Impact of insufficient TCP receive window on bulk data transfers in home broadband environments

Shahana Cumaranayagam, Djuro Mirkovic
Centre for Advanced Internet Architectures, Technical Report 160923A
Swinburne University of Technology
Melbourne, Australia
scumaranayagam@swin.edu.au, dmirkovic@swin.edu.au

***Abstract*—This report examines the way in which insufficient (or misconfigured) advertised receiver window can cause TCP connections to be capped or otherwise unable to utilise a sufficiently high bandwidth delay product path. We will explore two anomalous scenarios where the relationship between the advertised receiver window, bandwidth delay product, and bottleneck buffering can lead to failure of TCP connection to move from slow start mode to congestion avoidance mode, which in turn results in poor TCP performance. We demonstrate that such scenarios are rare, but when they exist, it is because of a large bandwidth delay product and high bottleneck buffering (higher than advertised receiver window of any single individual receiver window) which resulted in send window being capped at a certain rate. Our recommendations are to ensure that advertised receiver window is sufficiently sized to maximise utilisation of the path and to avoid excess buffering – which makes it difficult for the receiver (with their default advertised receiver window) to allow TCP connection to fully utilise the path.**

***Index Terms*—TCP, TEACUP, anomalous TCP behaviour, BDP, sender window, advertised receiver window, congestion window, bottleneck buffer, queuing delay, NewReno, PFIFO**

I. Introduction

The embodiment of Internet applications today have increased user reliance and expectation for optimal performance. Over 90% of the Internet traffic relies on TCP inter-connectivity [1], [2], and has elevated the interest on extensive performance understanding on how TCP can be significantly (and perhaps unexpectedly) sub-optimal under network conditions that are not unlikely in today's Internet [3], [4]. Whilst TCP immensely contributes towards network performance it also provides significant impact on the network resource utilisation. The scope of this report targets on the context of application performance experienced by end users and analyses two specific unusual, yet plausible scenarios where standard compliant TCP performance is not exhibited.

The remainder of this paper is organised as follows, Section II provides the background information of the report, Section III outlines the testbed and experimental scenario conditions, Section IV discusses unintended anomalous scenarios. Section V summarises our work and suggests how to avoid undesired TCP behaviour from the identified anomalous scenarios.

II. Background

A. Categories of home user applications

With rapid emergence of new broadband technologies and last mile applications, home broadband networks are becoming the major connection point for Internet users in providing access to the Internet infrastructure [5]. The home gateway encompasses competing traffic flow from a wide combination of different application services with different requirements. Such competing traffic flows strive to gain sufficiently fair capacity sharing. The dependency of providing reliable performance for Internet applications is based on the characteristics of the network.

To better understand the characteristics of traffic in the home broadband, we categorise home user application that uses TCP based on application traffic patterns: (a) latency-tolerant applications; (b) latency-sensitive applications [6].

Latency-tolerant applications include long-lived bulk file transfers, traffic with repeated burstiness such as video streaming, etc.

Latency-sensitive applications are interactive real time applications that need low RTTs for optimal performance and end user experience. Such applications are sensitive to delays and packet loss in a very transparent way to the end user. Latency-sensitive applications such as interactive real time applications, VoIP, first person shooter

(FPS) gaming, video conferencing, web-page loading and medical telemetry device traffic are most likely to have performance degradation when concurrently present with latency-tolerant applications.

B. Factors that lead to unexpected TCP behaviour

Each individual TCP connection depends upon several characteristics [7]. TCP uses sliding window flow control to represent how many packets or bytes can be in-flight, at any one time without having been acknowledged (ACKed). Most common TCP congestion control (CC) [8] algorithms that are deployed utilise packet loss as their congestion signal to detect congestion in the network and to indicate that the network path capacity has been exceeded [9], [10].

1) Impact of TCP window size: Send window (`swnd`) is a TCP variable that dictates how many packets are allowed to be in-flight without being ACKed. Congestion window (`cwnd`) is the TCP sender's current estimation of path capacity. Advertised receiver window (`rwnd`) is the maximum window size that the receiver is willing to tolerate from the sender. Window size for all three variables will be dynamically updated throughout the TCP connection to reflect that, `swnd` is restricted by the minimum value of `cwnd` and `rwnd` as shown in Equation 1. Provided that `rwnd` is really high, during early stages of the TCP connection, `swnd` is essentially controlled by `cwnd`. Additionally theoretical TCP papers will simply use `cwnd` as a shorthand notation of `swnd`, because they assume that `rwnd` is not a limiting factor – in our case `rwnd` is a limiting factor, and it should be taken into consideration.

$$swnd = min(cwnd, rwnd) \tag{1}$$

2) The need for buffering: TCP slow start mode has the ability to rapidly grow `cwnd` in response to successful packet transfers and ACKs [7]. The rapid growth of `cwnd` will allow for quicker approximation of the path capacity. In the event of packet loss, TCP will switch into congestion avoidance mode, where `cwnd` growth is much stable and less aggressive. In congestion avoidance mode, `cwnd` continually probes and oscillates around the optimal point, where `swnd` puts as many packets as possible without exceeding the path capacity.

For an optimal transmission rate, the `swnd` should be at least able to exceed the bandwidth delay product (BDP) [11], [12]. When `swnd` starts to exceed BDP, the remaining packets are accumulated in a bottleneck which require additional buffering (B_{buf}). A bottleneck is a point in the network where there is likely to be transient brief periods of time when there is more packets arriving (or bursts of packets) faster than it can be forwarded out. During these bursts, packets need to be stored in a local B_{buf}. Each network equipment will have a B_{buf} size in order to process bursts of packets that arrive on the inbound link and then export packets on the outbound link.

