AN EXPERIMENTAL STUDY ON THE PERFORMANCE AND FAIRNESS OF LOSS DIFFERENTIATION FOR TCP

Johan Garcia and Anna Brunstrom
Karlstad University
Universitetsg 2, SE-651 88 Karlstad, Sweden

Keywords: Loss differentiation, TCP, fairness, multiple connections, performance evaluation

Abstract: This paper investigates the performance and fairness of receiver-based loss differentiation for TCP. Experiments have been performed with a FreeBSD kernel implementation. As expected, the results verify the effectiveness of receiver-based loss differentiation when corruption losses are present. However, if the the corrupting link is shared with users that do not employ loss differentiation, the performance gain typically comes at the expense of increased unfairness. The results show that a single loss differentiating user may in some cases reduce the bandwidth of users without loss differentiation with up to 35 percent, but there are also cases where loss differentiation has limited fairness implications. The results further show that if a user employs multiple TCP connections over the corrupting link the negative effects of corruption losses are reduced. This is true even if all connections employ regular TCP. Hence, multiple connections at the application level can to some extent be used as a simple mechanism to limit the impact of corruption losses.

1 INTRODUCTION

Packet switched networks by design provide resources in a dynamic way. In the absence of admission control, packet switched networks need to ensure that the traffic does not saturate the network completely and create a congestion collapse. To control congestion in the Internet, the Transmission Control Protocol (TCP) (Postel, 1981) employs congestion control. TCP congestion control interprets packet loss as congestion and applies some congestion avoidance behavior, typically by the TCP sender halving the congestion window (Allman et al., 1999).

TCP's mechanisms for congestion control have been used with considerable success. There are, however, circumstances where losses occur that are not caused by congestion. When congestion avoidance is incorrectly applied also for non-congestion related losses, TCP performance is degraded. Protocols with similar congestion control, such as the Datagram Congestion Control Protocol (DCCP) (Kohler et al., 2004) and the Stream Control Transfer Protocol (SCTP) (Stewart, 2000), share this basic problem. One important communication environment in which non-congestion related losses may occur is wireless networks. Two major causes of non-congestion related losses in this environment are wireless link errors and mobility induced losses resulting from the inability to do perfect hand-offs. In this paper we focus on wireless link errors.

A common approach to handle wireless link errors is to augment the lower layers with forward error correction (FEC) and/or link-layer retransmissions. This is done in many operational wireless networks. As perceived by TCP, link-layer retransmissions transform the unreliability of the wireless medium into round-trip time variations. In some instances these variations can cause spurious timeouts which incur both an unnecessary retransmission and a reduction of the congestion window. The D-SACK (Floyd et al., 2000) and TCP Eifel (Ludwig and Katz, 2000) algorithms provide solutions that can detect the spurious retransmissions and undo the unnecessary window reduction. Although link-layer retransmissions effectively shield TCP from bit errors over the wireless channel, they are not practical in all situations and hence there are occasions when TCP is subjected to non-congestion losses resulting from the traversal of a wireless link.

A considerable amount of research has been done on solutions to the performance degradation problem that occurs when wireless link errors are misinter-

Garcia J. and Brunstrom A. (2004).
AN EXPERIMENTAL STUDY ON THE PERFORMANCE AND FAIRNESS OF LOSS DIFFERENTIATION FOR TCP.
In *Proceedings of the First International Conference on E-Business and Telecommunication Networks*, pages 410-419
DOI: 10.5220/0001403904100419
Copyright © SciTePress

preted as congestion by TCP. Examples include using split connections (Bakre and Badrinath, 1995), TCP-aware local retransmissions (snoop) (Balakrishnan et al., 1995) or loss differentiation at the transport layer (Balan et al., 2002; Cen et al., 2002; Garcia and Brunstrom, 2002; Kim et al., 1999) which is the focus of this paper.

Loss differentiation is the process of discriminating between different loss causes, making it possible to differentiate between congestion losses and other types of losses. Loss differentiation makes it possible to treat congestion losses and losses caused by bit errors on a wireless link differently, not applying the standard congestion control behavior for wireless link losses. The work presented in this paper examine different aspects of loss differentiation using an actual TCP implementation modified to use loss differentiation. The performance gains of using receiver based loss differentiation are examined. The results show that the gains are considerable in most cases, which is consistent with previous studies. Further, we relate these results to the performance of our simple multiple connections approach. The results from these experiments show that the multiple connections approach considerably improve the throughput when corruption losses are present. In some cases, multiple connections perform better than loss differentiation. Finally, we examined the fairness aspects of loss differentiation. The results show that when a corrupting link is shared between a mix of normal TCP user flows and loss differentiating user flows, the gain for the loss differentiation flows in some instances came at the expense of a noticeable throughput reduction for the competing normal TCP flows.

The outline of this paper is as follows. In the next section a background on loss differentiation is provided followed by a section presenting the results. Finally, the conclusions of the work are presented.

2 LOSS DIFFERENTIATION

As discussed above, loss differentiation is the process of classifying a loss as being caused either by congestion in the network or by some other event, typically corruption of some bits by a wireless link. This section presents a selection of loss differentiation schemes, grouped together according to where in the network support for loss differentiation is placed: within the network infrastructure, at the sender-side or at the receiver-side. The placement typically influences the loss differentiation precision that can be obtained. The precision of a loss differentiation scheme describes how well the scheme distinguishes between congestion losses and wireless losses.

