Complementing TCP Congestion Control with Forward Error Correction

Vicky Sharma[1], Kadangode Ramakrishnan[2], Koushik Kar[3], and Shivkumar Kalyanaraman[4]

[1] Rensselaer Polytechnic Institute, Troy, NY 12180
sharmv@rpi.edu
[2] AT&T Labs Research, Florham Park, NJ 07932
kkrama@research.att.com
[3] Rensselaer Polytechnic Institute, Troy, NY 12180
koushik@ecse.rpi.edu
[4] IBM Research, Bangalore, India
shivkumar-k@in.ibm.com

Abstract. In this paper, we examine an emerging combination of challenges for TCP: increasingly bursty background traffic that is not subject to flow and congestion control, higher bandwidth networks and small buffers at network routers. As a result, TCP experiences short-term bursty packet losses that may not reflect long-term congestion. In this work, we propose a balanced approach that uses TCP congestion control mechanisms including Explicit Congestion Notification (ECN) to identify and overcome congestion, supported by adaptive FEC for short-term packet loss recovery due to bursty flows. We demonstrate the effectiveness of using FEC with TCP SACK congestion control by showing that such an approach improves performance for TCP flows for the emerging bursty traffic, small buffer, high bandwidth-delay product environments.

Keywords: Next-generation transport, Congestion control, Loss recovery.

1 Introduction

TCP has been remarkably successful as a transport protocol, and has evolved over the years to deliver the two core functions of reliability and congestion control. In this paper, we examine potential ways in which TCP may evolve to address a new combination of emerging challenges to TCP's performance.

The first challenge for TCP is to co exist with the exploding growth of multi media communications, including streaming and interactive video applications. Multimedia applications predominantly use RTP/UDP, and manifest themselves as bursty background traffic to regular TCP flows.

The second challenge is the increasing capacity of the end to end communication path coupled with smaller buffers at core routers (relative to the bandwidth delay product). The purpose of buffering is to absorb short term burstiness, so

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 378–391, 2009.

that any loss signal received by the end system is a reasonably unambiguous indication of medium to long term congestion, in terms of the round trip time and the time scales of end to end congestion control, and not burstiness. However, small buffers when subjected to background traffic that is highly bursty result in buffer overflows that are correspondingly bursty. Packet loss is transient and short lived, operating in time scales well beyond the dynamic range of end to end feedback based congestion control mechanisms. Importantly, such packet loss patterns significantly blur the distinction between medium term congestion and short term burstiness. TCP will increasingly confuse short term burstiness for medium term congestion, and cut its transmission window multiplicatively (following the traditional AIMD principle, and even with the newer versions that are more aggressive in their increase policies) each time, thereby paying a significant performance penalty. Worse yet, TCP may also timeout during larger bursts when a complete window of packets or a retransmission is lost. Such timeouts are a more severe response to the bursty cross traffic than is "truly necessary", and recovery from such "mistakes" takes several round trip times.

The right response to these short term bursty losses is to combat it with short term recovery mechanisms that don't necessarily operate in the larger multiple RTT time scales of feedback based congestion control schemes. Approaches that would help TCP recover from short term transient losses can greatly improve TCP's goodput and delay performance in these situations. Forward Error Correction (FEC) coding presents an alternative. If packets at the transport layer are FEC coded, the receiver would be able to recover the data even when some packets have been lost in transmission. This "insurance" like property of FEC coding results in a reduction of packet re transmissions and consequently, saves time and bandwidth.

The role of FEC coding in recovering from losses due to noisy/lossy links has been explored earlier [1],[2],[3]. In prior work, we developed enhancements to TCP like transport protocols MPLOT and LT TCP in [1] and [2] (for multiple paths and single path respectively) that employed FEC coding at the transport layer as a means to counter packet losses from noisy/lossy wireless links. Our protocols distinguished congestion losses from link losses by using the Explicit Congestion Notification (ECN) [4] as definitive indicator of congestion. Another useful feature of these protocols was the dynamic adaptation of FEC coding to match the losses incurred in the network. This helped in minimizing bandwidth wastage due to coding overhead. The end to end packet delay was also reduced due to the reduction in re transmissions needed to recover from losses. We showed in [1],[2] that such features are very effective in recovering from losses due to noisy links. As a result, the enhanced transport achieves a significantly higher goodput and lower delay than existing TCP protocols.