When `swnd` exceeds the maximum available capacity ($BDP + B_{buf}$) as shown in Equation 2, we will observe packet loss because there is no available space for new packets. NewReno TCP CC reacts to this packet loss by halving (or multiplier factor of 0.5)[1] its `cwnd`.

$$swnd > BDP + B_{buf} \tag{2}$$

Using NewReno TCP CC algorithm we would like to satisfy Equation 3 to achieve maximum TCP utilisation when packet loss occurs. In order to satisfy Equation 3, $BDP = B_{buf}$ to achieve optimal utilisation.

$$BDP = 0.5 * (B_{buf} + BDP) \tag{3}$$

However, any $B_{buf} < BDP$ will result in less than maximum utilisation and reduce queuing delay. On the other hand, any $B_{buf} > BDP$ will have additional queuing delay without any gain in utilisation. Another thorn in side, which end users may not have control over is the vendor equipment B_{buf} size. Typically B_{buf} size is a fixed value, and the effect of this rationale is that it places an upper bound limit on the BDP.

3) Implications of RTT: One way delay (OWD) is the time it takes to transmit a packet from a source to the destination (OWD_{up}) or from destination to the source (OWD_{down}). RTT_{base} is the sum of both OWDs ($OWD_{up} + OWD_{down}$). Both downstream and upstream flows will have their own B_{buf} which in turn causes RTT_{queue} delays – which depends on the B_{buf} packet occupancy and B_{buf} link speed, as shown in Equation 4.

$$RTT_{queue} = \frac{B_{buf}\, packet\ occupancy * MTU}{B_{buf}\, link\ speed} \tag{4}$$

'B_{buf} packet occupancy' in Equation 4 can be altered to include the maximum B_{buf} size in order to determine the maximum RTT_{queue} delay.

In a scenario, where there is network equipment with an inbound link of 1Gbps and a outbound link of

[1]Not all TCP CC halve their cwnd, for instance Cubic uses a multiplier factor of 0.7.

100Mbps, on average the inbound rate will be approximately 100Mbps, however, the inbound link will have to process bursts of packets at the outbound link rate of 1Gbps. Buffering plays an important role in this scenario because the network equipment should be able to process bursts of packets in the inbound link and export them on the outbound link. In this particular scenario, having a large buffer reduces potential idle time of processing packets in the queue, but adds additional delay from RTT_{queue} to the RTT_{base} which increases the overall RTT_{actual} as shown in Equation 5.

$$RTT_{actual} = RTT_{base} + RTT_{queue} \tag{5}$$

III. TEACUP Testbed

The experiments involved TCP connections developed for a range of representative home broadband scenarios, using CAIA's TEACUP testbed as shown in Figure 1.

All scenarios have either upstream traffic pushing content and/or downstream traffic pulling content through 8 end hosts (4 either side) with a Linux-based bottleneck router. Each scenario trial ran for 90 seconds, using single and multiple streams of iperf traffic (from client to server only) with 0% intrinsic loss. FreeBSD end hosts used NewReno as their TCP CC. The B_{buf} sizes for Packet First In First Out (PFIFO) trials were 20 and 320 packets.

All scenario trials were emulated with iperf traffic: (a) a single downstream flow; (b) three downstream flows and one upstream flow (upstream traffic starting last); (c) three upstream flows and one downstream flow (downstream traffic starting last). Table I summarises each experiment's emulated path condition(s).

IV. Anomalous scenarios

Section IV-A scenario outlines the importance of `swnd`. In this section we observe the behaviour of the `swnd` when the maximum `rwnd` is less than BDP.

Section IV-B scenario presents the results that show the effect of having multiple flow experiments with an initial dominant flow in a home gateway router.

A. *Scenario 1: `cwnd` capped*

Figures 2a, 3a and 4a represent experimental trials that resulted in `cwnd` capped at a certain rate for the entire duration and/or `cwnd` capping within the initial time frame after slow start.

Figure 4a is an experiment that has its `cwnd` capped for the entire duration of the experiment, which is imposed by Equation 1, thus (assuming no other sources of packet loss) `swnd` is capped at `rwnd`.

Figure 4b is a single downstream flow experiment that investigates the impact on achieved throughput when `cwnd` is capped for the entire duration of the experiment. Figure 4b is expected to achieve high link utilisation and throughput, however, it only exhibits about 25% link utilisation with consistent throughput. Figure 4b achieve consistent throughput compared to Figures 2b and 3b.

Figures 2c and 3c have RTT_{base} of 240ms and 340ms respectively, and due to large B_{buf} of 320 packets, we could see additional delay introduced (as described in Equation 5) to the overall RTT_{actual}. This is exhibited in Figure 2c where after the initial spike, the delay is kept at a rate higher than 240ms which is the consequence of additional delay. On the other hand, Figure 4c shows an initial RTT spike caused by the additional delay but it stabilises and maintains RTT_{base} throughout the experiment.

Figures 2b and 3b show multiple flows sharing the bottleneck capacity. Figure 2b shows three downstream flows reaching maximum throughput during the initial 10s. Once the single upstream flow starts at 30s, `cwnd` is reduced as shown in Figure 2a (at 30s mark). The effect of `cwnd` being reduced meant that `swnd` was also reduced by the minimum value of `cwnd` instead of `rwnd` (as described in Equation 1) which resulted in degraded throughput.