Loss differentiation schemes that rely on infrastructural support typically require some changes to the wireless link base station. Although they may demand considerable modifications, the precision of these schemes is generally very good. Examples include (Cobb and Agrawal, 1995) who describe a solution in which base stations send first-hop and last-hop acknowledgments (acks) for the traffic going from and to a mobile host, respectively. The extra acks allow the sender to infer where the loss occurred, and also works for mobile-to-mobile communication. Another example is the syndrome approach by (Chen et al., 2002) which requires that the base station counts the number of packets sent per flow, and then forwards this number to the receiver in a TCP option. The receiver can then infer if a packet loss occurred on the wireless link. By employing the support of routers, the CETEN approach described by (Krishnan et al., 2002) notifies the sender of the cumulative non-congestion losses along a path.

Sender-based loss differentiation schemes require modification only to the sending end-host and are based solely on processing the incoming ack stream to infer the network state. Losses occurring when the network is in a congested state are then classified as congestion losses. If the network, as perceived by the sender, is in a non-congested state, losses are classified as wireless. However, relying on the ack stream introduces considerable uncertainty. This uncertainty translates to low loss differentiation precision. The precision also varies with factors such as traffic load, reordering, amount and distribution of cross traffic, buffer sizes, link delays, queue sizes and mechanism parameterization. Examples include (Kim et al., 1999) who suggest the use of Linear Increase/Multiplicative Decrease with History (LIMD/H). In this scheme the effective transmission rate history and congestion throttling history are maintained to improve precision. Depending on the relation of the current rate to the effective rate, losses are classified into one of three categories: congestion loss, probe loss, or non-congestion loss. Another example is (Barman and Matta, 2002) who present a scheme named NewReno-FF. This scheme uses an adaptive Flip Flop filter for the ack round-trip times (RTTs) in addition to the usual TCP exponential weighted moving average filter. The underlying assumption is that for packets that suffer random losses, the observed RTT varies significantly more than when congestion loss occurs.

Receiver-based loss differentiation schemes require changes only to the receiver side in order to perform loss differentiation. They do not share the sensitivity of the sender-based predictors to the back-channel conditions. However, since they are receiver-based, they require a loss notification mechanism to convey the loss differentiation information to the sender. The precision of receiver based schemes is typically very favorable compared to the sender-based schemes. Examples include (Cen et al., 2003) who suggest schemes which are based on the time relations of incoming packets.They also consider the fairness aspects of using loss differentiation. Another class of receiver based schemes are based on the observation that wireless corruption often do not cause the whole packet to be lost, but rather a number of bits in the packet become corrupted. These corrupt bits will then lead to checksum failure. When a checksum control function detects an erroneous checksum the packet is discarded. The fact that data was discarded due to a checksum error is however not known outside the checksum control function. By extending the checksum control the discarded packet can be classified as a wireless loss and not congestion. Examples include (Balan et al., 2002) who describe TCP HACK, a scheme that uses a TCP option containing a checksum for the TCP header. When a corrupted packet arrives, a correct header checksum guarantees the integrity of the header information which is used to map the corrupted packet to the correct flow. Another example is (Garcia and Brunstrom, 2002) who discuss a similar approach but without the requirement of a checksum option.

Of particular interest is the receiver based loss differentiation used in the Datagram Congestion Control Protocol (DCCP) (Kohler et al., 2004). DCCP is a transport protocol that provides an unreliable but congestion-controlled flow of datagrams. DCCP allows for flexibility as to the details of the congestion control by using congestion control identifiers (CCID) that govern the exact behavior of the congestion control. Two different CCIDs are currently being proposed; TCP-like congestion control (CCID 2) and TFRC congestion control (CCID 3). DCCP has a flexible checksumming mechanism in the header which can be set to cover only header information, header information and partial data or the whole packet. In addition to the checksum in the header, DCCP also has a data checksum option. If the data checksum option is used, a packet having an incorrect data checksum and a correct header checksum can be classified as a corruption loss. For DCCP, corruption errors that do not affect the header can thus be detected and mapped to the correct flow. For the TCP-like congestion control, loss notification is then performed by the data dropped option of DCCP, which allows the receiver to specify that packets were dropped due to corruption. Loss differentiation in DCCP with TCP-like congestion control is thus quite similar to the scheme used in the experiments in this paper as discussed in the next section.

Although this paper specifically reports on experiments on the performance and fairness of loss differentiation, we note that there are other schemes that consider the problem of wireless losses for TCP. They do not explicitly use loss differentiation, but rather rely on bandwidth estimation techniques to improve performance. Examples of such work include TCP Westwood+ (Mascolo et al., 2004), WTCP (Sinha et al., 2002) and TCP Real (Zhang and Tsaoussidis, 2001). TCP Real has lately been extended with loss classification functionality (Zhang and Tsaoussidis, 2004).

3 EXPERIMENTAL SETUP

This paper reports on performance and fairness experiments performed using a real TCP implementation. In order to perform experiments a checksum-based loss differentiation mechanism was implemented in the FreeBSD 4.5 kernel. In short it works as follows[1]. When a corrupted packet reaches the IP-input function, the IP header checksum is controlled. If the checksum is invalid, the packet is discarded since the IP-addresses are needed to map the packet to a flow. In this case, the loss is implicitly classified as a congestion loss. If the IP header checksum is correct, the packet is forwarded to the TCP input function. Since it is a corrupted packet the TCP checksum will be erroneous, and the packet would normally be discarded. However, when checksum based loss differentiation is used, a flow match is attempted before discarding the packet. A flow match uses the IP-addresses and the port numbers. After the flow match the sender is notified that a corruption loss has occurred via the use of a TCP-option. The implementation provides a high precision since it can classify losses accurately as long as there are no bit errors in the sensitive header fields. The exact probability of classifying a corruption loss as a congestion loss depends on the bit error distribution and the relation between the header size and packet size, but in the experiments presented in this paper the probability is below 0.03. There is no possibility of falsely classifying a congestion loss as a corruption loss.