In this paper, we are interested in the question whether adaptive FEC mechanisms have a role beyond lossy wireless environments, and in the overcoming the challenge of short term transient losses in high bandwidth delay product networks with small buffers, combined with the presence of non congestion controlled traffic such as UDP. We show that adaptive FEC mechanisms can be

used to recover more efficiently from bursty losses in such situations. Beyond loss recovery, we are also interested in complementing congestion control by distinguishing true congestion from short term bursty losses. In this paper, we propose a balanced approach for TCP congestion control evolution: to use explicit congestion notification (ECN) to respond to congestion, supported by adaptive FEC for short term packet recovery. While fixed coding overhead may be undesirable as it reduces goodput, we show that adaptive coding feature of MPLOT and LT TCP, on the other hand, actually boosts goodput. We demonstrate that this approach results in a much smoother degradation in goodput with increasing packet losses as compared to conventional TCP SACK. This is one of our fundamental goals to achieve a much more gradual degradation in goodput as compared to the highly non linear (exponential) decay in goodput seen by TCP applications as the level of transient overload increases. It is worth noting that our scheme does not make TCP more aggressive, or send beyond the constraints placed by congestion control mechanisms such as self clocking, congestion window, AIMD, timeout window reduction etc. We use FEC primarily as a reliability technique to enhance TCP SACK, and it has collateral benefits of helping in "brittle" phases of congestion control caused by burstiness and small buffers. In essence, our paper makes the case that FEC as a reliability technique has broader applicability beyond lossy wireless environments, and can complement TCP SACK's congestion control mechanisms for emerging bursty, small buffer, high bandwidth delay product environments.

The paper is structured as follows: in the next section, we discuss the motivation for our work along with the related work, followed by a brief discussion on transport layer FEC coding and how TCP SACK can be modified to make effective use of FEC. We then discuss the results of our simulation based evaluation and conclude by summarizing our results.

2 Motivation and Related Work

Liu *et al.* [5] have observed that medium/high speed streams exhibited high burstiness due to packet accumulation. One stream was observed to peak at 600 Mb/s for a few micro-seconds when it transferred only 3.1 Mb/s in 3 seconds. Fitzek *et al.* [6] observed peak-to-mean values between 15 and 45 for bit-rates of audio/video streams. These studies indicated a high burstiness in traffic patterns for new streaming applications.

Appenseller *et al.* [7] showed that router buffer sizes can be much smaller than the end-to-end bandwidth-delay product. Their theoretical results showed that the router buffer size is bounded by $1/\sqrt{n}$ of the bandwidth-delay product when a link is being used by n identical TCP-like flows. Mascolo *et al.* [8] take this study a step further by proposing a TCP congestion control framework aimed at minimizing end-to-end bandwidth loss due to congestion. However, the results in [7],[8] were based on the assumption that only identical TCP-like flows use the network. Furthermore, in deriving bounds on the buffer sizes, these works only focus on maintaining a desired throughput value, and ignore the goodput metric which is more relevant in practice.

A bursty unresponsive flow (e.g. UDP) can quickly overflow small queue buffers leading to congestion losses for TCP flows. Consequently, TCP flows would reduce their packet rates, thereby reducing throughput. Due to packet losses, a TCP flow can take multiple Round Trip Times (RTTs) to recover lost packets by re-transmissions. Due to the nature (AIMD policy) of TCP congestion control, after a packet loss has occurred, it can take a long time for a TCP flow to build up it's throughput and goodput to levels before the congestion losses. Consequently, a TCP flow will potentially lose significant amounts of bandwidth and incur large delays due to the bursty losses caused by an unresponsive flow.

In presence of such bursty losses, therefore, it serves TCP well to recover from losses quickly to reclaim lost bandwidth. FEC has been used to recover from packet losses due to noisy links in [1],[2] and [3]. Hayasaka *et al.* [9] use FEC to recover lost video packets due to congestion. Their solution needs a long buffering time to transmit FEC packets before video transmission, however. Yu *et al.* [10] use simplistic M/M/1 queueing models to derive optimal block size and code-rate for FEC to recover from congestion losses, but only under very idealistic conditions and assumptions (like all flows use TCP and are identical) which may not hold in reality. Li *et al.* [11] use fixed coding overhead and flexible packet scheduling to counter packet losses from noisy links. The packet scheduling is varied to control the average loss-rate. A fixed coding rate approach may however reduce goodput due to unnecessary coding overhead, as we demonstrate later in this paper. Nguyen *et al.* use an exhaustive line-search to schedule a block of coded packets for minimizing the likelihood that the receiver would not be able to recover lost data. Here the authors only consider bandwidth constraints in their work, ignoring any increments in delays incurred in the process.

Ahlswede *et al.* [12] present network-coding as a possible solution to recover from losses in multi-cast applications. Application of network-coding has not been applied/studied for unicast TCP flows extensively, however. There have been a few attempts to study the effects of network-coding on TCP flows, but in the context of wireless networks. For example, Huang *et al.* [13] and Ghaderi *et al.* [14] study the impact of network-coding on unicast TCP flows running over wireless mesh networks, and conclude that significant gains cannot be attained unless the MAC layer is significantly changed.