Single flow experiment with high bandwidth and RTT scenario (Figure 4a) shows a capped maximum `cwnd` of 1228kB[2], but the receiver never advertised a larger `rwnd` of 1047kB[2], which meant that `swnd`[3] was restricted by the `rwnd,` as described in Equation 1. The effect of this was that there was not enough packets to fill up the BDP and/or B_{buf} to cause packet loss, as described in Equation 2. On the other hand, for experiments with multiple flows (Figures 2a and 3a), `swnd` satisfied Equation 2 which resulted in overflow packets in the queue causing one or more flow(s) to experience packet loss and to trigger `swnd` cycling. As soon as there are enough flows such that the aggregate of all of these flows exceed the number of packets in Equation 2 to cause packet loss, the normal TCP behaviour occurred, which meant that `swnd` was changing the window size based on Equation 1.

[2]Confirmed by inspecting SIFTR field 9 ("The current congestion window for the flow") and SIFTR field 12 ("The current receive window for the flow") respectively. [13]

[3]`swnd` was capped at `rwnd` value of 1047kB. This was confirmed by inspecting SIFTR field 11 ("The current sending window for the flow").

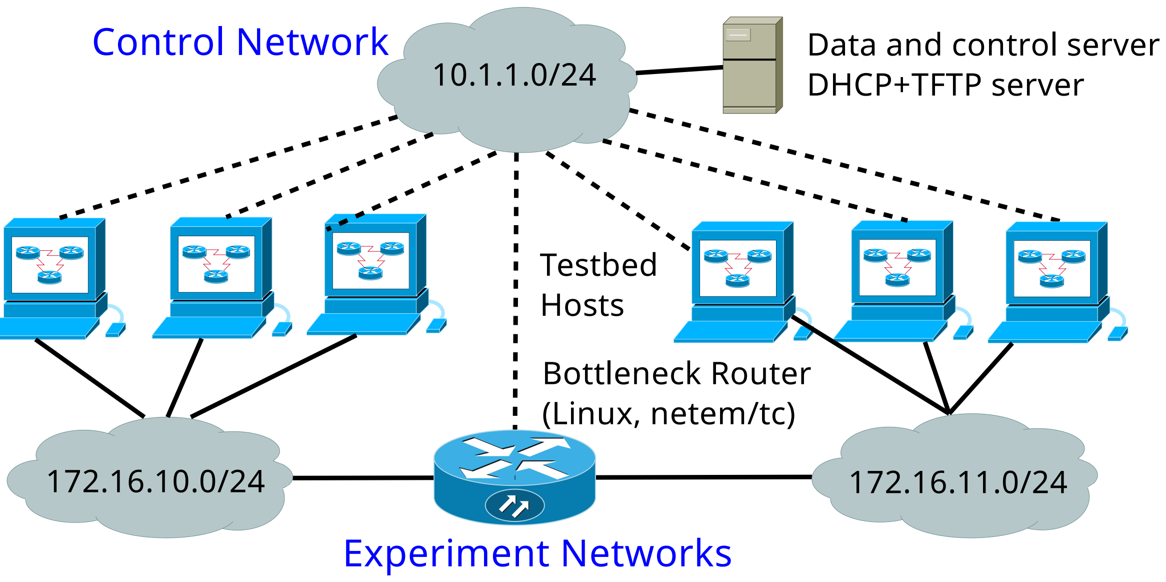


Figure 1: TEACUP Testbed

Scenario	TCP CC	AQM	B_{buf} size (packets)	Bandwidth (down/up)	RTT$_{\text{base}}$ (ms)	Number of flows (down/up)	cwnd Figure	Throughput Figure	RTT$_{\text{actual}}$ Figure
1	NewReno	PFIFO	320	100/40Mbps	240	3/1	2a	2b	2c
					340		3a	3b	3c
						1/0	4a	4b	4c
2				1500/500Kbps	240	1/3	5a	5b	5c
					340	3/1	6a	6b	6c
			20	50/20Mbps		1/3	7a	7b	7c

Table I: Scenario conditions

B. Scenario 2: Flow domination

When there are multiple competing flows, the flow that starts first often has a distinctive advantage of getting the chance to grow its cwnd without having to share the path with any other flows. Such behaviour occurs as a result of synchronisation or timing effects, where a single flow dominates the queue while preventing any other flows from placing their packets in the queue [14]. Figures 5a, 6a and 7a illustrate that the initial flow dominates the path before any other flows start. When other flows start, they are forced to experience the negative impact of sending packets into a path where the available capacity is already consumed by the initial flow.

Figures 5 and 6 have small BDP (Figure 5 initial upstream flow has 14kB and Figure 6 initial downstream flow has 62kB) and large B_{buf} (320 packets or 480kB[4])

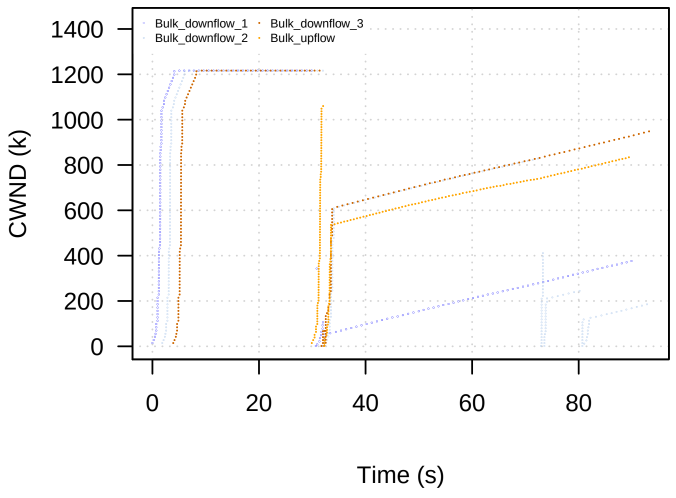
[4]If MTU is 1500 Bytes.

whereas Figure 7 has a large BDP (initial upstream flow has 830kB) and a small B_{buf} (20 packets or 30kB[4]).