When performing the experiments a setup consisting of a client, a server and a router/emulator was used as illustrated in Figure 1. The router/emulator ran the Dummynet software (Rizzo, 1997) to emulate various link conditions. Three different bandwidths (1Mbps, 384Kbps, 64Kbps), three different delays (10ms, 50 ms, 200 ms), and uniformly distributed bit errors with seven bit error rates (BER) varying between 0 and $1.2 * 10^{-5}$ were used. The BER values were chosen so that they correspond to packet loss ratios (PLR) of approximately 1, 2, 5, 7.5, 10 and 15 percent for the

[1]This paper focuses on performance and fairness aspects of receiver-based loss differentiation so we only provide a short description. Further discussion of relevant details are available in (Garcia and Brunstrom, 2002).

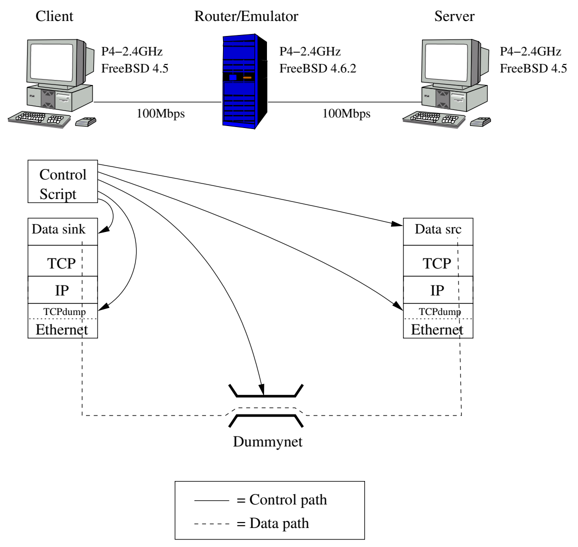

Figure 1: Physical experiment setup

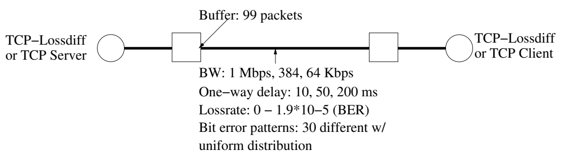
Buffer: 99 packets
TCP–Lossdiff or TCP Server
TCP–Lossdiff or TCP Client
BW: 1 Mbps, 384, 64 Kbps
One–way delay: 10, 50, 200 ms
Lossrate: 0 – 1.9*10–5 (BER)
Bit error patterns: 30 different w/ uniform distribution

Figure 2: Scenario 1 logical experiment setup

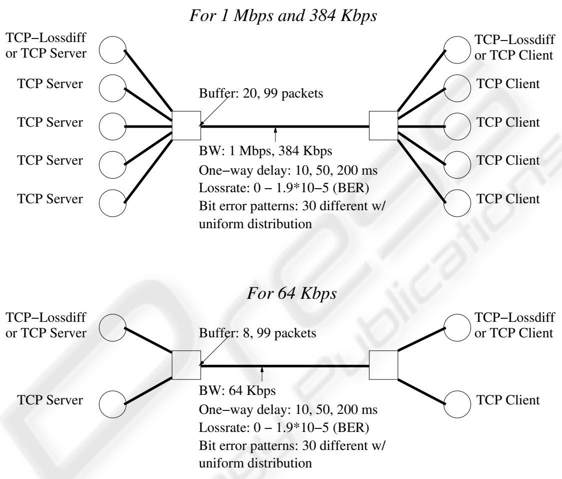
For 1 Mbps and 384 Kbps
TCP–Lossdiff or TCP Server
TCP Server
TCP Server
TCP Server
TCP Server
TCP–Lossdiff or TCP Client
TCP Client
TCP Client
TCP Client
TCP Client
Buffer: 20, 99 packets
BW: 1 Mbps, 384 Kbps
One–way delay: 10, 50, 200 ms
Lossrate: 0 – 1.9*10–5 (BER)
Bit error patterns: 30 different w/ uniform distribution
For 64 Kbps
TCP–Lossdiff or TCP Server
TCP Server
TCP–Lossdiff or TCP Client
TCP Client
Buffer: 8, 99 packets
BW: 64 Kbps
One–way delay: 10, 50, 200 ms
Lossrate: 0 – 1.9*10–5 (BER)
Bit error patterns: 30 different w/ uniform distribution

Figure 3: Scenario 2 logical experiment setup

packet size used (1500 bytes). To be noted is that we used a modified version of the Dummynet FreeBSD kernel code that enabled us to control the bit errors using bit error pattern files. The same pattern file was reused when testing with and without loss differentiation, thus creating identical positioning of the bit errors for both cases. To avoid the risk of a specific bit error pattern skewing the results, 30 different, randomly generated, bit error pattern files were used for each parameter combination. The receiver window was set to its default value of 64kB.

