Collateral Damage: The Impact of Optimised TCP Variants on Real-Time Traffic Latency in Consumer Broadband Environments

Lawrence Stewart, Grenville Armitage, and Alana Huebner

Swinburne University of Technology
Melbourne, Australia
{lastewart,garmitage}@swin.edu.au, alanahuebner@gmail.com

Abstract. In recent years a number of TCP variants have emerged to optimise some aspect of data transport where high delay-bandwidth product paths are common. We evaluate a different scenario - latency-sensitive UDP-based traffic sharing a consumer-grade 'broadband' link with one or more TCP flows. In particular we compare Linux implementations of NewReno, H-TCP and CUBIC. We find that dynamic latency fluctuations induced by each TCP variant is a more significant differentiator than 'goodput' (useful throughput), and that CUBIC induces far more latency than either H-TCP or NewReno when multiple TCP flows are active concurrently. This potential for 'collateral damage' should influence future efforts to re-design TCP for widespread deployment.

Keywords: TCP, congestion control, latency, interactive, broadband.

1 Introduction

Transmission control protocol (TCP) [1] deserves significant credit for the internet's wide-spread utility over the past 25+ years. The relatively modern *NewReno* variant of TCP [2] 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 performance [3]. As the dominant transport protocol for internet-based applications [4], maximisation of TCP performance has been an active and challenging area for academic and industry research into congestion control (CC) techniques [5]. A common focus has been on the interactions between multiple TCP sessions when sharing a common congestion point or path segment, particularly on long paths with high link speeds.

This paper looks at an increasingly important new scenario - TCP flows sharing consumer-grade 'broadband' links with non-reactive, latency-sensitive UDP-based applications such as Voice over IP (VoIP) and online games. In particular we focus on the impact of the Linux implementations of NewReno, *CUBIC* [6,7] (currently the default CC algorithm for Linux TCP connections), and *H-TCP* [8,9] variants of TCP. Each variant provides different dynamic response to congestion within an IP network, which has a direct impact on the

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 392–403, 2009.

latency experienced by other flows sharing a congestion point. We characterise the extent to which both CUBIC and H-TCP induce greater median latency than NewReno yet provide little nett gain in *goodput* (useful throughput as measured by the application on top of TCP).

Section 2 begins with an over-view of issues surrounding TCP CC research. Section 3 describes our testbed, instrumented to allow precise tracking of latency versus TCP's congestion window size and congestion events. In section 4 we illustrate the range of latencies, and frequency of latency fluctuations, experienced by unrelated UDP flows sharing a congestion point with long-lived NewReno, CUBIC and H-TCP flows.

2 Background

The evolution of the internet's IP-based network and underlying infrastructure has exceeded initial architectural design assumptions and expectations in a number of areas. Since congestion control (CC) was first proposed [10] and subsequently mandated [11], there has been significant ongoing research to ensure CC kept pace with the underlying network it was designed to efficiently utilise and protect from congestion collapse.

The Internet Research Task Force's (IRTF) Internet Congestion Control Research Group (ICCRG) [12] and Transport Modeling Research Group (TMRG) [13] have been established to shepherd the development, evaluation and (where applicable) standardisation of improved and altogether new CC mechanisms and schemes, with a focus on transport protocols.

A particularly large body of work has developed around improvements to the Additive Increase Multiplicative Decrease (AIMD) congestion window (cwnd) scaling factors and congestion detection mechanisms used by the defacto-standard NewReno CC algorithm. Wireless environments (where packet loss is not indicative of congestion) and large bandwidth-delay product (BDP) paths (which take multiple minutes to re-probe network capacity after congestion back-off) are some of NewReno's current problem areas where it is failing to meet the goal of efficient network utilisation.

A raft of new protocols have been proposed in recent years, which typically aim to solve a weakness in NewReno under a specific subset of network scenarios. The resulting proposals are being discussed, evaluated and refined within the context of the ICCRG and TMRG, with an eye to eventual publication as an experimental or fully fledged Internet Engineering Task Force (IETF) standard.

