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}). The $gentle\_$ option of RED is enabled.