Improved coexistence and loss tolerance for delay based TCP congestion control

David A. Hayes*, and Grenville Armitage*

*Centre for Advanced Internet Architectures,
Swinburne University of Technology Melbourne, Australia
Email: {dahayes,garmitage}@swin.edu.au

***Abstract*—Loss based TCP congestion control has been shown to not perform well in environments were there is non-congestion related packet losses. Delay based TCP congestion control algorithms provide a low latency connection with no congestion related packet losses, and have the potential for being tolerant to non-congestion related losses. Unfortunately, delay based TCP does not compete well with loss based TCP, currently limiting its deployment.**

We propose a delay based algorithm which extends work by Budzisz et al. [1] to provide tolerance to non-congestion related losses, and better coexistence with loss based TCP in lightly multiplexed environments. We demonstrate that our algorithm improves the throughput when there are 1 % packet losses by about 150 %, and gives more than 50 % improvement in the ability to share capacity with NewReno in lightly multiplexed environments.

I. Introduction

The internet's wide-spread utility over the past 25+ years owes much to the transmission control protocol (TCP) [2], the dominant transport protocol for internet-based applications [3]. The relatively modern *NewReno* variant [4] balances two key goals: Provide reliable transfer of byte-streams across the IP layer's unpredictable packet-based service, and minimise congestion inside end hosts and the underlying IP network(s) while maximising throughput [5]. The latter goal has been an active and challenging area for academic and industry research into congestion control (CC) techniques [6].

Most approaches to TCP CC have traditionally treated packet loss as an indicator of network or end-host congestion (not only retransmitting the lost packet, but temporarily slowing down their transmission rate). While probing for a path's maximum capacity, traditional TCP will itself also tend to induce packet losses (as it has no other way of determining when the path's capacity is reached).

This behaviour is broadly reasonable where internet traffic flows over link layers with extremely low intrinsic bit error rates (such as wires or optical fibers) and the traffic is largely loss-tolerant. However, it is increasingly *unreasonable* in today's internet where we see a mix of loss-tolerant and loss-sensitive traffic (such as TCP intermingled with UDP-based interactive online games or Voice over IP) flowing over a mixture of fixed and wireless link layer technologies (such as 802.11-based wireless LANs, 802.16 WiMAX last-mile services, IEEE 802.15.4 ZigBee wireless links to smart energy meters, as so on).

We face two issues. Loss-sensitive applications tend not to appreciate the packet losses induced as intermingled TCP flows cyclically probe for path capacity. Wireless link technologies often exhibit low levels of intrinsic packet loss entirely unrelated to congestion, so TCP's traditional response of "slowing down" can be a counter-productive.

There is emerging interest in a different category of TCP – *delay-based* CC algorithms that utilise trends in round trip time (RTT) estimates to infer congestion along an end to end path. Such algorithms can optimise their transmission rates without inducing packet losses, and offer the potential to be insensitive to packet losses that aren't being caused by congestion.

In this paper we propose a significant enhancement to a delay-based TCP algorithm first proposed by Budzisz et al. [1]. Our approach shows improved performance in the face of packet losses, and can better share capacity with traditional NewReno TCP in lightly multiplexed environments (such as home Internet scenarios).

The rest of our paper is structured as follows. Section II summarises the key delay-based CC algorithms to date. Section III summarises the algorithm of Budzisz et al. [1], whilst our proposed algorithm is described in Section IV. Section V covers our experimental analysis and results, while future work and conclusions are covered in sections VI and VII respectively.

II. Background

Proposals for using delay based congestion indications for TCP have been around since 1989, when Jain [7] proposed his CARD (Congestion Avoidance using Round-trip Delay) algorithm. Table I summarizes some of the key proposals since then that have sought to utilise variations in packet transit delay as an earlier indication of congestion than may be achieved by waiting for packet loss. These proposals may be distinguished by:

- the way they measure delay — RTT, one way delay, per packet measurements, etc;
- how they infer congestion — set thresholds, etc;

978-1-4244-8389-1/10/$26.00 ©2010 IEEE

- how they react to congestion by changing the congestion window (`cwnd`) — traditional TCP's Additive Increase Multiplicative Decrease (AIMD), Additive Increase Additive Decrease (AIAD), Multiplicative Increase Multiplicative Decrease (MIMD), and equation based window size setting.

All delay based methods of inferring congestion rely on there being a correlation between delay and congestion. Studies by Martin et al. [17] look at a number of TCP traces and show that there is only a low correlation between loss events and increases in RTT. McCullagh and Leith [18] show that this is not an obstacle to congestion control, since it is the aggregate behaviour of flows which is important.

Despite their promise of low delay and zero congestion related loss, delay based TCPs are difficult to deploy due to the way they interact with standard loss based TCP.

III. Hamilton Delay based congestion control algorithm (HD)

Leith et al. [16] expound the case for delay-based Additive Increase Multiplicative Decrease (AIMD) congestion control, which promises to provide end to end control with high utilization, low delay and zero congestion-induced packet loss. This idea was enhanced (in a similar manner to PERT [15, 19]) by Budzisz et al. [1] to improve fair coexistence with loss based TCP algorithms[1].

On receipt of every ack, `cwnd` (w) is evaluated as follows:

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(q_i) \\ w_i + \frac{1}{w_i} & \text{otherwise} \end{cases} \quad (1)$$

where $g(q_i)$ is the backoff probability function shown in figure 1, $X \in [0, 1]$ is random number, $p_{\max}$ is the maximum probability of backoff, $q_{\max} = \text{RTT}_{\max} - \text{RTT}_{\min}$ is an estimate of the maximum observed queueing delay, $q_{\min}$ is a target minimum queueing delay, and q_{th} is a threshold that divides regions A and B.