Evaluation of TCP related CC mechanisms is itself an active area of research. Steps are being taken to develop a set of public baseline test scenarios and metrics with which to compare new CC algorithms [14]. Evaluation by individual algorithm implementers thus far has largely focused on aspects such as intra-protocol fairness, inter-protocol fairness with NewReno, throughput and convergence.

While there are good reasons to maintain multiple CC implementations for research purposes and specific application use cases, the need for a suitable default for the average network stack is important. It is therefore crucial to test

real-world implementations of proposed protocols in common network scenarios that a standardised protocol would likely be deployed in.

To date, we have found very little prior work [15,16] investigating home broadband scenarios and the behaviour of emerging TCPs, or TCPs interacting with non-congestion reactive, latency-sensitive traffic in this environment. The continuing convergence towards IP based entertainment, information access and communication service delivery ensures this is an area of increasing importance.

Gaming and voice over IP are two such services of interest, both typically delivered using constant rate, non-congestion reactive flows of small UDP packets. Online multiplayer computer games, and the popular first person shooter genre in particular, have well known latency sensitivity requirements for effective game play [17]. Voice over IP is another real-time service with well characterised latency tolerances, previously examined in the context of traditional telephony [18,19] and now IP networks more recently [20]. The requirements of these increasingly important applications must be taken into consideration within the context of transport protocol research and development.

3 Experimental Methodology

Figure 1 shows our experimental topology - one congestion point (a FreeBSD router[1] with configurable forwarding latency and instrumented to log actual queue utilisation over time), four Debian Linux hosts[2] acting as TCP or UDP sources and sinks, and a precision traffic capture tool[3] to calculate one way delay (OWD) through the router. Our simple *dumbbell* testbed is not topologically equivalent to the actual network paths traversed by typical consumer traffic. Nevertheless it is suitable for this paper's focus on the interactions between TCP and UDP flows.

As consumer broadband links vary widely around the world we settled on emulating 1Mbps links with drop-tail queues of 60000 bytes in each direction (based on previously published estimations of buffering in consumer routers [21,22]). The drop-tail queues create the bottleneck shared by all traffic traversing the router, and $RTT/2$ of delay is added in each direction using *dummynet* [23]. We experimented with total RTTs of 50ms and 100ms to emulate delays conceivably experienced by typical consumer activities. Configured latency is accurate to within 0.5ms as the router's kernel was set to tick at 2000Hz ($kern.hz = 2000$).

Hosts A, B, C and D are instrumented with Web100 [24,25] - tracking TCP connection parameters (such as *cwnd*) by polling every 1ms over the lifetimes of active TCP sessions.

[1] FreeBSD 7.0-RC1 on a 2.80GHz Intel Celeron D (256K L2 Cache), 512MB PC3200 DDR-400 RAM, with two Intel PRO/1000 GT 82541PI PCI gigabit Ethernet cards as forwarding interfaces.

[2] A 2.6.25 kernel ticking at 1000Hz, each one a 1.86GHz Intel Core2 Duo E6320 (4MB L2 Cache) CPU, 1GB PC5300 DDR2 RAM and Intel PRO/1000 GT 82541PI PCI gigabit Ethernet NIC.

[3] Two Endace DAG 3.7GF gigabit ethernet capture card ports.

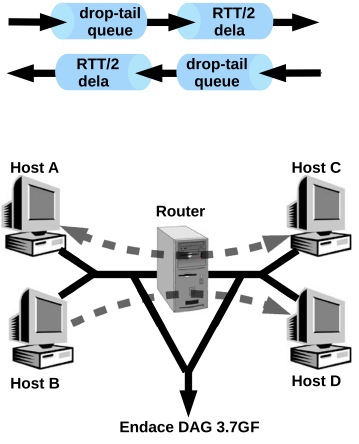


Fig. 1. Testbed for measurement of shared, congestion-induced queuing delays

Bulk TCP traffic was generated using Iperf [26], with data flowing from Host A to C and ACKs from Host C to A. Uni-directional UDP traffic from Host B to D (186byte IP packets every 20ms, emulating non-reactive VoIP traffic) was generated using Tcpreplay [27]. No traffic flowed in the reverse direction from Host D to B.