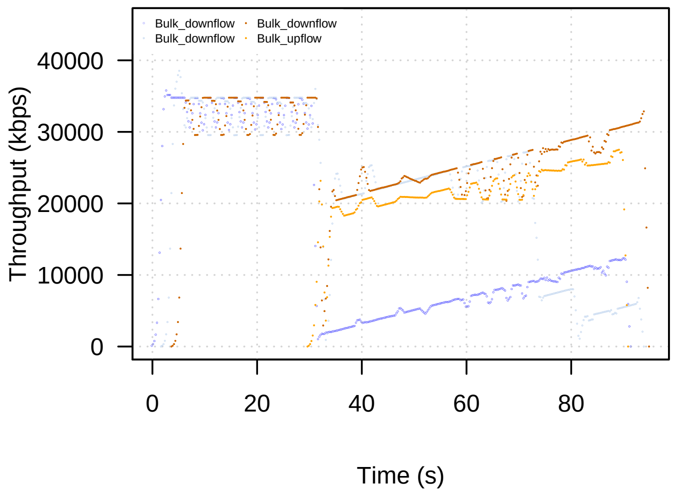
Equation 3 states when $B_{buf} > BDP$ that there will be additional queuing delay to the overall RTT_{actual} (as discussed in Equation 5) without any further gain in utilisation. Figures 5 and 6 have a $B_{buf} \gg BDP$ which will result in a significantly large RTT_{queue} value in Equation 5 to the point where already existing high RTT_{base} of 240ms and 340ms respectively will be negligible – this is evident in Figures 5c and 6c, where the overall RTT_{actual} throughput the whole experiment was predominantly high. Additionally having $B_{buf} \gg BDP$ should result in maximum utilisation of throughput. This was the case for Figures 5 and 6, but the initial starting flow consumed the most throughput than the other subsequent flows during the experiment. The initial dominant upstream flow in Figure 5b reached and maintained a maximum throughput of 500Kbps throughout the experiment, and the initial dominant downstream flow in Figure 5b reached and maintained a maximum throughput of 1500Kbps during 30s of the experiment.

On the other hand, when $B_{buf} < BDP$ the maximum utilisation and queuing delay are decreased. Figure 7 has a $B_{buf} \ll BDP$ which means the additional queuing delay RTT_{queue} term in Equation 5 becomes negligible and the overall RTT_{actual} should approximately be the same as the RTT_{base} of 340ms as shown in Figure 7c. Additionally having $B_{buf} \ll BDP$ should result in underutilised throughput. The initial dominant upstream flow in Figure 7b never managed to fully utilise the 20Mbps upstream bandwidth, however, all three upstream flows managed to fully utilise the 20Mbps upstream bandwidth during very brief period of time (at 50s and 90s) throughout the experiment – as expected, at this period of time, the cwnd was at its maximum point as seen in Figure 7a.

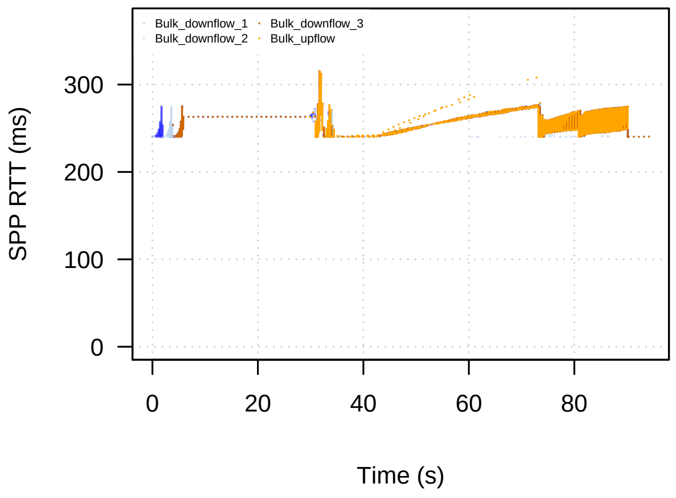
As discussed in Section II-B2, $BDP = B_{buf}$ is required to achieve optimal utilisation without introducing additional delay.