When loss based flows are on the link, the queue is pushed into region B. The delay based flows have a lower probability of backoff in this region, enabling them to receive a fairer share of the available bandwidth. When loss based flows are no longer on the bottleneck link region B is unstable, ensuring delay based flows converge to a low delay state in region A.

A. Shortcomings of HD

Delay based congestion control algorithms in general react to congestion much earlier than loss based algorithms. This makes it hard for them to compete fairly with loss based flows. But HD has two specific shortcomings.

1) Probability of backoff per RTT: The probability of backoff per RTT increase with increasing cwnd. This helps ensure flows with smaller cwnds can compete better, but diminishes the performance of flows with large cwnd.

[1]We call this "Hamilton" delay based (HD) congestion control as the primary authors are from the Hamilton Institute

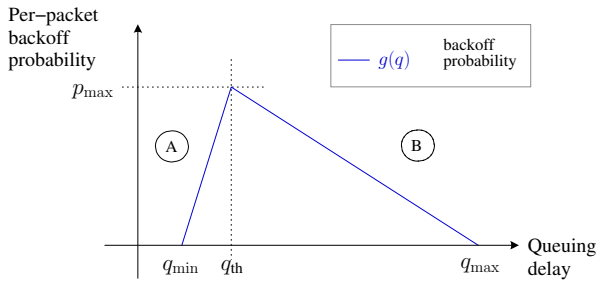


Fig. 1. Per-packet backoff probability as a function estimated queueing delay[1]

2) Probability of backoff when competing with loss based flows: Traditional loss based TCP congestion control algorithms keep increasing cwnd, and thus their potential transmission rates, until they detect loss. When enough loss based flows concurrently use a bottleneck link, this has the effect of keeping the bottleneck queue close to maximum capacity. As shown in Figure 1's region B, HD decreases its probability of backing off due to delay based congestion to zero at the maximum inferred queueing delay. However, the inferred queueing delay will fluctuate up to $q_{\max}$. Consequently HD will backoff due to delay as well as loss, giving it a smaller share of the available bandwidth than the competing loss based flows.

With HD, the probability of having these extra backoffs in a given RTT will increase as cwnd increases. This problem is exacerbated when fewer flows are multiplexed together on the bottleneck link, as this tends result in wider queue fluctuations. Having a relatively small number of TCP flows share a bottleneck link is common in, for example, home internet access scenarios.

IV. Caia–Hamilton Delay based congestion control algorithm (CHD)

Our novel modification to Budzisz et al. [1] (CHD) uses the same probability function shown in Figure 1, but with three key modifications:

- Delay based `cwnd` operations are performed only once per RTT
- We infer (and tolerate) when packet losses may be unrelated to congestion
- We improve coexistence with loss based TCP algorithms

As with HD, CHD only modifies the TCP sender's behaviour. No change is required to existing TCP receivers.

A. Delay based window updates

Our modified delay based backoff operates on `cwnd` in a similar way to equation (1), except that we update once every RTT and use the maximum queueing delay experienced during the last RTT[2]. In other words, if $h_r = \max_r(q_i)$ is the

[2]We measure an RTT from the time a marked packet is sent to when the acknowledgement for that packet (or nearest equivalent when packet losses occur) is received. Then the next packet sent is marked, and so on. The process starts with the first data packet. See [20] for details.

	Delay measurements	Congestion Inference	Congestion Control
Congestion Avoidance using Round-trip Delay (CARD) [7]	RTT	Normalized Delay Gradient, $\left(\frac{\tau_i-\tau_{i-1}}{\tau_i+\tau_{i-1}}\right) > 0$	AIMD ($\beta = \frac{7}{8}$)
DUAL [8]	every 2nd RTT	$\tau_i > \frac{(\tau_{\min}+\tau_{\max})}{2}$	AIMD ($\beta = \frac{7}{8}$)
Vegas[9]	RTT	$\tau_i > \tau_{\min}+\theta$, normalized by the data sent	AIAD
Fast TCP [10, 11]	smoothed RTT	similar to Vegas	smoothed equation based window update at regular (not RTT) intervals(MIMD)
TCP-LP [12]	smoothed one way delay using TCP time stamps	$\overline{d}_i > d_{\min} + \delta(d_{\max} - d_{\min})$	AIMD with a minimum time between successive window decreases
TCP-Africa [13]	smoothed RTT	similar to Vegas	Dual mode: equation based window increase when delay is low, otherwise additive increase. Multiplicative decrease on loss.
Compound TCP (CTCP) [14]	smoothed RTT	similar to Vegas	Dual mode: When delay conditions are favorable it is Reno AMID plus MIMD (two windows are used (delay and loss), otherwise it is Reno AIMD
Probabilistic Early Response TCP (PERT) [15]	RTT and smoothed RTT	dynamic thresholds based on inferred queueing delay ($q_j = \tau_j - \tau_{\min}$)	Random Early Discard (RED) inspired probabilistic reaction to queueing delay, with loss probability matching when $q_j \geq 0.5q_{\max}$.
Hamilton Delay [1, 16]	RTT	threshold based on inferred queueing delay	Probabilistic reaction based on queueing delay and a back off function (see III)