We note that FEC has been primarily used to recover from non-congestion losses. The overhead required by FEC is an undesirable feature which can potentially reduce goodput (compared to the case with no coding) if FEC provisioning is not done appropriately. In the next section we show how FEC can be used intelligently to employ it's loss-tolerance properties while limiting the overhead incurred to negligible levels in presence of bursty non-responsive flows. The basic idea is to employ FEC only when losses are incurred, and in proportion to the amount of losses incurred. We then show through simulations that our proposal benefits TCP by allowing it to recover from losses and obtain more goodput than the conventional TCP-SACK.

3 Packet Recovery Using FEC Coding

3.1 FEC Coding at the Transport Layer

In this paper, we use the term FEC to refer to erasure codes which have excellent loss tolerance properties. In brief, a (n, k) block FEC code adds $(n-k)$ redundant units to k data units in such a fashion that the original k data units can be recovered from any of the k units. This implies that a few FEC coded packets can be lost during transmission, but the receiver would still be able to decode and recover the original data packets from the remaining packets received.

Since our goal is to complement TCP congestion control with FEC, we propose to code packets (or TCP segments) at the transport layer itself. As a result, data can be recovered at the receiver even when a few packets are lost due to short-term congestion. FEC would reduce the number of re-transmissions, thus increasing TCP goodput during lossy phases. Reduction in re-transmissions also reduces the delay incurred in recovering from packet losses, allowing the TCP flow to quickly build up to throughput and goodput levels prevailing before the losses.

The performance of an (n, k) FEC block code depends on two key parameters – (i) code-rate $\frac{n}{k}$, and (ii) block size n. The choice of these two parameters significantly influences the goodput and delay properties of the transport protocol.

The degree of loss-tolerance attained by FEC coding is directly proportional to the code-rate. A higher code-rate also indicates a higher coding overhead. Hence, there exists a trade-off between degree of loss-tolerance and coding overhead. A fixed code-rate is clearly not optimal because it would consume bandwidth even in zero-loss conditions, reducing goodput. A fixed code-rate is also useless if packet losses exceed the degree of loss-tolerance provided by the code.

Clearly, we require a code-rate that would adapt to variations in network conditions. An adaptable code-rate would only provide loss-tolerance during lossy conditions (more specifically, provision enough FEC based upon an estimate of loss statistics), thereby reducing bandwidth wastage. Therefore, the transport protocol must be able to estimate network losses as well as establish requirements for the degree of loss-tolerance required from the FEC code. We show in [1] that in order to decode any block without additional re-transmissions with high probability, a code-rate that is inversely proportional to the sum of average packet loss-rate and packet loss-rate deviation is appropriate. The likelihood of decoding bounds the expected goodput from below, providing a minimum performance guarantee.

The choice of block size determines the average delay experienced by the application. Since the receiver must at least wait for the amount of time it takes to transmit the complete block of n packets, a larger block size would lead to an increase in delay. As a block must consist of an integral number of packets, the desired code rate often imposes restrictions on the block size. For instance, a code-rate of 1.01 can be realized exactly only by a block of size 101 (as a $(101, 100)$ code, for example), or multiples of it.

Lundqvist *et al.* [3] and our prior work [1] argue for a maximum block size equal to the window size (which is a measure of the connection's bandwidth-delay product) to attain the optimal trade-off between goodput and delay.

3.2 TCP-SACK Modifications for Effective Use of FEC

We modify the standard TCP-SACK to employ FEC coding as mentioned in the previous section. We denote this version of TCP as *TCP-cd (coded TCP)*.

The focus of TCP-cd design is to emphasize that FEC, a mechanism for reliability and loss recovery, when integrated into TCP, also has collateral benefits by reducing the "brittleness" of TCP congestion control. The brittleness is observed particularly in small-window phases and during transient burst losses, in addition to other situations. The "insurance" provided by FEC helps provide additional resilience under these conditions, and specifically leads to reduced latency during the loss recovery phase and reduced fall back to timeouts that hurt performance.