(a) cwnd vs. Time

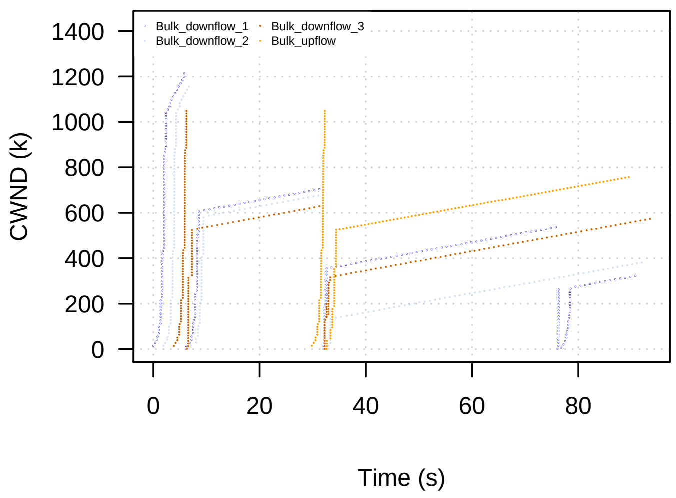


(b) Throughput vs. Time

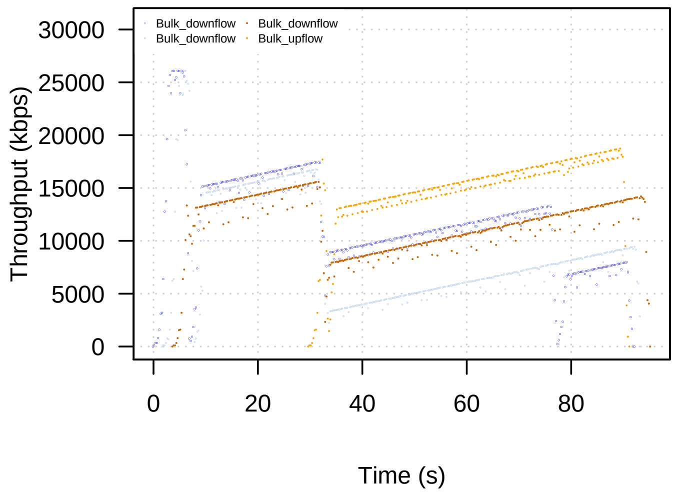


(c) $\text{RTT}_{\text{actual}}$ vs. Time

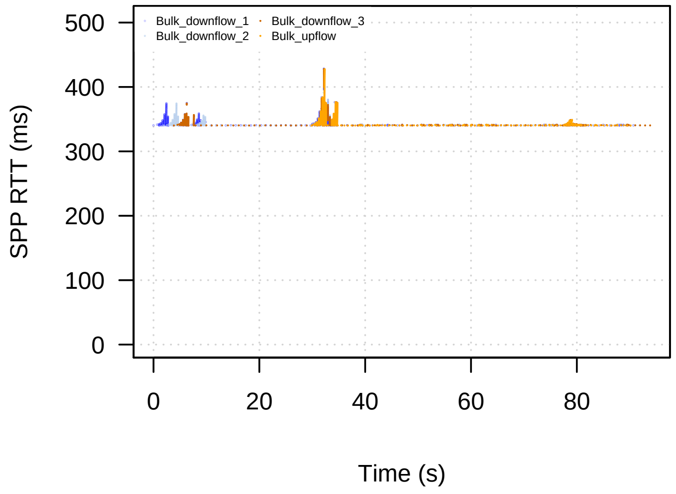
Figure 2: Three downstream and one upstream NewReno flows over 240ms RTT path with 100/40Mbps bandwidth and 320 packet pfifo buffer