Load-time tunable variables of the Linux CUBIC and H-TCP implementations were left at their default values, except as noted here. For CUBIC, we set "max increment" to 100[4]. To ensure H-TCP's consistency with the published internet draft [8] we disabled "adaptive backoff" and "adaptive reset", but left "RTT scaling" on[5].

Trials were run five times for each combination of TCP algorithm, testbed RTT and number of TCP flows. Trials lasted for at least three minutes or twenty congestion epochs, which ever was longer. For each group of five runs we discarded the highest and lowest results and took the average of the remaining three.

4 Results

Our initial experiments focus on measuring the following characteristics: TCP goodput, latency as experienced by the UDP flow over time, the frequency of congestion events as observed in the router's bottleneck queue and the number of packets retransmitted by TCP.

[4] This variable has been removed in more recent CUBIC specifications and setting it to 100 effectively disabled any effect it might have in our test scenarios.

[5] The H-TCP code also ran with a patch functionally equivalent to [28] that fixed a visible bug in H-TCP's behaviour.

4.1 Goodput Achieved by NewReno, CUBIC and H-TCP

Goodput for each trial run was calculated using tcptrace [29], providing a measure of the application level (or "usable") bitrate achieved during each trial. All three TCP variants exhibited much the same average goodput when RTT was both 50ms and 100ms. Table 1 lists the average aggregate goodput across all flows.

Table 1. Aggregate 'goodput' of one, two and five concurrent flows - NewReno, H-TCP or CUBIC congestion control, 50ms or 100ms RTT, 1Mbit/sec bottleneck

RTT (ms)	Algorithm	No. Flows					
		1		2		5	
		Goodput Statistics (KB/s)[a]					
		μ	σ	μ	σ	μ	σ
50	NewReno	111.8	0.01	112.2	0.19	112.2	0.10
	H-TCP	111.9	0.06	112.5	0.11	114.2	0.50
	CUBIC	111.9	0.04	112.6	0.24	114.5	0.39
100	NewReno	111.9	0.01	112.2	0.01	112.5	0.23
	H-TCP	111.8	0.04	112.7	0.19	114.0	0.43
	CUBIC	111.9	0.04	112.7	0.25	114.3	0.39

[a] Where 1KB = 1000 bytes.

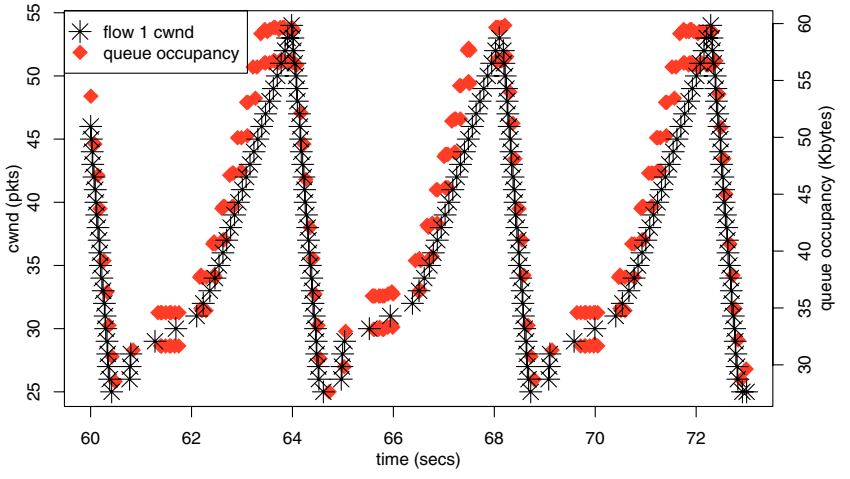
Consumers constrained by 1Mbit/sec links and 50ms or 100ms paths were not the original design target for TCP variants such as CUBIC and H-TCP. Indeed, H-TCP and CUBIC were intended to be NewReno-like over links with low bandwidth delay product (BDP), so our goodput results are not unexpected.