TCP-cd calculates the packet loss rate from loss measurements using Exponentially Moving Weighted Average (EWMA), and then calculates the variance of packet losses accordingly. We use the method described in [1] for the MPLOT protocol to measure and update packet loss statistics. TCP-cd then scales the TCP congestion window by the inverse of the packet delivery rate, i.e., (1 - packet loss rate). This extra redundancy ensures that the "loss adjusted" congestion window remains at the level as originally targeted, on an average. TCP-cd encodes the data packets with a $(n, (1 - \mu - \sigma)n)$ block code where n is the scaled window size, μ is the average packet loss rate and σ^2 is the packet loss variance. TCP-cd dynamically adapts the code-rate and block size as the measured/estimated loss rate varies. Unlike some implementations of TCP, the window is not reduced simply based on the receipt of three duplicate acknowledgements. Instead, we use ECN to detect congestion and reduce the window. TCP-cd is a simpler version of MPLOT [1] in the sense that it only includes FEC coding from MPLOT while excluding any capabilities to take advantage of out-of-order transmission, packet size adjustment and other policies employed by MPLOT to take advantage of multiple paths.

In order to estimate the effect of FEC, we also simulate another variant of TCP-SACK – *TCP-la (loss aware TCP)*, that measures the packet loss statistics like TCP-cd but does not employ FEC coding. Thus in TCP-la, packet loss measurements are only used to scale up the congestion window as described above. We simulate TCP-SACK, TCP-la and TCP-cd under different conditions to measure the throughput, goodput, packet losses and timeouts. The simulations in the next section show that TCP-cd achieves better goodput than TCP-SACK and TCP-la even though all the three protocols have the same throughput.

4 Simulation Results

In this section, we present results from simulation experiments which show that intelligent use of FEC coding, as discussed in section 3.2, helps in substantially

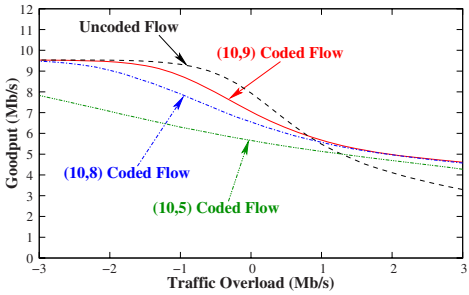
improving the performance of TCP by increasing end-to-end goodput. The effect of bursty packet losses due to short-term transient congestion are overcome without triggering the more severe congestion response of TCP. Through simulations, we aim to investigate the following issues: (i) whether and by how much does coding help in improving TCP throughput and goodput in the presence of bursty unresponsive flows, (ii) how buffer sizes at network core routers impact the effectiveness of the use of FEC, and (iii) the extent of burstiness (in terms of the the peak-to-mean ratio of the bursts) for which of FEC is effective in improving TCP goodput.

Our ns-2.30 simulations use the typical single bottleneck dumb-bell topology. We use varying numbers of different types of sources (responsive (TCP), non-responsive (UDP), with/without congestion control) for assessing the contribution of FEC coding in recovering from congestion losses. We use a bottleneck link bandwidth of 10 Mb/s and a Round Trip Time (RTT) of 40 ms in the simulations, unless specified otherwise. To understand the impact of our work for networks with high bandwidth-delay product environments, we examine our results with varying buffer sizes, as a function of the bandwidth-delay product, at the bottleneck.

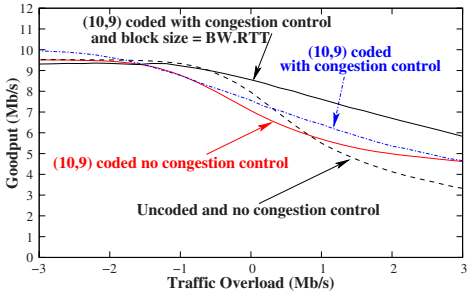
4.1 Complementarity of FEC and Congestion Control

We first investigate the effectiveness of FEC coding in attaining higher goodput in the presence of losses due to buffer overflows, both with and without congestion control. To first understand the role of coding in improving goodput when a transport protocol does not use flow and congestion control, we simulate an end-end reliable transport protocol similar to TCP except that the sources have no window flow control and therefore transmit at a constant packet rate (e.g., UDP with reliability built-in). We simulate 10 such sources over a single-bottleneck topology and vary the bottleneck bandwidth from 6 Mb/s to 15 Mb/s. Each source transmits application (data) packets at 1 Mb/s. If the packet flow is (n, k) coded, then the sending rate is increased to $\frac{n}{k}$ Mb/s to keep the rate of data packets the same. For the coded flows, a block size of 10 packets is used. The queue is a simple drop-tail queue with a buffer size equal to the bandwidth-delay product.