(a) cwnd vs. Time

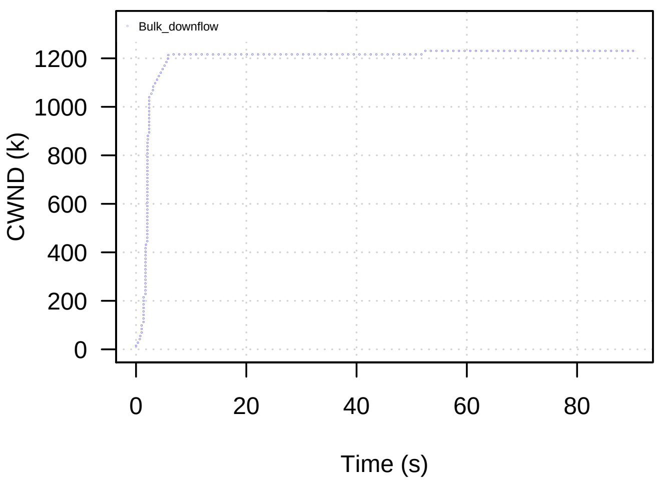


(b) Throughput vs. Time

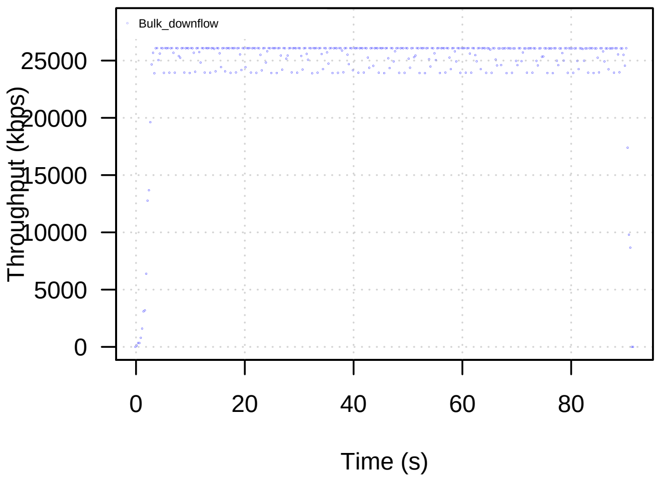


(c) $\mathrm{RTT}_{\mathrm{actual}}$ vs. Time

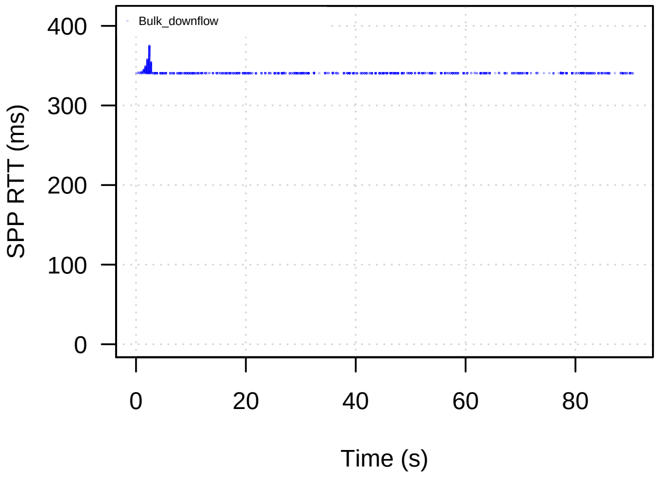
Figure 3: Three downstream and one upstream NewReno flows over 340ms RTT path with 100/40Mbps bandwidth and 320 packet pfifo buffer



(a) cwnd vs. Time

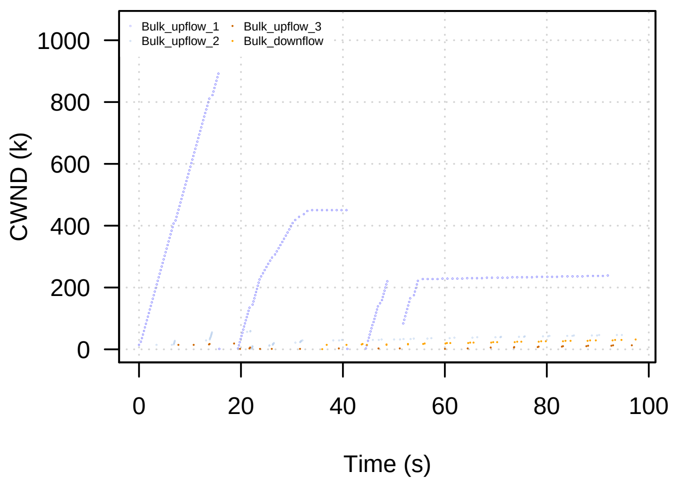


(b) Throughput vs. Time

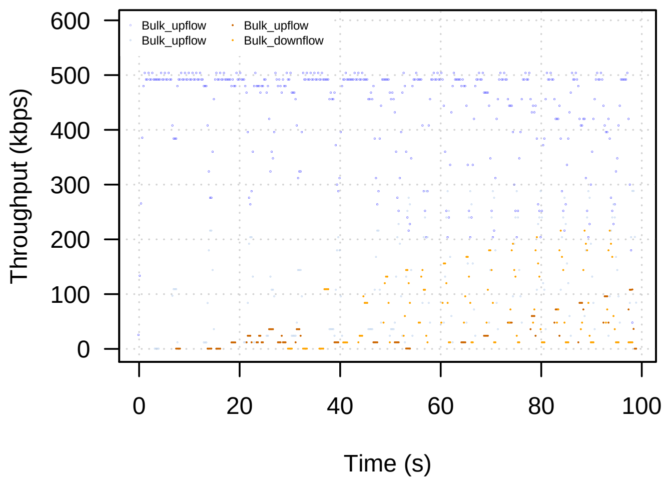


(c) $\mathrm{RTT}_{\mathrm{actual}}$ vs. Time

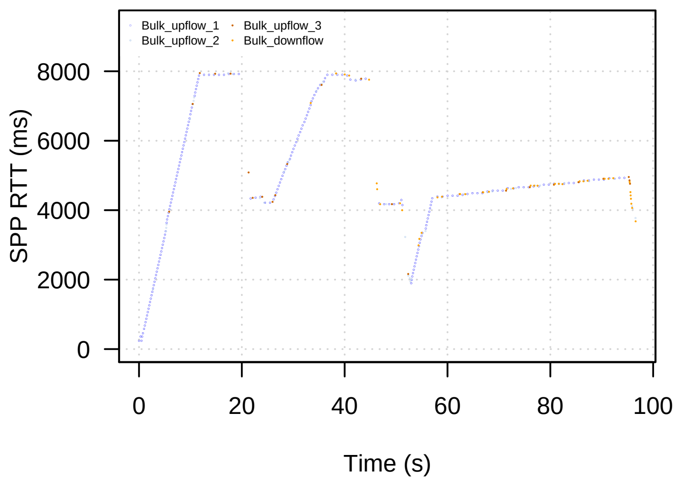
Figure 4: One downstream NewReno flow over 340ms RTT path with 100/40Mbps bandwidth and 320 packet pfifo buffer