TABLE I

OVERVIEW OF KEY DELAY BASED TCP ALGORITHMS IN TERMS OF DELAY MEASUREMENTS, CONGESTION INFERENCE AND CONTROL WHERE β IS THE MULTIPLICATIVE DECREASE FACTOR, θ REPRESENTS A DELAY THRESHOLD, i^{TH}RTT MEASUREMENT $= \tau_i$, SMALLEST RTT $= \tau_{\min}$, LARGEST RTT $= \tau_{\max}$, AND THE i^{TH} ONE WAY DELAY $= d_i$

maximum queueing delay observed in RTT r, we update w as follows:

$$w_{i+1} = \begin{cases} \frac{w_i}{2} & X < g(h_r) \\ w_i + \frac{1}{w_i} & \text{otherwise} \end{cases} \quad (2)$$

The case where $X < g(h_r)$ represents a delay triggered window reduction.

B. Loss based window updates

CHD's ability to compete fairly with loss based flows involves the use of a *shadow window*. The shadow window, s, is initialised to w at the first hint of competing with a loss based flow, and then kept in check with every delay triggered window reduction.

The heuristic used to keep s and w in check follows:

$$s_{i+1} = \begin{cases} \max(w_i, s_i) & X < g(h_r) \wedge \mathcal{A} \\ \max(w_i, s_i) & X < g(h_r) \wedge h_r > q_{\text{th}} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $\mathcal{A} = h_r < q_{\text{th}} \wedge h_r > h_b$, and h_b is the value of h_r when the last delay triggered w reduction occurred.

The first hint of competing with a loss based flow occurs when the queueing delay begins to increase. Therefore if h_r is in region A, and the delay that triggered the previous congestion indication $h_b < h_r$, then $s_{i+1} = \max(w_i, s_i)$. If h_r is in region B, s is set at every delay based trigger. Otherwise, $s = 0$, since the heuristic guesses CHD is not competing with any loss based flows.

When $s \neq 0$ the shadow window mimics the behaviour of NewReno's congestion window, and is used if a packet loss occurs when the last recorded h was in region B. If a packet loss occurs in region A, w remains unchanged (since it is assumed that this is a non-congestion related loss). CHD's rule for updating w on packet loss is:

$$w_{i+1} = \begin{cases} \frac{\max(w_i, s_i)}{2} & \text{packet loss} \wedge h_r > q_{\text{th}} \\ w_i & \text{otherwise} \end{cases} \quad (4)$$

Equations (2), (3) and (4) allow CHD to both tolerate non-congestion related losses and lose less transmission capacity to loss based TCP flows.

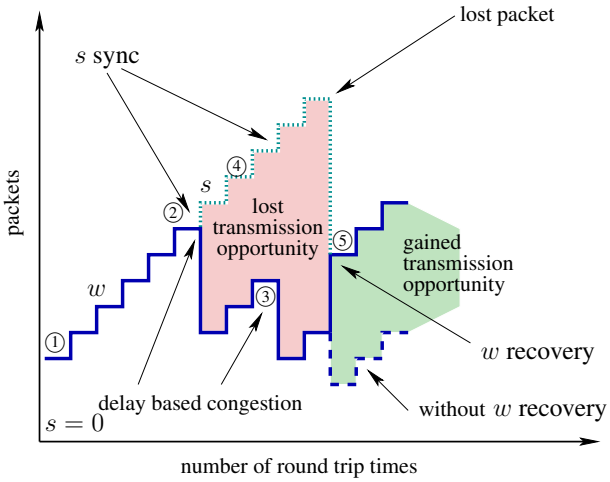


Fig. 2. Interaction of the shadow window, s, and the congestion window w when competing with loss based CC flows.

Figure 2 illustrates how this mechanism works[3]. Referring

[3]Although our actual implementation uses byte based windows, a packet based window is shown for simplicity

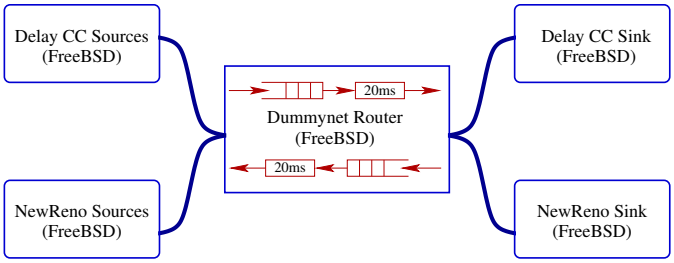


Fig. 3. Experimental Testbed

to the regions indicated by circled numbers:

1) w grows as normal for TCP congestion avoidance, increasing by 1 packet per RTT.
2) When acting on a delay based congestion indication meeting equation (3)'s criteria, s is synchronised to w.
3) w continues to react to delay based congestion
4) s is incremented as NewReno would have been.
5) A packet loss occurs in region B, and w is set to $\frac{s}{2}$ rather than $\frac{w}{2}$ (per equation (4))

Our addition of the shadow window s improves CHD's coexistence with loss based flows. We do not reclaim the lost transmission opportunities, but our approach does lessen the impact of the extra delay based backoffs[4]. (Note that the extra delay based backoffs are still necessary to provide the back pressure that ensures queueing delays revert to region A when there are no loss based flows competing.)

V. Experimental Analysis

We experimentally verified the performance of HD and CHD against NewReno using a FreeBSD-based testbed as depicted in Figure 3. As part of our NewTCP project [21] we implemented both HD and CHD algorithms as new sender-side modules in the FreeBSD 9.0 kernel, using our modular congestion control framework [22] with extensions to better measure the RTT [20].