Figure 1(a) shows the cumulative goodput achieved by all the sources as the *traffic overload*, defined as the difference between the aggregate application traffic rate and the bottleneck bandwidth, is increased. The uncoded flow experiences losses only when the bottleneck is overloaded (i.e., the bottleneck bandwidth is 10 Mb/s or lower). However, for the coded flows, packet loss occurs earlier, once the sending rate exceeds $10\frac{k}{n}$ Mb/s for an (n, k) coded flow (i.e., the aggregate sending rate exceeds the bottleneck bandwidth). Furthermore, the uncoded flows exhibit higher goodput as compared to coded flows when the traffic overload is less than 1 Mb/s. This is intuitively expected, since the code-rates used for the coded flows in the three cases shown in the simulations are at least 10%. However, as the traffic overload increases, the coded flows experience a more gradual decay in goodput than uncoded flows, thereby achieving higher goodputs



(a) Effect of coding on goodput of flows with no congestion control.



(b) Effect of coding and congestion control on goodput.

Fig. 1. The use of FEC coding in congested networks can have a significant effect on goodput. The combination of coding and congestion control leads to more gains.

at higher levels of congestion. These results show that FEC coding can certainly help reduce packet loss rates at high congestion levels. In addition, the extent of coding overhead (low with (10,9); high with (10,5)) has a significant impact on the goodput.

In the next set of simulations (results shown in figure 1(b)), we introduce a rudimentary congestion control function (similar to the AIMD algorithm in TCP) along with FEC coding to gauge their combined effect, and compare the performance of coded flows with the uncoded case subject to similar congestion control. For coded flows, we only consider the $(10, 9)$ block code case. We use two block sizes: one a fixed block size of 10 as before, and another where the block size is set to the bandwidth-delay product. Note that the two curves corresponding to no congestion control are the same as in figure 1(a), and are included here for comparison. We observe that the goodput is substantially improved with the introduction of the simple congestion control mechanism. However, performance of the coded case improves more than the uncoded case. This is particularly true if the block size is set to the bandwidth delay product, which attains significantly higher goodput compared to the uncoded case. More importantly, the reduction in goodput for coded flows is linear (gradual) in nature as compared to uncoded flows which experience a more severe non-linear reduction in goodput. These

results imply that FEC coding can work well with congestion control mechanisms in improving goodput by reducing losses, particularly if the block size is set to the bandwidth-delay product. We explore this issue in more detail in the rest of our simulation experiments.

4.2 Benefits of FEC with TCP Flows under Bursty Losses

In order to represent the conditions of unresponsive and bursty flows that expose TCP flows to congestion losses, we simulate a periodic UDP flow that transmits at a peak rate for a certain time-period and then turns off for an equal time period (50% duty cycle). We use an ON-period of 50 RTT (2000 ms) and vary the peak rates from 3 Mb/s (low congestion) to 15 Mb/s (high congestion) in our simulations.

We first compare the throughput values (aggregate rate of packets sent, including retransmissions and in the coded case, redundancy packets) obtained by the three TCP variants for different UDP rates and router buffer sizes, and establish the fact that our use of coding stays within the window flow control limits of TCP. Figure 2 shows the average throughput values (as a percentage of

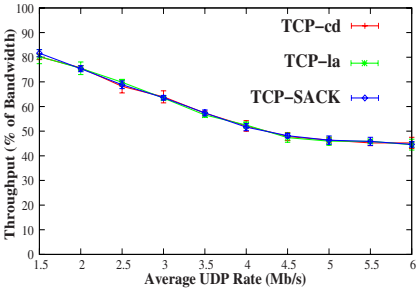


Fig. 2. Throughput when router buffer size equals bandwidth-delay product

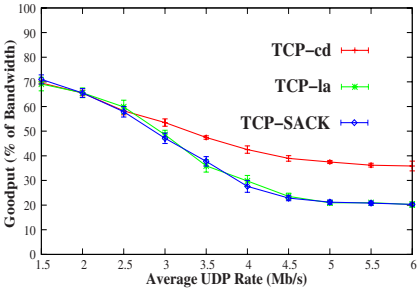


Fig. 3. Goodput when router buffer size equals bandwidth-delay product

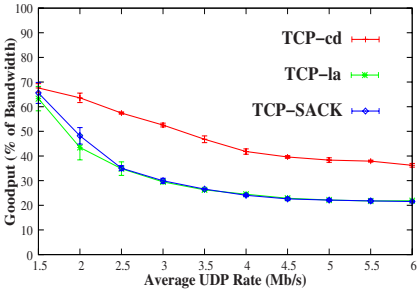


Fig. 4. Goodput when router buffer size equals *half* of the bandwidth-delay product