(a) `cwnd` vs. Time

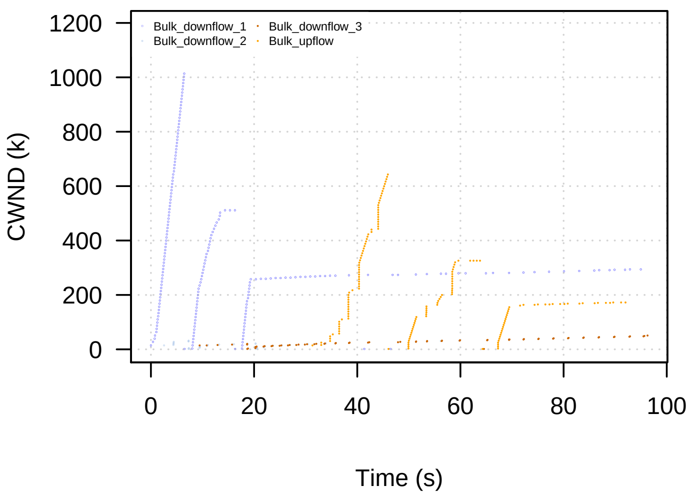


(b) Throughput vs. Time

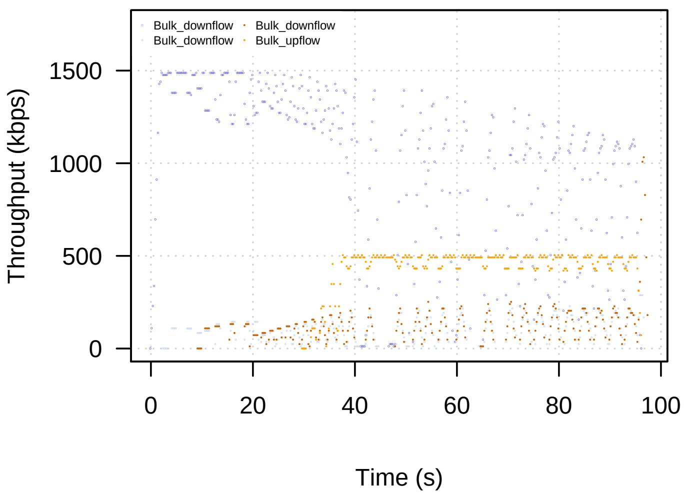


(c) $\mathrm{RTT}_{\mathrm{actual}}$ vs. Time

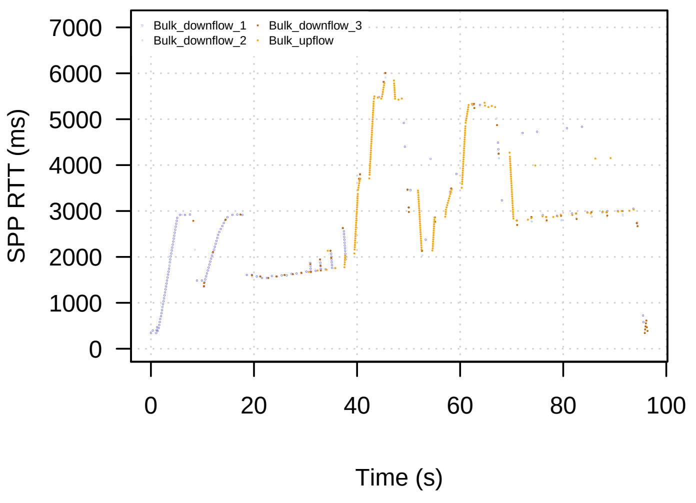
Figure 5: One downstream and three upstream NewReno flows over 240ms RTT path with 1500/500Kbps bandwidth and 320 packet pfifo buffer



(a) `cwnd` vs. Time

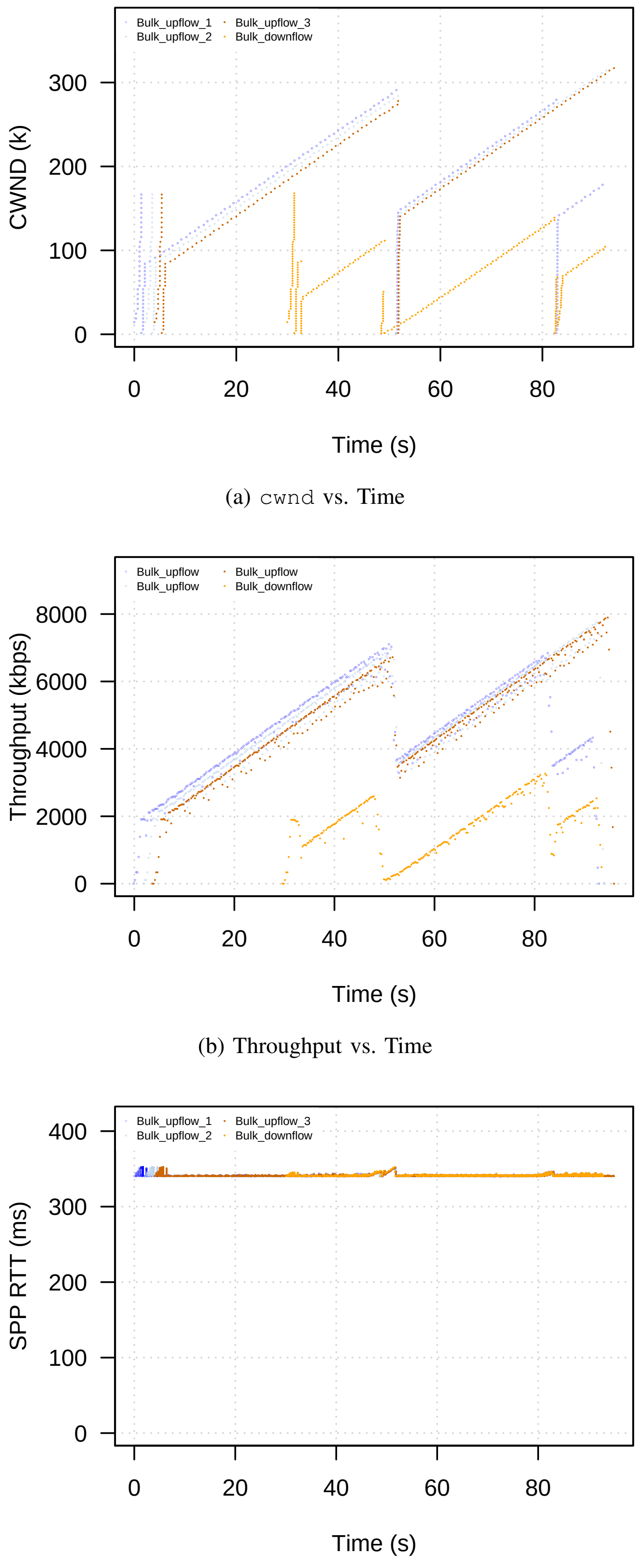


(b) Throughput vs. Time



(c) $\mathrm{RTT}_{\mathrm{actual}}$ vs. Time

Figure 6: Three downstream and one upstream NewReno flows over 340ms RTT path with 1500/500Kbps bandwidth and 320 packet pfifo buffer