Our experiments compare the following aspects of NewReno, HD, and CHD

- Tolerance of NewReno, HD, and CHD to non-congestion related losses.
- Sharing dynamics between three homogeneous flows.
- Competition of up to two NewReno flows and up to two HD or CHD flows.

Tests are conducted with Gigabit Ethernet connected through a 10 Mbps bottleneck link created using dummynet [23], with a base round trip time of 40 ms (20 ms in each direction). The bottleneck queue is 84 packets long, corresponding to a maximum queueing delay of about 100 ms with 1500 byte packets.

TCP traffic is generated using Netperf [24]. NewReno flows use the default parameters. HD and CHD both use $q_{\min} = 5$ ms and $q_{\text{th}} = 30$ ms. $q_{\max}$ is dynamic, equal to the connections maximum observed queueing delay (as inferred by RTT measurements). For HD, the per packet $p_{\max} = 0.02$. For CHD, which makes decisions once per RTT, a $p_{\max} = 0.25$ gives comparable queueing delay backoff dynamics.

[4]This concept is also different to Compound TCP [14]. They supplement their loss based window with a delay based window to allow for a faster growth in `cwnd` if path delay characteristics allow.

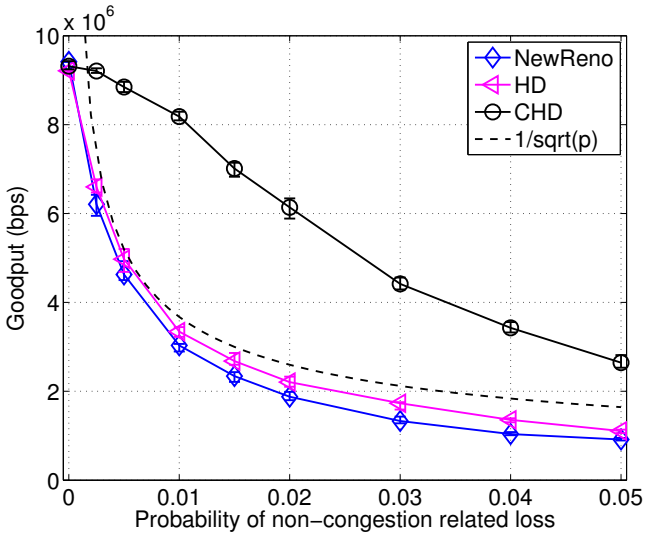


Fig. 4. Comparison of the goodput of NewReno, HD, and CHD when there are non-congestion related losses

The non-congestion related loss is implemented as random packet loss introduced by dummynet in the the forward (data) path. No random loss is applied to the return (ACK) path. Each experiment is repeated 10 times. Where appropriate, graphs show the 20th, 50th, and 80th percentiles (marker at the median, and error bars spanning the 20th to 80th percentiles).

As neither HD nor CHD require changes to the receivers, the "sinks" in Figure 3 are standard FreeBSD hosts.

A. Tolerance to non-congestion related losses

Our first experiment looks at how non-congestion related losses effect the goodput[5] of a TCP session. Each CC algorithms transmits separately for 60 s. Figure 4 shows the average goodput over 60 s plotted against the probability of non-congestion related packet loss.

NewReno and HD perform similarly over the range of loss probabilities. For comparison we also plot the theoretical maximum throughput under loss conditions given by the 1/sqrt(p) model first proposed by Mathis et al. [25]:

$$B = \frac{\text{pkt\_size}}{\text{rtt}} \frac{C}{\sqrt{p}} \tag{5}$$

where B is the expected throughput, rtt $= 40$ ms, p is the probability of packet loss, and $C = \sqrt{\frac{3}{2}}$.

CHD performs substantially better, maintaining a high goodput on losses of up to 1 %, and much better goodput than either HD and NewReno at even higher loss probabilities. This is a direct consequence of CHD reacting differently to loss while operating in Figure 1's region A versus region B. If a

[5]Defined as usable data transferred per unit time, excluding the payloads of retransmitted packets

loss occurs in region A, CHD recovers the lost packet but does not reduce `cwnd`. And since CHD's operation keeps queueing delays low, it is almost always operating within region A.

CHD's reaction to non-congestion related losses is deliberately conservative, in that it does not increase its window during the RTT that a lost packet is recovered. As a result goodput drops at higher loss probabilities because CHD spends more time retransmitting without getting much chance to grow `cwnd`. (CHD does not allow `cwnd` to increase during the recovery process, even though it would make CHD even more robust in high loss environments. If such a modified CHD incorrectly decided recent losses were unrelated to congestion, increasing `cwnd` would exacerbate whatever congestion caused the recent losses.)

B. Homogeneous Sharing Dynamics

Now we explore how each CC algorithm shares bandwidth with other instances of its own 'type'. Each experiment involves three 60 s flows using the same CC algorithm and sharing the bottleneck link, with the first, second and third flows starting at 0 s, 20 s and 40 s respectively.

Figure 5 shows the capacity sharing dynamics when there is no non-congestion related packet loss. Each goodput point is the average of 4 s of data transfer, plotted at the middle of each 4 s period. All CC algorithms share quite fairly among themselves.