However, our experiments reveal that H-TCP and CUBIC diverge from NewReno in significant ways when we consider the latency caused by each TCP variant's congestion control behaviour in low BDP environments.

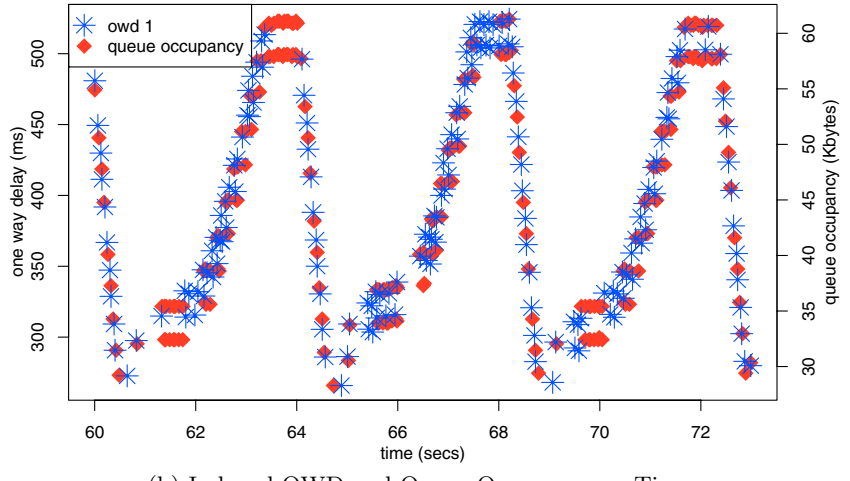
4.2 Latency Induced by NewReno, CUBIC and H-TCP

Latency-sensitive applications such as VoIP and online games typically generate UDP flows that have simplistic (or non-existent) reaction to congestion within the underlying IP network path. Such applications are increasingly important in the consumer market, and TCP's cyclical variation of *cwnd* is known to cause bottleneck queuing delays to fluctuate regularly. Our experiments suggest that this 'induced latency' should be considered an important differentiator between TCP variants that expect deployment in consumer contexts.

Illustrative examples of the cyclical variation in cwnd, one way delay (OWD) and queue length induced by TCP are provided by Figures 2a and 2b. These figures clearly highlight the relationship between cwnd, queue occupancy and induced OWD during three consecutive congestion epochs of a single H-TCP flow sharing the bottleneck link with a single UDP flow over a 100ms RTT path.



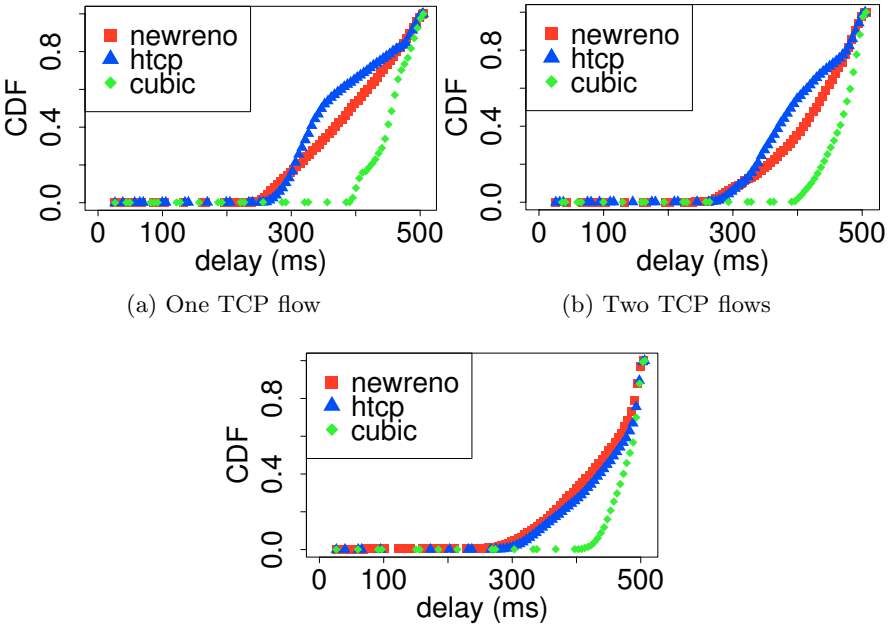
(a) Cwnd and Queue Occupancy vs Time



(b) Induced OWD and Queue Occupancy vs Time

Fig. 2. Cwnd, Induced OWD and Queue Occupancy vs Time across three consecutive congestion epochs of a single H-TCP flow. Path is 1Mbit/sec @ 100ms RTT.