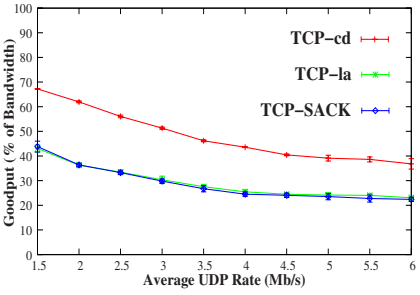


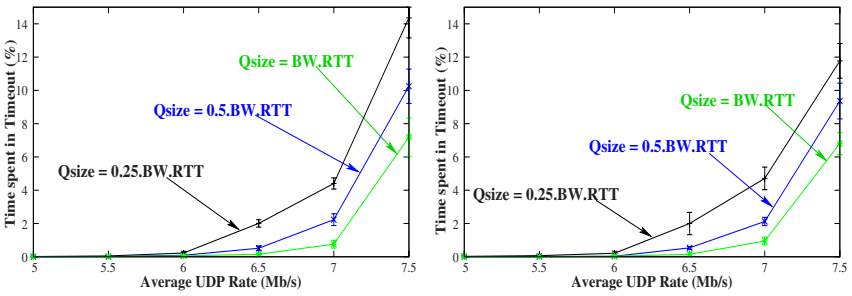
Fig. 5. Goodput when router buffer size equals *quarter* of the bandwidth-delay product

the bottleneck bandwidth) along with their 95% confidence intervals for TCP-cd, TCP-la and TCP-SACK, when the buffer size at the routers is equal to the bandwidth-delay product. We note that the throughput values of the three algorithms are roughly the same. (This trend is also observed for buffer sizes of $BW.RTT/2$ and $BW.RTT/4$ – not shown due to space limitations.) The similar throughput for the three algorithms is because they use the same TCP congestion control mechanism (AIMD algorithm with similar parameters), and therefore increase and decrease window sizes similarly in response to congestion losses. However, for the same throughput, we show below that TCP-cd is able to attain much higher goodput by being able to convert a larger fraction of the throughput to goodput, as compared to TCP-SACK and TCP-la.

By using dynamic coding, we limit the coding overhead appropriately (as compared to using a fixed coding rate), and adapt the coding rate nicely to match the time-varying loss rate on the path of a flow. We now measure how dynamic adaptive coding affects goodput in presence of TCP congestion control. The average goodputs (as a percentage of the capacity of the bottleneck) with their 95% confidence intervals are shown in figures 3-5, for the three variants and buffer sizes of $BW.RTT$, $BW.RTT/2$ and $BW.RTT/4$, respectively. Note that the bottleneck capacity is 10 Mb/s; thus even at an average UDP rate of 6 Mb/s, only 60% of the bottleneck bandwidth is used by the UDP flow.

For TCP-SACK and TCP-la, the goodput reduces by almost 75% as the average UDP rate increases from 1.5 Mb/s to 6 Mb/s. This shows the severe impact that bursty flows can cause to goodput, even if they do not lead to long-term congestion. In contrast, the goodput for TCP-cd decays by about 60% across the same range of average UDP rates. This shows that use of FEC helps in rapid recovery from losses, thereby reducing re-transmissions and increasing goodput. This behavior is consistent across all values of buffer sizes simulated. Comparing the throughput and goodput values, we note that TCP-SACK is only able to convert 30% of its throughput to goodput (at an average UDP rate of 6 Mb/s) while TCP-cd can convert 66.7% of throughput to goodput at the same average UDP rate. This is more than a 100% improvement in converting throughput to goodput. Since the goodput performance of TCP-la and TCP-SACK is very similar, this implies that the performance improvement observed for TCP-cd is exclusively due to the effective use of coding, and not due to the loss-rate based scaling of the congestion window. Comparing the three figures (the three different bandwidth-delay product cases) we observe that the performance difference between TCP-cd and the other two TCP versions is more pronounced at lower values of the average UDP rates as the buffer size becomes smaller. This is particularly important as we go to higher bandwidth-delay product environments, where correspondingly the amount of buffering available at routers may be smaller.

To understand the cause for the improved goodput with TCP-cd, we examine the packet loss and timeout behavior of the different alternatives. The packet loss rate increases with an increase in average UDP rate as expected. In addition to the reduction in window size as a result of TCP's loss-based congestion response, there is also a likelihood of TCP flows suffering timeouts. We look at the average