(a) `cwnd` vs. Time

(b) Throughput vs. Time

(c) $\text{RTT}_{\text{actual}}$ vs. Time

Figure 7: One downstream and three upstream NewReno flows over 340ms RTT path with 50/20Mbps bandwidth and 20 packet pfifo buffer

V. Conclusion

We have explored two scenarios where the relationship between `rwnd`, BDP, and B_{buf} can lead to anomalous behaviour of TCP while switching from TCP slow start mode into congestion avoidance mode. Scenario 1 in Section IV-A investigated the effect of `swnd` when `cwnd` is greater than `rwnd` resulting in certain limited window size as detailed in Equation 1. The results presented in scenario 1 are caused by large BDP and high B_{buf} (higher than `rwnd` of any single individual receiver window) which resulted in `swnd` being capped at a certain rate, predominately limited by the receiver (with their default advertised receiver window) not allowing TCP connection to fully utilise the path. Scenario 2 in Section IV-B investigated the effect of BDP and B_{buf} on throughput and overall RTT_{actual} in multiple flow experiments with an initial dominant flow. The results show that when $B_{buf} \gg BDP$ there will be additional queuing delays with no additional utilisation in throughput and that the RTT_{base} value in Equation 5 becomes negligible. Whereas, when $B_{buf} \ll BDP$ the throughput will be underutilised with no (or very small) queuing delay and that the RTT_{queue} term in Equation 5 becomes negligible.

In scenario 1, if `swnd` in Equation 2 is not satisfied, then there will be no packet loss. NewReno is a loss based TCP CC algorithm and relies on packet loss to indicate that the network path capacity has been exceeded. Since the receiver never increases its `rwnd`, `swnd` is capped at the rwnd (Equation 1). To ensure that we achieve an optimal path utilisation, receiver must increase its `rwnd` to increase `swnd` (assuming `cwnd` is maintained at a high rate) to satisfy Equation 2. In scenario 2, $BDP = B_{buf}$ is required to achieve optimal utilisation without introducing additional delay. Deploying active queue management (AQM) would potentially be a feasible solution to dynamically adjust B_{buf} of individual flows.

Acknowledgements

This work was supported by a Swinburne University of Technology / Cisco Australia Innovation Fund project titled "An evaluation of household broadband service requirements for educational innovation and Internet of Things". We would like to thank Prof. Grenville Armitage for his supervision and assistance. We would also like to thank Jonathan Kua for his version of the TEACUP testbed topology.

References

[1] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of tcp/red and a scalable control," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 239–248, IEEE, 2002.

[2] J. Postel, "Transmission control protocol," STD 7, RFC Editor, September 1981. http://www.rfc-editor.org/rfc/rfc793.txt.

[3] H. Wu, K. Long, S. Cheng, J. Ma, and Y. Le, "Tcp friendly fairness in differentiated services ip networks," in *Networks, 2001. Proceedings. Ninth IEEE International Conference on*, pp. 81–86, IEEE, 2001.

[4] S. c. Tsao, Y. c. Lai, and Y. d. Lin, "Taxonomy and evaluation of tcp-friendly congestion-control schemes on fairness, aggressiveness, and responsiveness," *IEEE Network*, vol. 21, pp. 6–15, November 2007.

[5] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, "Measuring and mitigating web performance bottlenecks in broadband access networks," in *ACM Internet Measurement Conference*, 2013.

[6] L. Stewart, D. A. Hayes, G. Armitage, M. Welzl, and A. Petlund, "Multimedia-unfriendly tcp congestion control and home gateway queue management," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, (New York, NY, USA), pp. 35–44, ACM, 2011.

[7] W. R. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, RFC Editor, January 1997. http://www.rfc-editor.org/rfc/rfc2001.txt.

[8] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control," RFC 5681, RFC Editor, September 2009. https://tools.ietf.org/html/rfc5681.

[9] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug 1993.

[10] N. I. A. Altahir and H. A. Ali, "Performance evaluation of tcp congestion control mechanisms using ns-2," in *2016 Conference of Basic Sciences and Engineering Studies (SGCAC)*, pp. 146–151, Feb 2016.

[11] T. V. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 336–350, Jun 1997.

[12] M. A. Alrshah and M. Othman, "Performance evaluation of parallel tcp, and its impact on bandwidth utilization and fairness in high-bdp networks based on test-bed," in *Communications (MICC), 2013 IEEE Malaysia International Conference on*, pp. 23–28, Nov 2013.

[13] L. Stewart and J. Healy, "siftr(4) manual page." https://www.freebsd.org/cgi/man.cgi?query=siftr.

[14] B. Braden, D. D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the internet," RFC 2309, RFC Editor, April 1998. http://www.rfc-editor.org/rfc/rfc2309.txt.