More usefully, Figures 3a, 3b and 3c show the CDF of OWD experienced by our single UDP flow over at least twenty congestion epochs when the bottleneck link is shared with one, two and five TCP flows respectively. Plots for 100ms looked identical. In this case the path has a 50ms baseline RTT (25ms OWD), thereby fixing the best case OWD at 25ms, which is visible in the figures. The upper bound is the maximum queuing delay (480ms, being 60000 bytes at 1Mbps) plus the path's intrinsic 25ms OWD. A statistically insignificant number of UDP packets were dropped during queue full events, and these are excluded from the latency statistics presented here. NewReno, H-TCP and CUBIC clearly interact very differently



(a) One TCP flow

(b) Two TCP flows

(c) Five TCP flows

Fig. 3. Cumulative distribution of OWD for NewReno, H-TCP and CUBIC. Path is 1Mbit/sec @ 50ms RTT.

with the bottleneck queue over time - behaviours directly attributable to their particular algorithms for managing the TCP congestion window.

H-TCP mimics NewReno cwnd growth for the first second after congestion, and then follows a parabolic growth function proportional to $x^2/2$ until the next congestion event (where x is related to the time since last congestion).

CUBIC follows a cubic growth function proportional to $0.4x^3$ (where x is also related to the time since last congestion). Unlike H-TCP, CUBIC takes NewReno cwnd growth as a baseline. The algorithm switches to the NewReno growth function if the cubic growth function calculates a value less than NewReno would have achieved in the same amount of time since congestion.

The difference between the growth functions themselves is also pertinent. Unlike parabolas, cubic functions grow quickly in their concave region, gradually slowing as they reach an inflection point before switching to convex mode operation and grow quickly again. This behaviour is particularly noticeable in Figure 3a. The CUBIC data shows a steep increase in latency at 400ms (concave growth), which slows between 440ms-480ms (near inflection point) before starting to grow again (convex growth).

As more TCP flows are added (Figures 3b and 3c) the CUBIC flows continue to induce significant additional delay whereas the H-TCP flows tend to induce NewReno-like latency. (More concurrent flows create more frequent congestion

Table 2. Latency induced by NewReno, H-TCP or CUBIC congestion control on one, two or five concurrent flows, 50ms or 100ms RTT, 1Mbit/sec bottleneck

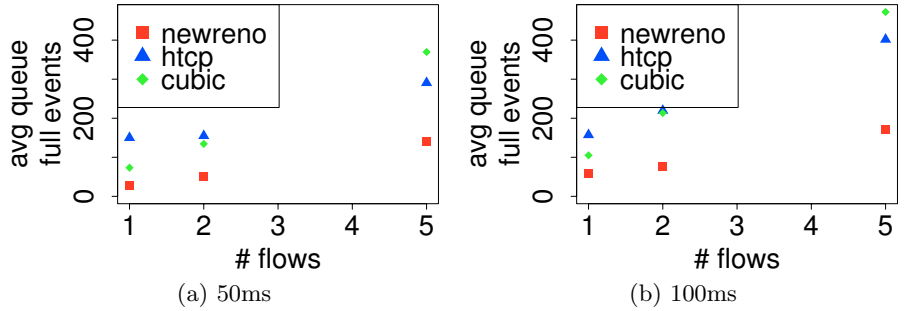
RTT (ms)	Algorithm	No. Flows								
		1			2			5		
		Induced Latency Statistics (ms)								
		Median	μ	σ	Median	μ	σ	Median	μ	σ
50	NewReno	393.2	387.7	74.2	422.8	410.1	67.1	445.5	427.3	68.2
	H-TCP	347.0	371.1	73.5	390.8	397.8	66.3	459.6	439.2	59.7
	CUBIC	456.2	452.2	35.5	474.1	465.7	31.9	485.0	476.4	26.1
100	NewReno	409.8	402.4	81.0	442.4	427.6	73.7	466.0	447.1	74.1
	H-TCP	353.3	383.6	85.1	405.5	414.8	71.5	479.0	459.1	62.9
	CUBIC	487.8	476.7	42.0	494.5	485.2	37.8	504.3	494.7	32.1