Two test scenarios were set up. In Scenario 1 the focus was on evaluating the performance improvements for a single flow and confirm and expand on the positive results reported in earlier studies. The Scenario 1 setup measured the throughput of a single TCP flow. Figure 2 show the logical setup and a summary of the parameters for scenario 1. One measurement set was performed with a normal TCP flow and one set with a TCP flow using loss differentiation. The buffer-size in the router was set to 99 packets, which is larger than the receiver window. Hence there is no congestion losses in this scenario, and all losses will be caused by bit errors on the link.

Scenario 2, which is illustrated in Figure 3, was setup to examine both the throughput of multiple connections and the fairness effects of loss differentiation. To examine the throughput of multiple connections between the same endpoints five flows (1Mbps & 384 Kbps) or two flows (64Kbps) were started simultaneously. For these experiments the buffer-size in the router was set to 99 packets to be comparable to the single flow case.

When a wireless link is shared between flows, some of which employ loss differentiation and some which do not, there is a possibility that the loss differentiating flow could unfairly decrease the throughput of the other, non loss differentiating, flows. To examine this, five flows (1Mbps & 384 Kbps) or two flows (64Kbps) were started simultaneously and competed for the available link capacity. The first set of measurements had four or one normal TCP flow(s) competing against a normal TCP flow. The second set had four or one normal TCP flows competing against a flow with loss differentiation. The router buffer-size was set to 20 packets (1Mbps & 384 Kbps) or 8 packets (64Kbps) thus causing buffer overflow, resulting in congestion losses.

For all experiments 1 Mbyte of data was transferred to get each throughput value. The throughput values for the 30 different bit error patterns were for each combination of parameters averaged to a mean value and a 95% confidence interval was calculated.

4 RESULTS

4.1 Effectiveness of loss differentiation

By utilizing loss differentiation to distinguish corruption losses from congestion losses improved performance is expected. The results for the Scenario 1 experiments are shown in Figures 4-10. Note that the relative positions of the data points are comparable between figures since the scale is normalized to the maximum throughput for the different bandwidths. Due to space restrictions two of the 200ms figures are not shown. The mean throughput of one normal TCP flow is shown by the filled square marks and the mean throughput of one TCP flow with loss differentiation (called ld flow in the figures) is shown by the filled circle marks. The 95% confidence intervals are also included in the figures.

As expected, regular TCP pays a considerable performance penalty in the presence of bit-errors. The positive effect of loss differentiation is evident looking at the higher throughput obtained by the flow employing loss differentiation. Loss differentiation is clearly beneficiary, and the relative benefit increases as the loss rate increases. A general trend is that regular TCP becomes more sensitive to losses at high bandwidths (BW) and, to a lesser extent, at high link delays (D). Both increased bandwidth and increased delay lead to an increase in the bandwidth-delay product (BW*D). A larger BW*D requires a larger average congestion window size to keep the link fully utilized. The requirement of a larger average window makes the flow more sensitive to losses. The trends visible in these results confirm the positive results in previous simulation based studies such as (Krishnan et al., 2002) and (Cen et al., 2003).

The experimental results discussed above were obtained with a large router buffer (99 packets). Consequently, no congestion losses occurred since the receiver window restricted the sender before a router overflow could occur. In addition to these experiments, we also performed experiments with smaller buffer sizes. This lead to the occurrence of both congestion and corruption losses. The results showed that the buffer size difference in general had little effect, but the additional losses generated by the buffer overflows lowered the throughput somewhat. This was most visible for the loss differentiating flow. This flow experienced the most buffer overflows since the normal flow in most cases did not have sufficient throughput to cause a buffer buildup. Figure 11 shows the results for the 1 Mbps and 10 ms case with a buffer size of 20 packets. These results are comparable with the results for a large buffer in Figure 4. Examination of the graphs reveal only small differences that can be attributed to the reduced buffer size and the resulting additional congestion losses.

4.2 Effectiveness of multiple connections

As seen in the previous section, there is a large performance gain in using loss differentiation. This subsection focus on an alternate way to address the problem of decreased throughput when traversing wireless links, namely to open multiple connections between the same end-points. This approach is commonly used by web browsers to speed page load times when downloading multiple objects on a web page. We propose an adaptation of this approach, to encompass general data transfer, and with the aim of improving the throughput over a corrupting link rather than improving page load times. In this presentation we focus on the performance aspects of this approach. We do not consider the more practical aspects such as application layer re-assembly and other issues related to modification of the applications to use multiple connections instead of single connections. We do, however, note that this approach can be realized by application layer modifications only, whereas the loss differentiation approach discussed in the previous section requires changes at the transport and lower layers.

The results when using multiple connections are shown as empty squares in Figures 4-10. As evident in the graphs there is a considerable gain in using multiple connections, in some cases it is even more advantageous to use multiple connections than to use a single loss differentiating flow. When we examine the reason for this improved performance, we find that it is twofold; additive stream throughputs and additive initial congestion window.

As stated by (Mathis et al., 1997), the throughput of a TCP flow is related to the amount of losses. Relevant in this context is that the throughput TCP_{BW} is limited by the error rate. If the corruption error rate is such that the throughput of a flow is limited by the corruption errors and not the link bandwidth, then using n multiple connections allows an aggregate throughput closer to the link bandwidth, or up to $n * TCB_{BW}$. The throughput of each flow thus adds to the total aggregate throughput that can be achieved by multiple connections. This improved throughput is visible in Figures 4-7 and Figure 10. If the single flow throughput is not severely limited by corruption errors, then the relative gain is not so large. This is the case in Figures 8-9 where the single flow bandwidth is limited by the link capacity and not by corruption errors.