First looking at NewReno (Figure 5(a)), the first flow starts sending with slow start, increasing `cwnd` until the queue overflows. At this point there is a 140 ms RTT (40 ms base RTT plus 100 ms queueing delay). As a result NewReno continues to increase its send rate until it detects the loss 140 ms later. This results in a large number of lost packets, NewReno backing off, and the lower initial goodput. Another thing to notice, is the comparatively slower convergence to a fair rate. This is due to the queueing delay being much longer for NewReno than for the delay based CC algorithms, resulting in a feeback loop (RTT) of up to 140 ms, compared with less than 70 ms (40 ms + q_{th}) in this scenario.

Both HD and CHD share the available capacity among themselves well. Since the flows begin to back-off once q_i or h_r is above $q_{\min}$, keeping the delays within Figure 1's region A, the average RTT is much smaller than in the NewReno case. Consequently, the sharing converges much faster.

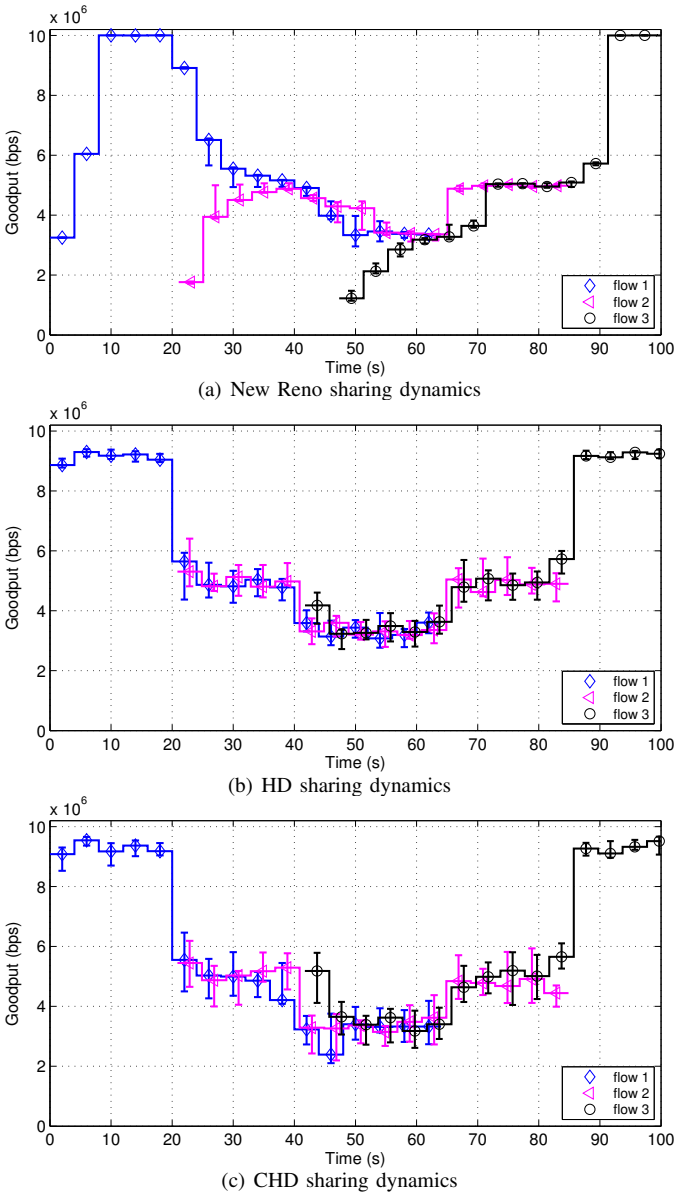
Figure 6 illustrates the same dynamic sharing when there is a probability of non-congestion related loss of 0.01. Both NewReno and HD are unable to utilise the available capacity, their send rates limited by the non-congestion related losses. CHD manages to use more of the available capacity due to its better tolerance to non-congestion related losses, while still sharing the available capacity evenly among the competing CHD flows.

C. Competing with NewReno

Finally we evaluate how HD or CHD flows compete against NewReno for available capacity. Each experiment consists of four flows transmitting for 80 s each, and starting up at 20 s intervals. The flows start in this order:

1) delay based CC (HD or CHD)
2) loss based CC (NewReno)
3) loss based CC (NewReno)
4) delay based CC (HD or CHD)

Figure 7 shows how HD and CHD coexist respectively with NewReno in a lightly multiplexed environment. Figure 7(a) shows that HD does not compete very well with NewReno