(a) TCP-SACK (b) TCP-cd

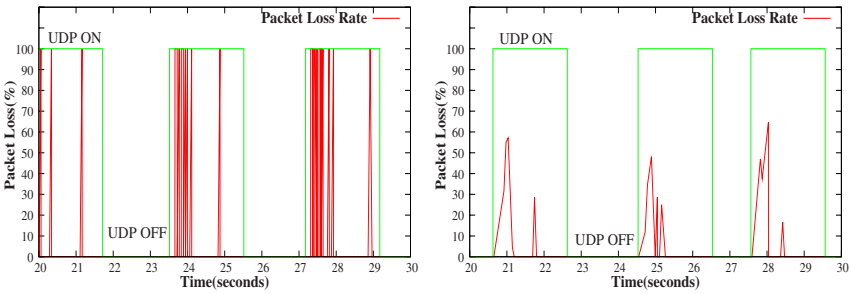
Fig. 6. The time spent in timeout by TCP-SACK and TCP-cd

time spent in timeouts by TCP-SACK and TCP-cd respectively in figures 6(a) and 6(b), for the three different buffer sizes. The timeouts become significant when the average UDP rate exceeds about 60% of the bottleneck capacity (which corresponds to the peak UDP rate exceeding 100% of the bottleneck capacity). However, while there is no dramatic difference between TCP-SACK and TCP-cd in terms of time spent in timeouts, TCP-cd does in fact see improvements in the amount of timeouts experienced with smaller router buffers.

Figure 7(a) overlays the variation in UDP traffic and the corresponding average packet loss rate measured at a TCP receiver when the UDP flow transmits at a peak rate of 10 Mb/s on a path with a bandwidth 10 Mb/s. TCP experiences losses when the UDP flow turns on, resulting in TCPs reducing their window, thereby reducing throughput and goodput as well. Packet retransmission and the potential of retransmitted packets being lost result in further reduction in goodput. In figure 7(a), which shows the loss-rates for TCP-SACK, the average packet loss is high, frequently experiencing 100% packet loss (resulting in timeouts). We now look at the average loss rate measured by a TCP-cd receiver in figure 7(b). While the packet loss rate is still high, it is considerably lower than the loss-rate observed for TCP-SACK. FEC coding thus enables a TCP flow to tolerate the co-existence of non-cooperating flows and overcome the effect of transient packet losses. Even under heavy congestion useful information is delivered through the use of coding.

We now study the impact of increasing the burstiness of UDP flows on TCP-SACK and TCP-cd. For this purpose, we maintain the peak rate of the UDP flow equal to the bottleneck bandwidth of 10 Mb/s and vary the ON-period for the UDP flow. The OFF-period of the UDP flow is fixed (at 2 secs), and we observe the throughput and goodput achieved by TCP-SACK and TCP-cd, while varying the ON-period from 0.4 secs to 20 secs, resulting in a peak-to-mean rate ratio of the UDP flow varying from 1.1 to 6. We present the throughput and goodput values for the two protocols in figures 8 and 9 respectively, where the router buffer size equals half the bandwidth-delay product.

We note from figure 8 that the throughput achieved by TCP-cd is the same as TCP-SACK, across the range of UDP burstiness values considered. In contrast,



(a) Packet loss seen by a TCP-SACK flow. (b) Packet loss seen by a TCP-cd flow.

Fig. 7. Comparison of packet losses for TCP-cd and TCP-SACK when UDP is transmitting at peak rate of 10 Mb/s and the buffer size is quarter of the bandwidth-delay product. The losses seen by TCP-cd are significantly less than those of TCP-SACK.

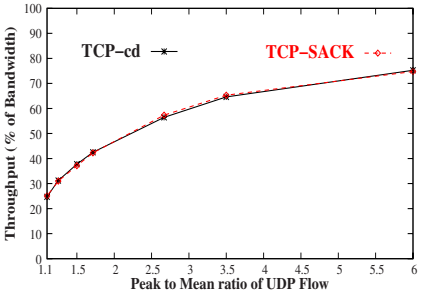


Fig. 8. Throughput comparion for different levels of burstiness of UDP flow (buffer=half BW-Delay product)

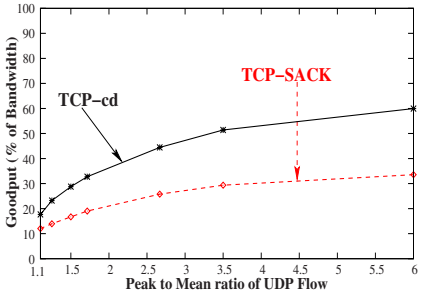


Fig. 9. Goodput comparison for different levels of burstiness of UDP flow (buffer=half BW-Delay product)

the goodput achieved by TCP-cd gets significantly better as the burstiness of UDP flow increases, as we observe in figure 9. For a peak-to-mean ratio of 6, the goodput achieved by TCP-cd is twice that of TCP-SACK. As the burstiness reduces, the UDP flow acts more like a constant rate flow, reducing the gains made by FEC.