The other effect improving the results for multiple connections is the additive initial congestion window.

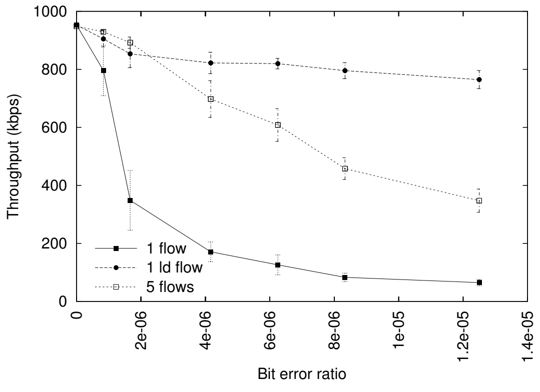

Figure 4: Scenario 1 results (1Mbps, 10ms)

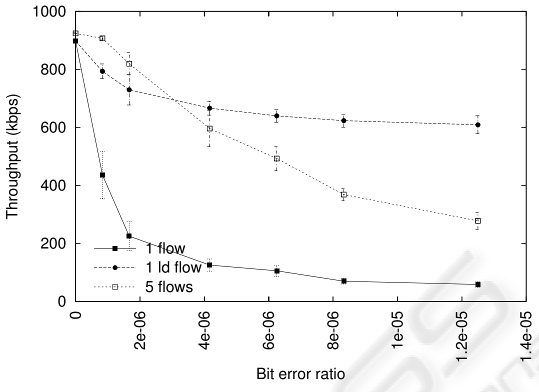

Figure 5: Scenario 1 results (1Mbps, 50ms)

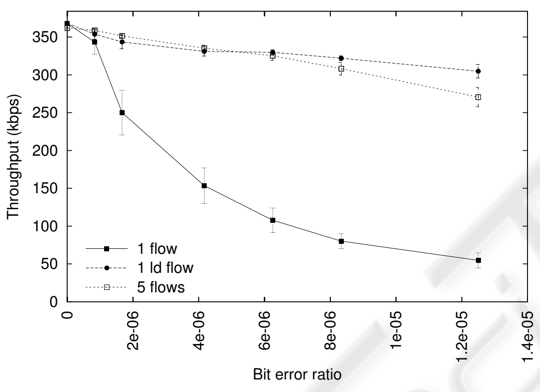

Figure 6: Scenario 1 results (384kbps, 10ms)

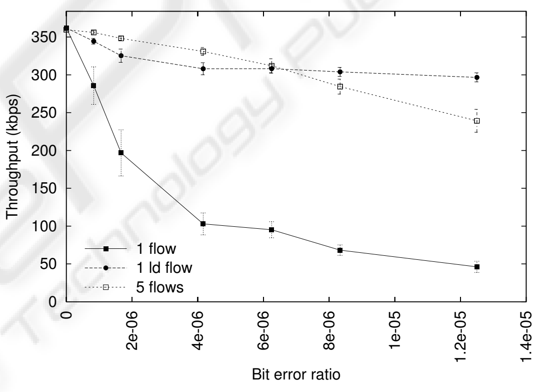

Figure 7: Scenario 1 results (384kbps, 50ms)

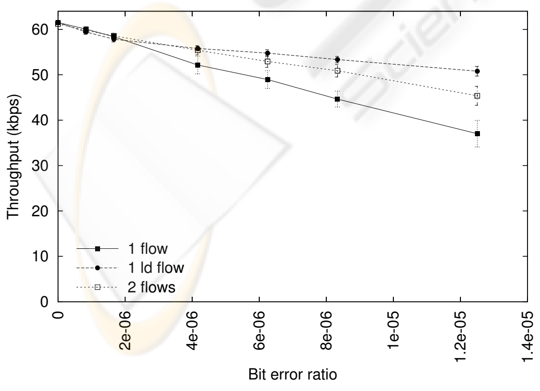

Figure 8: Scenario 1 results (64kbps, 10ms)

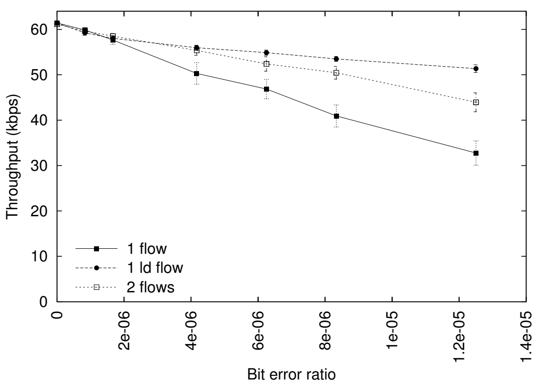

Figure 9: Scenario 1 results (64kbps, 50ms)

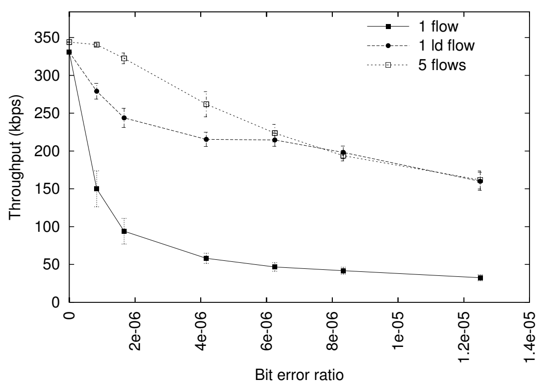

Figure 10: Scenario 1 results (384kbps,200ms)