events, and the individual H-TCP flows spend proportionally more time exhibiting NewReno-like cwnd growth.) Table 2 quantifies the median, mean and standard deviation of latencies induced during the separate trials of NewReno, H-TCP and CUBIC congestion control with one, two or five concurrent flows.

4.3 Impact of Congestion Events

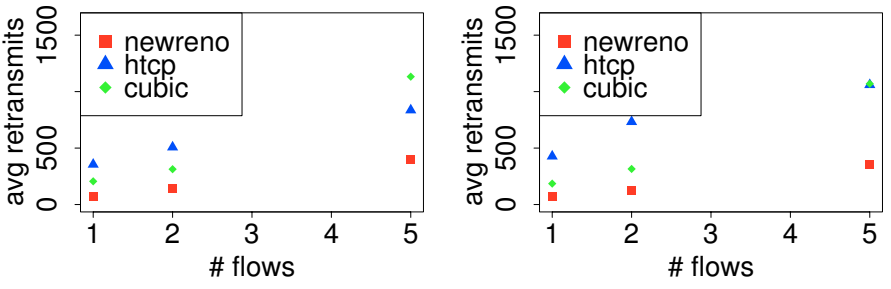
Figure 4 plots the average number of queue full events per minute, observed during a sixty second period in the middle of each trial. A queue full event occurs when the dummynet bottleneck queue reaches capacity and drops packets until it is able to accept a new packet.

In the one and two TCP flow cases, H-TCP's parabolic cwnd growth function causes it to consistently overshoot the queue capacity by some margin, resulting in the larger number of queue full events. However for the five flow case, H-TCP's tendency to remain in NewReno mode due to the increased frequency of congestion events results in fewer queue full events. CUBIC on the other hand



(a) 50ms (b) 100ms

Fig. 4. Average number of "queue full" events per minute



(a) 50ms (b) 100ms

Fig. 5. Average number of retransmitted packets

demonstrates no such restraint, which results in faster increases in queue full events per flow than H-TCP or NewReno.

Retransmitting packets due to network and receiver losses should ideally be kept to a minimum to ensure efficient network resource utilisation. In the low BDP bottleneck scenarios under investigation, retransmissions are almost entirely caused by a flow's cwnd overshooting the path capacity and subsequently overflowing the bottleneck queue. A TCP's cwnd growth function therefore plays an integral part in balancing protocol scalability (quickly probing the path to utilise available bandwidth) and stability (minimising the time a connection is not sending user data).

Figures 5a and 5b plot the average aggregate number of retransmissions for the 50ms and 100ms trial sets respectively[6]. Because they are optimised to aggressively probe for network capacity for benefit in high BDP environments, H-TCP and CUBIC both trigger significantly more retransmissions than NewReno.

These results also align with those of Figure 4, as we would expect queue full events and retransmissions due to lost packets to be closely correlated.

5 Conclusion and Further Work

We have performed a comparison of the Linux implementations of NewReno, H-TCP and CUBIC TCP over an emulated path that (roughly) approximates a congested consumer broadband link. First we observed that 'goodput' (useful throughput) was essentially equivalent for each TCP variant when pushing data over a 50ms or 100ms RTT path with a 1Mbps bottleneck link speed and 60000 byte queue. Then we explored the latency that would likely be experienced by VoIP-like UDP traffic sharing such a congested 'consumer' link with each TCP variant.