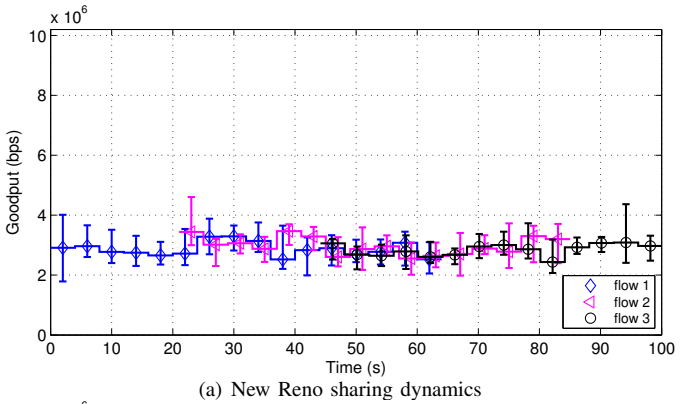


(a) New Reno sharing dynamics

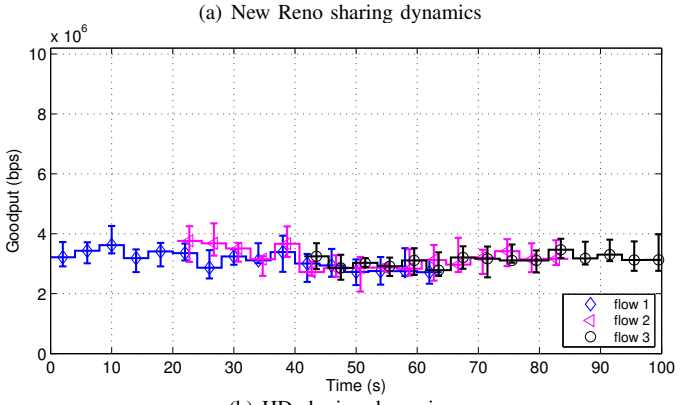
(b) HD sharing dynamics

(c) CHD sharing dynamics

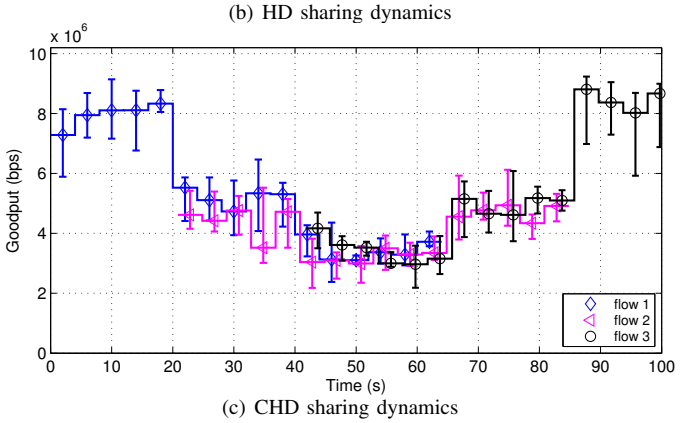
Fig. 5. Homogeneous capacity sharing dynamics between three flows with no non-congestion related losses. Each flow transmits for 60 s, starting at 20 s intervals. Points are 4 s averages, with the point at the beginning of the 4 s interval.



(a) New Reno sharing dynamics



(b) HD sharing dynamics



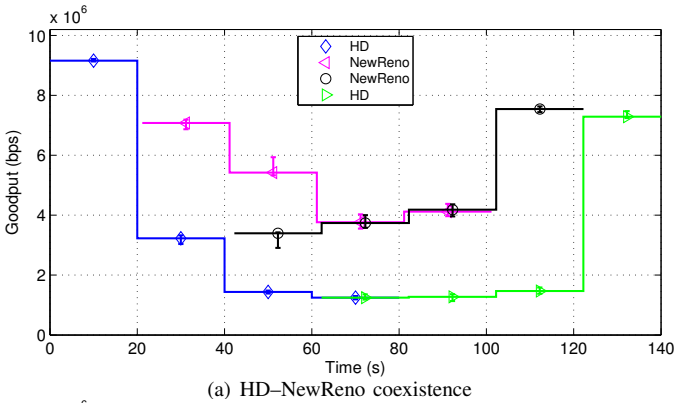
(c) CHD sharing dynamics

Fig. 6. Homogeneous capacity sharing dynamics between three flows with a 1 % random probability of non-congestion related losses. Each flow transmits for 60 s, starting at 20 s intervals. Points are 4 s averages, with the point in the middle of the 4 s interval.

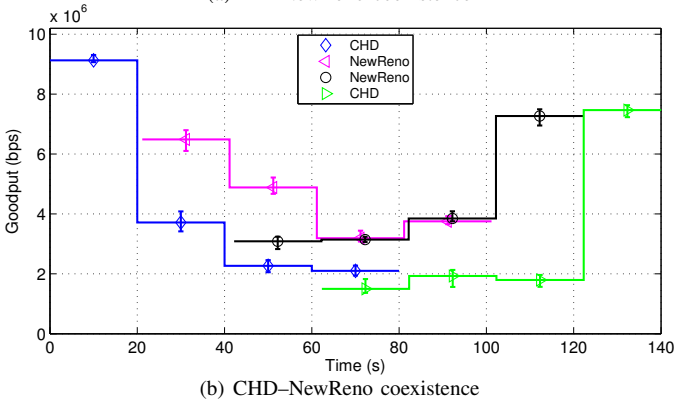
here[6]. Both the NewReno and HD flows compete well among themselves, but not with each other. Real traffic flows do not behave as fluid models, so the dynamics of the measured queue size results in HD reacting to a significant number of delay based congestion indications as wells as the loss based indications, halving `cwnd` for both cases.

Figure 7(b) shows the benefit of CHD keeping a NewReno-like shadow window. Although CHD still backs off to delay based congestion indications in Figure 1's region B (a necessity so that CHD will move back from region B to region

[6]Budzisz et al. [1] show that in NS2 simulations of highly multiplexed environments HD is able to coexist with NewReno relatively well, though still at a disadvantage. However, most home internet scenarios will be lightly multiplexed



(a) HD–NewReno coexistence



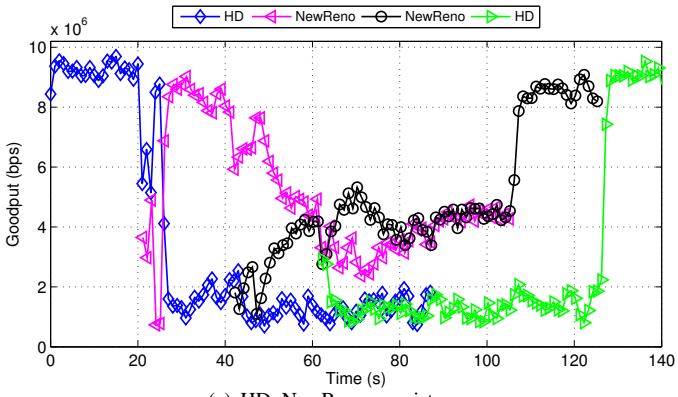
(b) CHD–NewReno coexistence

Fig. 7. Coexistence between loss and delay based CC algorithms with no non-congestion related losses. Points show 20 s averaged goodput (plotted point is in the middle of the averaging period).