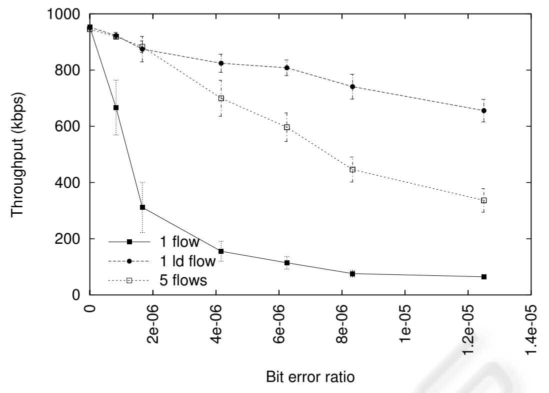

Figure 11: Scenario 1 results with smaller buffer (1 Mbps, 10ms)

For a single flow the initial congestion window is normally 2 segments, which is then doubled each round trip time during slow start. Until the window is large enough to fully utilize the link, (i.e. window>BW*D) the link is underutilized. For n multiple connections that are started concurrently, the aggregate initial congestion window is instead $n * 2$. Consequently, this window reaches full utilization sooner, leaving the link underutilized for a shorter period. The influence of this effect is dependent of the length of the transfer, having greater impact on shorter connections. In the experiments 1Mbyte was transferred, and for some of the measurements the effect was large enough to manifest itself clearly in the graphs. For 1Mbps and 384Kbps the throughput of multiple connections is actually better than for a loss differentiating connection for small bit error rates. The difference is larger for the longer delays, which is consistent with the above discussion. Larger delay gives larger BW*D which requires a larger congestion window to keep the link from being underutilized. At startup, a larger required window takes longer to achieve for a single connection than for multiple connections, which leaves the link underutilized for a longer period for the single connection.

Since employing multiple connections can be done at the application level, this approach may in some cases be preferable compared to modifying the network stack to employ loss differentiation. However, this approach also makes the end host behavior more aggressive compared to using a single flow as discussed in the end of the next section. The performance and fairness of multiple TCP connections together with non-congestion losses has previously been examined by(Hacker et al., 2002). However, they used simulation to examine the effect of non-congestion losses on large bulk TCP transfers in the high performance Abilene network, rather than wireless networks causing corruption losses.

4.3 Fairness of loss differentiation

The previous two subsections focused on evaluating the performance when a single user is traversing a corrupting link. Another case is when the corrupting link is shared by several users who compete for the available link bandwidth. In this case there may be some users who employ loss differentiation and some who do not. The fairness effects of using loss differentiation are thus relevant and are highlighted by the results for the Scenario 2 measurements, as shown in Figures 12-18. A total of five flows (or two for the 64Kbps case) were simultaneously competing for the link bandwidth. The graphs show the five flows as groups of four normal TCP flows plus a single flow. The single flow can be either a normal TCP flow or a loss differentiating flow. Each graph shows one set of measurements where the single flow is a normal flow competing against the four normal flows, or where the single flow is a loss differentiation flow competing against the four normal flows.

Using Figure 12 as example, it is visible that the throughput of a single normal TCP flow (filled squares) decrease when the bit error ratio increase. The four competing normal TCP flows (empty squares) also decrease their aggregate throughput as the bit error ratio increase. The set of measurements using an ld flow shows quite different results. The throughput of the single ld flow (filled circles) actually increase as the bit error ratio increase. To a certain extent, this increase might be done without causing additional unfairness since the competing normal TCP flows often cannot fully utilize a link when the bit error rate increases. Bandwidth that could not be used by the competing flow can instead be used by the ld flow. As was shown in Section 4.1, an ld flow is practically insensitive to the bit errors that hamper the throughput for normal TCP flows. However, as shown by the empty circles, the aggregate throughput of the

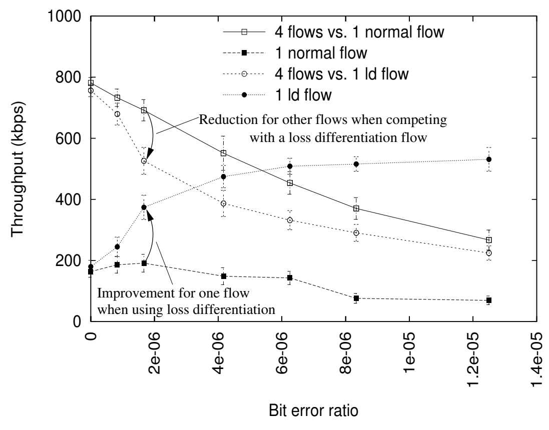

Figure 12: Scenario 2 results (1Mbps, 10ms)

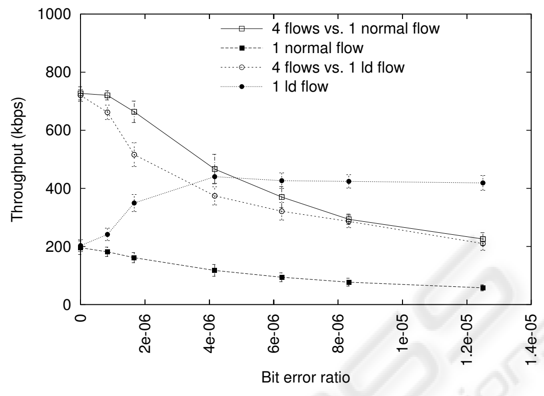

Figure 13: Scenario 2 results (1Mbps, 50ms)