CUBIC was observed to induce noticeably more latency than either H-TCP or NewReno when one or more active TCP flows congest a link. With one or two concurrent flows, it is even possible for H-TCP to induce less latency than

[6] Calculated using tcptrace and summed for each test run.

both CUBIC and NewReno. For example, two TCP flows sharing a 50ms RTT path with a 60000 byte queue at a 1Mbps bottleneck induced median latencies of 422.8ms, 390.8ms and 474.1ms for NewReno, H-TCP and CUBIC respectively. With five concurrent flows the median induced latencies rose to 445.5ms, 459.6ms and 485.0ms respectively.

The impact of induced latency on VoIP and online game applications is a form of 'collateral damage'. We believe future efforts to re-design TCP for widespread deployment should aim to minimise this impact, rather than aiming for fairness between TCP flows or maximisation of goodput in long, fast networks. Additionally, the largest difference between the latency induced by the optimised variants stems from differences in their "low speed" compatibility modes. Algorithm developers should consider these findings in the future refinement of compatibility modes, as there are likely gains to be had by defining better behaving options for broadband-like environments.

The current behaviour of the tested common TCP variants in broadband environments strongly indicates that CPE manufacturers need to provide better mechanisms to appropriately manage their device's queues. The current trend towards large unmanaged (by default) queues in CPE is demonstrably bad news for consumers.

Our analysis suggests a number of areas for future work. In order to rule out implementation effects, a similar suite of tests should be performed using independent implementations of NewReno, H-TCP and CUBIC (ideally under a different operating system, such as FreeBSD). Extending the evaluation to a wider set of TCP variants and further exploring useful metrics to differentiate the impact of TCPs in the home broadband environment is required. Finally, exploring simple ways in which CPE manufacturers can easily address the issues raised by this research would result in some material benefit to consumers. For example, evaluating the appropriate tuning and impact of various queue management schemes with a goal of making recommendations to CPE manufacturers would be a worthwhile outcome.

Acknowledgments

This work has been made possible in part by a grant from the Cisco University Research Program Fund at Community Foundation Silicon Valley.

References

1. Postel, J.: Transmission Control Protocol. RFC 793, Standard, Updated by RFC 3168. (September 1981), http://www.ietf.org/rfc/rfc793.txt
2. Floyd, S., Henderson, T., Gurtov, A.: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard) (April 2004), http://www.ietf.org/rfc/rfc3782.txt
3. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. RFC 2581 (Proposed Standard), Updated by RFC 3390 (April 1999), http://www.ietf.org/rfc/rfc2581.txt