In summary, we conclude that the use of adaptive FEC along with TCP congestion control can lead to significant goodput gains in scenarios that are typically observed in current networks, where there is likely to exist significant amounts of cross-traffic from non-cooperative UDP flows. The goodput improvement is more pronounced as the burstiness of the unresponsive flows, and hence the congestion losses, increases. This becomes particularly important in higher bandwidth-delay product environments, especially with routers having small buffers. Use of adaptive FEC coding helps in significantly extending the dynamic range of operation of feedback based congestion control mechanisms.

5 Summary and Conclusions

As networks evolve to higher bandwidth-delay product environments and have to carry traffic that is unresponsive to congestion control, the capabilities of congestion control will be pushed beyond their designed limits. In this paper, we demonstrated that existing TCP congestion control mechanisms can be effectively complemented by packet level FEC coding to quickly recover from short-term congestion losses caused by bursty unresponsive flows. The approach we propose is particularly useful in to help TCP perform reasonably well even when traffic is bursty causing transient overloads, buffer sizes are small and feedback delays are large, thus extending the dynamic range of traditional feedback-based congestion control protocols. The extent of goodput benefit provided by FEC coding increases as the router buffer size reduces. This is a particularly important as it allows routers to retain relatively small buffers for the emerging high bandwidth-delay environments, thereby reducing queueing delay while FEC coding protects goodput against short-term queue overflows. We also showed that the performance gains delivered by packet level FEC increases as the burstiness in the traffic and the associated packet losses due to congestion increase. The significant gains in goodput are achieved without a corresponding increase in throughput, as the overhead of FEC coding is minimized by adapting the code-rate to network losses. Our scheme is able to use the loss-tolerance property of FEC to attain higher goodput, while right-sizing the overhead associated with FEC.

In summary, intelligent/adaptive FEC coding can complement TCP's congestion control to effectively counter losses caused by the combination of bursty unresponsive flows and router buffer sizes that are small compared to the bandwidth delay product in high-speed networks.

References

1. Sharma, V., Kalyanaraman, S., Kar, K., Ramakrishnan, K., Subramanian, V.: MPLOT: A Transport Protocol Exploiting MultiPath Diversity using Erasure Codes. In: Proc. IEEE INFOCOM (2008)
2. Subramanian, V., Kalyanaraman, S., Ramakrishnan, K.: An end-to-end transport protocol for extreme wireless environments. In: Proc. IEEE Military Communications Conference (MILCOM 2006), Washington D.C, USA (October 2006)
3. Baldantoni, L., Lundqvist, H., Karlsson, G.: Adaptive end-to-end FEC for improving TCP performance over wireless links. In: IEEE International Conference on Communications, vol. 7, pp. 4023–4027 (2004)
4. Ramakrishnan, K., Floyd, S., Black, D.: RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP (2001)
5. Liu, Z., Hu, C., Zheng, K., Chen, S., Liu, B.: A trace study for characteristics of packet arrivals. In: IEEE 7th Malaysia International Conference on Communication & 13th IEEE International Conference on Networks, MICC-ICON, vol. 1 (November 2005)

6. Fitzek, F., Zorzi, M., Seeling, P., Reisslein, M.: Video and audio trace files of pre-encoded video content for network performance measurements. In: First IEEE Consumer Communications and Networking Conference (CCNC), pp. 245–250 (January 2004)
7. Appenzeller, G., Keslassy, I., McKeown, N.: Sizing router buffers. SIGCOMM Computer Communication Review 34(4), 281–292 (2004)
8. Mascolo, S., Vacirca, F.: Congestion control and sizing router buffers in the internet. In: IEEE Conference on Decision and Control & European Control Conference (CDC-ECC), pp. 6750–6755 (December 2005)
9. Hayasaka, M., Gamage, M., Miki, T.: An efficient loss recovery scheme for on-demand video streaming over the internet. In: The 7th International Conference on Advanced Communication Technology (ICACT), vol. 2, pp. 1301–1306 (2005)
10. Yu, X., Modestino, J., Bajic, I.: Performance analysis of the efficacy of packet-level fec in improving video transport over networks. In: IEEE International Conference on Image Processing (ICIP), vol. 2, pp. II–177–180 (September 2005)
11. Li, Y., Zhang, Y., Qiu, L., Lam, S.: Smarttunnel: Achieving reliability in the internet. In: Proc. of IEEE INFOCOM (2007)
12. Ahlswede, R., Cai, N., Li, S.Y.R., Yeung, R.W.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
13. Huang, Y., Ghaderi, M., Towsley, D., Gong, W.: TCP performance in coded wireless mesh networks. In: IEEE SECON (2008)
14. Ghaderi, M., Towsley, D., Kurose, J.: Reliability gain of network coding in lossy wireless networks. In: IEEE INFOCOM (2008)