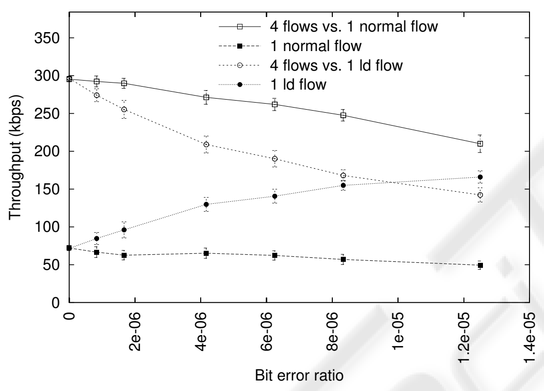

Figure 14: Scenario 2 results (384kbps, 10ms)

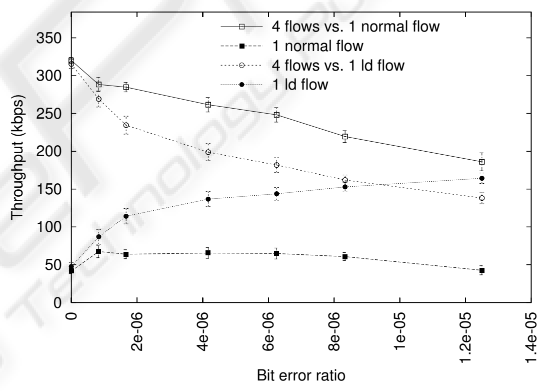

Figure 15: Scenario 2 results (384kbps, 50ms)

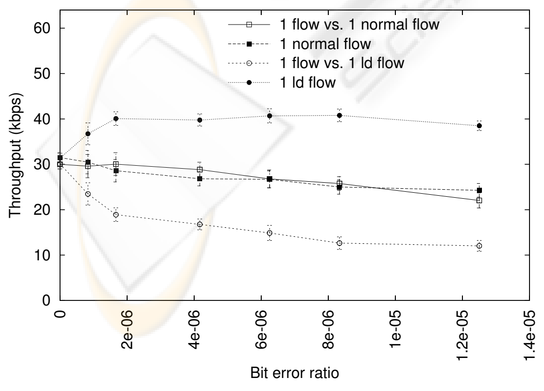

Figure 16: Scenario 2 results (64kbps, 10ms)

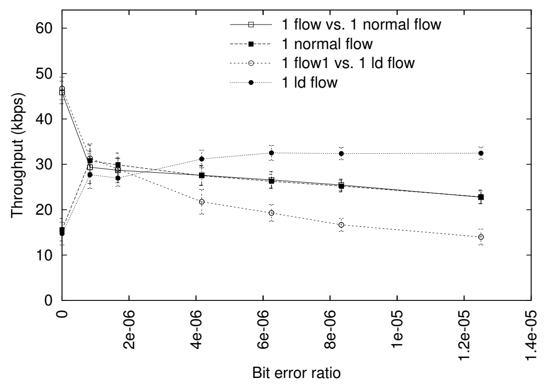

Figure 17: Scenario 2 results (64kbps, 50ms)

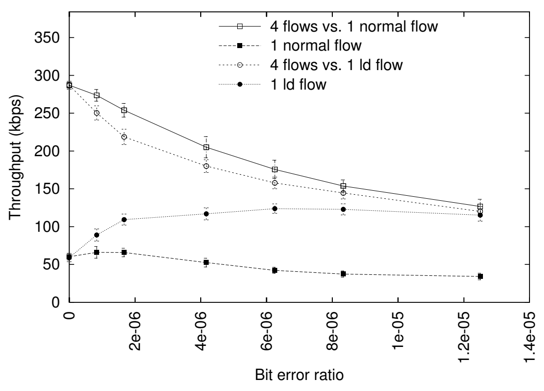

Figure 18: Scenario 2 results (384kbps, 200ms)

four normal TCP flows is lowered considerably when they compete with an ld flow. The difference between the two aggregate throughputs shows the degree of unfairness that comes as a result of the ld flow's ability to work well over a corrupting link and partly taking bandwidth from the competeing flows. The unfairness effect is dependent on the parameter combination. The aggregate throughput of the competing flows is reduced by up to 35 percent in Figure 14, but there is considerably less reduction in other cases. These results show that fairness indeed can be an issue when employing loss differentiation, and highlights the need to also evaluate the fairness implications of the various schemes proposed. The somewhat unexpected results for a bit error ratio of 0 in Figure 17 is caused by loss synchronization effect between the two flows in the 8 packet buffer. When bit error losses are also present the synchronization is broken.

The scenario 2 can also be used to illustrate graphs the unfairness of the multiple connections scheme. Assume that two users share a corrupting link. One user employs the multiple connections scheme to increase his throughput over the corrupting link and uses four normal TCP connections to transfer his data. The other user does not employ any scheme to improve throughput over the corrupting link. For this example, the throughput of the two users would be as illustrated in Figures 12-18. The first user, utilizing multiple connections, would in this case get the throughput indicated by the empty squares and the second, normal, user would get the throughput indicated by the filled squares. Clearly this is far from fair sharing of the link and it illustrates the unfairness of the multiple connections approach when it is used over a shared bottleneck link. The same unfair behavior is however to some extent already present in the Internet, in this case caused by the multiple connections employed by some web browsers to improve page load times.