4. Fomenkov, M., Keys, K., Moore, D., Claffy, K.: Longitudinal study of Internet traffic in 1998-2003. In: Winter International Symposium on Information and Communication Technologies (WISICT), Cancun, Mexico (January 2004), http://www.caida.org/publications/papers/2003/nlanr/nlanr_overview.pdf
5. Floyd, S.: Congestion Control Principles. RFC 2914 (Best Current Practice) (September 2000), http://www.ietf.org/rfc/rfc2914.txt
6. Rhee, I., Xu, L., Ha, S.: CUBIC for Fast Long-Distance Networks. Technical report, North Carolina State University (August 2008), http://tools.ietf.org/id/draft-rhee-tcpm-cubic-02.txt
7. Ha, S., Rhee, I., Xu, L.: CUBIC: A New TCP-Friendly High-Speed TCP Variant. ACM SIGOPS Operating System Review 42(5), 64–74 (2008), http://netsrv.csc.ncsu.edu/export/cubic_a_new_tcp_2008.pdf
8. Leith, D.: H-TCP: TCP Congestion Control for High Bandwidth-Delay Product Paths. Technical report, Hamilton Institute (April 2008), http://tools.ietf.org/id/draft-leith-tcp-htcp-06.txt
9. Li, Y.T., Leith, D., Shorten, R.N.: Experimental evaluation of tcp protocols for high-speed networks. IEEE/ACM Trans. Netw. 15(5), 1109–1122 (2007), http://dx.doi.org/10.1109/TNET.2007.896240
10. Jacobson, V.: Congestion avoidance and control. In: SIGCOMM 1988: Symposium proceedings on Communications architectures and protocols, pp. 314–329. ACM, New York (1988), http://doi.acm.org/10.1145/52324.52356
11. Braden, R.: Requirements for Internet Hosts - Communication Layers. RFC 1122, Standard, Updated by RFC 1349 (October 1989), http://www.ietf.org/rfc/rfc1122.txt
12. Internet Research Task Force: (Internet Congestion Control Research Group), http://www.irtf.org/charter?gtype=rg&group=iccrg (accessed November 8, 2008),
13. Internet Research Task Force: (Transport Modeling Research Group), http://www.irtf.org/charter?gtype=rg&group=tmrg (accessed November 8, 2008)
14. Andrew, L., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., Rhee, I.: Towards a Common TCP Evauation Suite. In: Sixth International Workshop on Protocols for Fast Long-Distance Networks, Manchester, GB (March 2008), http://www.hep.man.ac.uk/PFLDnet2008/paper/08_Lachlan_pfldnet2008.pdf
15. Andrew, L., Atov, I., Kennedy, D., Wydrowski, B.: Evaluation of FAST TCP on Low-Speed DOCSIS-based Access Networks. In: IEEE TENCON 2005, Melbourne, Australia (November 2005), http://ieeexplore.ieee.org/iel5/4084859/4084860/04085219.pdf?tp=&arnumber=4085219&isnumber=4084860
16. Armitage, G., Stewart, L., Welzl, M., Healy, J.: An Independent H-TCP Implementation Under FreeBSD 7.0: Description and Observed Behaviour. SIGCOMM Comput. Commun. Rev. 38(3), 27–38 (2008), http://doi.acm.org/10.1145/1384609.1384613
17. Armitage, G.: An experimental estimation of latency sensitivity in multiplayer quake 3. In: 11th IEEE International Conference on Networks (ICON 2003), Sydney, Australia, pp. 137–141 (2003), http://dx.doi.org/10.1109/ICON.2003.1266180

18. Helder, G.K.: Customer evaluation of telephone circuits with transmission delay. Bell System Technical Journal 45, 1157–1191 (1966)
19. Kitawaki, N., Itoh, K.: Pure delay effects on speech quality in telecommunications. IEEE Journal on Selected Areas in Communications 9(4), 586–593 (1991)
20. Markopoulou, A., Tobagi, F., Karam, M.: Assessing the quality of voice communications over internet backbones. IEEE/ACM Transactions on Networking 11(5), 747–760 (2003)
21. Claypool, M., Kinicki, R., Li, M., Nichols, J., Wu, H.: Inferring Queue Sizes in Access Networks by Active Measurement. In: Passive and Active Measurement Workshop, Antibes Juan-les-Pins, France (April 2004), http://www.pamconf.org/2004/papers/209.pdf
22. Dischinger, M., Haeberlen, A., Gummadi, K.P., Saroiu, S.: Characterizing residential broadband networks. In: IMC 2007: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp. 43–56. ACM, New York (2007), http://doi.acm.org/10.1145/1298306.1298313
23. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review 27(1), 31–41 (1997), http://doi.acm.org/10.1145/251007.251012
24. Mathis, M., Heffner, J., Reddy, R.: Web100: extended tcp instrumentation for research, education and diagnosis. SIGCOMM Comput. Commun. Rev. 33(3), 69–79 (2003)
25. Unknown (The Web100 Project), http://web100.org/ (accessed November 19, 2008)
26. Unknown: (Iperf), http://sourceforge.net/projects/iperf (accessed November 19, 2008)
27. Turner, A.: Tcpreplay, http://tcpreplay.synfin.net/ (accessed December 4, 2008)
28. Leith, D.: [2.6.27] tcp_htcp: last_cong bug fix, http://patchwork.ozlabs.org/patch/8341/ (accessed December 4, 2008)
29. Ostermann, S.: tcptrace, http://www.tcptrace.org/ (accessed December 4, 2008)