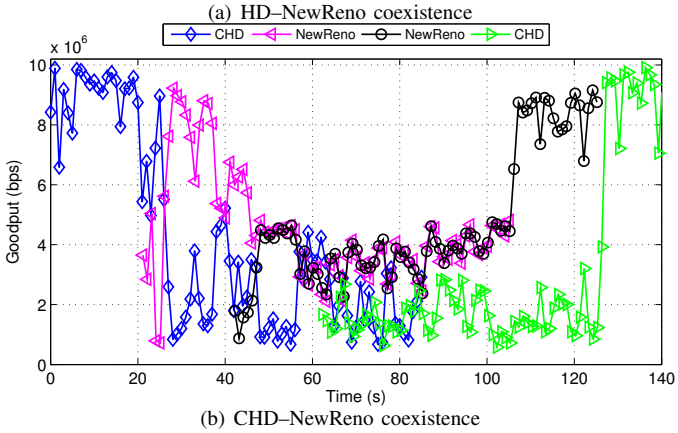
A once it is no longer competing with loss based flows), CHD regains some lost transmission capability if it detects a loss while in region B. NewReno still has a higher share of the available capacity than CHD, however CHD's use of the shadow window ensures it gains a fairer share than HD can. Once the NewReno flows have finished, the CHD flows quickly utilise the available capacity.

A single run of the same experiment is shown in Figure 8 at 1 s averages. This shows the dynamics of the loss–delay based congestion control interaction. In both cases when the first NewReno flow starts at about 20 s, it quickly captures most of the available capacity. HD is unable to reclaim much of its share of the capacity when congestion related losses occur (Figure 8(a)). However, as the NewReno flow fills the queue, and generates packet loss for itself and the CHD flow, it backs off while the CHD flow reclaims some of the capacity it lost to NewReno (Figure 8(b)). This activity continues as the next NewReno flow starts at 40 s and the final CHD flow starts at 60 s.

When non-congestion related packet loss is introduced, the dynamics change. Figure 9 shows the 20 s averaged goodput plots for HD and CHD coexisting with NewReno when there is a 1 % non-congestion related probability of loss. Both HD and NewReno are unable to fully utilise the available capacity in this scenario, but share fairly (see Figure 9(a)). CHD however is better able to use the available capacity in the presence of non-congestion related loss. It captures more of the available

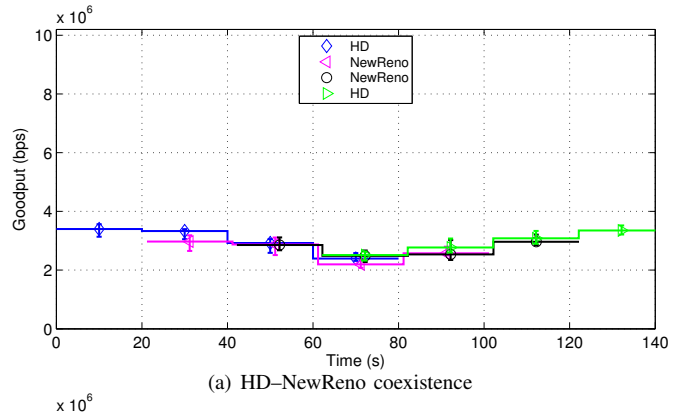


(a) HD–NewReno coexistence

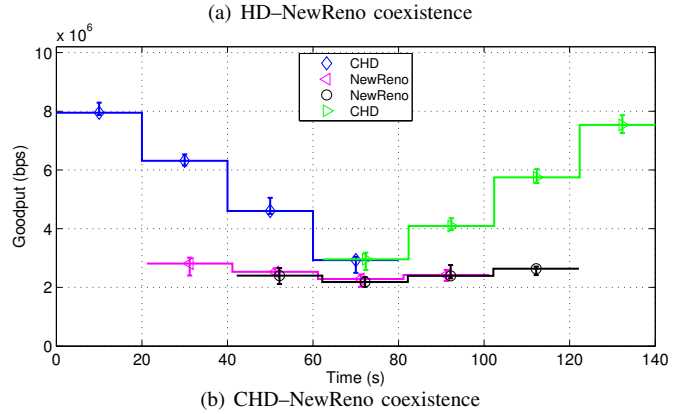


(b) CHD–NewReno coexistence

Fig. 8. Coexistence between loss and delay based CC algorithms with no non-congestion related losses. Points show 1 s averaged goodput of a single experiment run.



(a) HD–NewReno coexistence



(b) CHD–NewReno coexistence

Fig. 9. Coexistence between loss and delay based CC algorithms with a 1 % probability of non-ongestion related loss. Points show 20 s averaged goodput (plotted point is in the middle of the averaging period).

capacity than NewReno is able to, but not at the expense of NewReno, rather it is capacity that NewReno is unable to use.