5 CONCLUSIONS

This paper reports on experiments performed with a FreeBSD kernel implementation of a receiver-based loss differentiation scheme. With regards to the performance of loss differentiation, the results largely concur earlier work both concerning the problems that TCP has when used over wireless links with bit errors, and the effectiveness of receiver-based loss differentiation. Additionally, experiments were performed using multiple regular TCP connections working as one aggregate "virtual" connection to improve the throughput over a link with bit errors. The results show that employing multiple regular TCP flows improve the throughput considerably, especially for the lower bit error rates. In several cases, the multiple connections approach had better throughput than the loss differentiation approach when run over the same bit error generating link. However, the multiple connections approach has severe fairness issues if used over a bottleneck link shared with other users. Nevertheless, multiple connections can be easily achieved at the application level without requiring any protocol stack or other changes, so it is an interesting approach worthy of further study.

The fairness aspect is also relevant in conjunction with loss differentiation. Fairness issues arise when several users share a link with bit errors and some users employ loss differentiation and some do not. These effects are studied and the results show that a single loss differentiating user in some cases may unfairly reduce the bandwidth of the non loss differentiating users with up to 35 percent. However, there are also cases where loss differentiation has marginal fairness implications. All in all, the results highlight the need to consider the fairness issues of the various loss differentiation schemes proposed in the literature.

Many of the results in this study are applicable also to other receiver-based loss differentiation schemes. In particular, the DCCP protocol can employ a TCP-like congestion control and a loss differentiation functionally very similar to the implementation examined in this paper. The presented results should thus also provide a good indication of the behavior of DCCP loss differentiation.

REFERENCES

Allman, M., Paxson, V., and Stevens, W. (1999). RFC 2581: TCP congestion control.

Bakre, A. and Badrinath, B. R. (1995). I-TCP: Indirect TCP for mobile hosts. *15th International Conference on Distributed Computing Systems*.

Balakrishnan, H., Seshan, S., Amir, E., and Katz, R. H. (1995). Improving TCP/IP performance over wireless networks. *In proc. 1st ACM Int'l Conf. on Mobile Computing and Networking (Mobicom).*

Balan, R. K., Lee, B. P., Kumar, K. R. R., Jacob, L., Seah, W. K. G., and Ananda, A. L. (2002). TCP HACK: a mechanism to improve performance over lossy links. *Computer Networks*, 39(4):347–361.

Barman, D. and Matta, I. (2002). Effectiveness of loss labeling in improving TCP performance in wired/wireless networks. *CS Dept Technical report 2002-016, Boston University*.

Cen, S., Cosman, P., and Voelker, G. (2002). End-to-end differentiation of congestion and wireless losses. *Proc. Multimedia Computing and Networking (MMCN2002), San Jose, CA*, pages 1–15.

Cen, S., Cosman, P. C., and Voelker, G. M. (2003). End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking*, 11(5):703–717.

Chen, W.-P., Hsiao, Y.-C., Hou, J. C., Ge, Y., and Fitz, M. P. (2002). Syndrome: a light-weight approach to improving TCP performance in mobile wireless networks. *Wireless Communications and Mobile Computing*, (2):37–57.

Cobb, J. A. and Agrawal, P. (1995). Congestion or corruption? A strategy for efficient wireless TCP sessions. *IEEE Symposium on Computers and Communications, Alexandria, Egypt*, pages 262–268.

Floyd, S., Mahdavi, J., Mathis, M., and Podolsky, M. (2000). RFC 2883: An extension to the selective acknowledgement (SACK) option for TCP.

Garcia, J. and Brunstrom, A. (2002). Checksum-based loss differentiation. *Proceedings 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm, Sweden*.

Hacker, T. J., Noble, B. D., and Athey, B. D. (2002). The effects of systemic packet loss on aggregate TCP flows. *Proceedings of the 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland*, pages 1–15.

Kim, T., Lu, S., and Bharghavan, V. (1999). Improving congestion control performance through loss differentiation. *Proceedings International Conference on Computers and Communications Networks (ICCCN99), Boston, USA*.

Kohler, E., Handley, M., Floyd, S., and Padhye, J. (2004). Datagram congestion control protocol (DCCP). *draft-ietf-dccp-spec-06.txt, Work in progress*.

Krishnan, R., Allman, M., Partridge, C., and Sterbenz, J. P. (2002). Explicit transport error notification for error-prone wireless and satellite networks. *BBN Technical Report No. 8333, BBN Technologies*.

Ludwig, R. and Katz, R. H. (2000). The Eifel algorithm: Making TCP robust against spurious retransmissions. *ACM Computer Communication Review*, 30(1):30–37.

Mascolo, S., Grieco, L. A., Ferorelli, R., Camarda, P., and Piscitelli, G. (2004). Performance evaluation of Westwood+ TCP congestion control. *Performance Evaluation*, 55(1-2):93–111.

Mathis, M., Semke, J., and Mahdavi, J. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3).

Postel, J. (1981). RFC 793: Transmission control protocol.

Rizzo, L. (1997). Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41.

Sinha, P., Nandagopal, T., Venkitaraman, N., Sivakumar, R., and Bharghavan, V. (2002). WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2):301–316.

Stewart, R. (2000). RFC 2960: Stream Control Transmission Protocol.

Zhang, C. and Tsaoussidis, V. (2001). TCP Real: Improving real-time capabilities of TCP over heterogeneous networks. *Proceedings of the 11th IEEE/ACM NOSSDAV*.

Zhang, C. and Tsaoussidis, V. (2004). Error differentiation with measurements based on wave patterns. *Computer Communications*, 27(10):989–1000.