VI. Further work

A key issue for both CHD and HD is setting the various thresholds. This paper attempts to show HD and CHD at their best for the given scenario. However, for mass deployment it will be necessary for CHD to be able to cope with the Internet's widely varying delay characteristics and automatically choose appropriate values of q_{th} and possibly $p_{\max}$.

Since delay based CC reacts to congestion much earlier than loss based CC, it seems reasonable for them reduce `cwnd` by less than half on delay based congestion triggers. This may have fairness and convergence ramifications, so needs further investigation.

An intrinsic problem with delay based congestion indications, is that they rely on an accurate estimate of the base RTT in order to accurately infer congestion. Differences in the base RTT estimate among competing sources can cause unfairness in the way they share available capacity.

CHD reacts conservatively to what it infers to be non-congestion related loss, by holding its congestion window at the pre-loss value until the lost packet has been recovered. This conservative approach is fine for low loss scenarios, but prevents CHD from making use of the available capacity. If a less conservative approach was adopted that allowed the congestion window to grow during the recovery process, even by the single packet additive increase, CHD would be able to operate in much higher loss environments.

VII. Conclusion

We have proposed and demonstrated CHD, a significant enhancement to a delay-based TCP algorithm first proposed by Budzisz et al. [1] (HD). CHD is a sender-side algorithm that requires no changes to existing TCP receivers. Relative to HD, CHD provides improved tolerance to non-congestion related packet losses and improved coexistence with loss based TCP. This is achieved by performing delay based cwnd operations only once per RTT, and introducing a *shadow window* that mimics NewReno-like behaviour to allow better responsiveness when competing with loss based flows.

Using actual implementations of CHD, HD and NewReno under FreeBSD 9.0 we have examined two issues:

- The effect of low levels of non-congestion related packet loss (such as in wireless link technologies) on TCP congestion control;
- The coexistence of delay and loss based TCP congestion control algorithms on lightly multiplexed bottleneck links (such as home internet access scenarios).

CHD was found to provide good non-congestion related loss tolerance with a single flow achieving 82 % of the available capacity at 1 % packet loss, compared to 33 % and 30 % for HD and NewReno respectively (a 150 % improvement). CHD's concept of a shadow window helps it to recover from extra delay based congestion back-offs when competing with loss based TCP algorithms such as NewReno, with

improvements of over 50 % over those achieved by HD in the lightly multiplexed scenarios.

ACKNOWLEDGMENT

The development of delay based congestion control for FreeBSD is part of the newTCP project and was made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley.

REFERENCES

[1] L. Budzisz, R. Stanojevic, R. Shorten, and F. Baker, "A strategy for fair coexistence of loss and delay-based congestion control algorithms," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 555–557, Jul. 2009.

[2] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Sep. 1981, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[3] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of internet traffic in 1998-2003," in *Winter International Symposium on Information and Communication Technologies (WISICT)*, Cancun, Mexico, Jan. 2004. [Online]. Available: http://www.caida.org/publications/papers/2003/nlanr/nlanr_overview.pdf

[4] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782 (Proposed Standard), Apr. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3782.txt

[5] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control ," RFC 2581 (Proposed Standard), Apr. 1999, updated by RFC 3390. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

[6] S. Floyd, "Congestion Control Principles," RFC 2914 (Best Current Practice), Sep. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2914.txt

[7] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 5, pp. 56–71, 1989.

[8] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 9–16, Apr. 1992.

[9] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[10] C. Jin, D. Wei, and S. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, Mar. 2004, pp. 2490–2501.

[11] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006.

[12] A. Kuzmanovic and E. Knightly, "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 739–752, Aug. 2006.

[13] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP," in *IEEE INFOCOM 2005*, 2005, pp. 1838–1848.

[14] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, Apr. 2006, pp. 1–12.

[15] S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov, "Emulating AQM from end hosts," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2007, pp. 349–360.

[16] D. Leith, R.Shorten, G.McCullagh, J.Heffner, L.Dunn, and F.Baker, "Delay-based AIMD congestion control," in *Proc. Protocols for Fast Long Distance Networks*, California, 2007.

[17] J. Martin, A. Nilsson, and I. Rhee, "Delay-based congestion avoidance for tcp," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 356–369, Jun. 2003.

[18] G. McCullagh and D. J. Leith, "Delay-based congestion control: Sampling and correlation issues revisited," Hamilton Institute – - National University of Ireland, Maynooth, Tech. Rep., 2008.

[19] K. Kotla and A. Reddy, "Making a delay-based protocol adaptive to heterogeneous environments," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, Jun. 2008, pp. 100–109.

[20] D. Hayes, "Timing enhancements to the FreeBSD kernel to support delay and rate based TCP mechanisms," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 100219A, 19 February 2010. [Online]. Available: http://caia.swin.edu.au/reports/100219A/CAIA-TR-100219A.pdf

[21] "NewTCP project tools," [Accessed 26 April 2010]. [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools.html

[22] L. Stewart and J. Healy, "Light-weight modular TCP congestion control for FreeBSD 7," CAIA, Tech. Rep. 071218A, Dec. 2007. [Online]. Available: http://caia.swin.edu.au/reports/070717B/CAIA-TR-070717B.pdf

[23] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[24] R. Jones, "Netperf homepage," [Accessed 26 April 2010]. [Online]. Available: http://www.netperf.org/

[25] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997.