Effects of on-path buffering on TCP fairness

Saleem Bhatti, Martin Bateman
School of Computer Science, University of St Andrews, St Andrews, UK
email: {saleem, mb}@cs.st-andrews.ac.uk

Abstract

Keeping router buffering low helps minimise delay (as well as keeping router costs low), whilst increasing buffering minimises loss. This is a trade-off for which there is no single 'correct' solution. In order to maintain effective throughput for TCP, whilst minimising router buffer requirements, current results suggest that different amounts of buffering are needed depending on the position in the network (e.g., edge or core), and on the relative capacity of ingress and egress links to a router. However, today we have several different variants of TCP in use, and each is designed to have different behaviour especially on paths with high bandwidth-delay product (BDP) values. We use a testbed to investigate the effects of different amounts of 'on-path' buffering (OPB) on the performance of four TCP variants – TCP NewReno, BIC, CUBIC, and Compound TCP – over various end-to-end round-trip-times (RTTs). Specifically, we consider how the variants respond when competing for bandwidth on a bottleneck link. We find that overall performance depends on both the RTT and the OPB provision, and that the observed behaviour is not consistent across the range of RTT and OPB values.

1. Introduction

There has been much discussion on the amount of 'on-path' buffering (OPB) in routers – buffering that is needed to achieve good end-to-end performance for TCP. This is an important issue to consider as high-speed RAM for routers is expensive, and buffering adds delay for end-to-end packet transfer. The previously established 'rule', to have one bandwidth-delay product (BDP) worth of buffering [18], has been contested by the use of small buffers [1] and tiny buffers [15]. Results suggest that the nature of the buffering provision depends on where in the network a router is placed [8, 14, 19].

Meanwhile, it is known that TCP behaviour can be effected by the end-to-end delay on the path (e.g., [13]) and by the buffering used on the path (e.g., [8, 14, 19]). This is important information for end-users: the efforts they make to configure and tune the parameters of their end-system stack may not necessarily result in the throughput they expect because of the on-path buffering used between their host and the destination of a flow from that host.

Also several variants of TCP are currently in use. *TCP NewReno* (RFC3782) is the variant of TCP that is the default in Windows XP, Windows Vista, and MacOSX (amongst others). *BIC* [20] and *CUBIC* [21] are the default versions of TCP in place of NewReno in Linux kernels over the past few years[1]. *Compound TCP* [17] is available in Windows Vista, Windows Server 2008, and available as a hotfix to Windows 2003 server and Windows XP 64-bit[2]. Compound TCP is also available for Linux.

These TCP variants have different designs, even though they share a common goal: to operate in environments with high bandwidth-delay product (BDP) paths. BIC and CUBIC use loss signals to trigger congestion control behaviour, whilst Compound TCP uses delay/RTT information. Each variant offers some improvement in end-to-end throughput compared to NewReno, in a specific situation: to make use of available network capacity that NewReno cannot utilise effectively due to its relatively conservative congestion control behaviour. The use of congestion control in TCP has been the key to the Internet's stability, so any change to this behaviour merits investigation.

1.1. Contribution and structure

This paper addresses the question, "For out-of-the-box configurations, how do the common TCP variants behave with each other on paths with a range of on-path buffering provision and round trip time values?"

The contribution of this paper is an examination, using a testbed, of the *fairness* between the different TCP variants under a variety of OPB and RTT scenarios. We have used a wide range of values for both OPB and RTT to represent

[1]NewReno before kernel version 2.6.8, August 2004; BIC from Linux kernel version 2.6.8, August 2004; CUBIC from Linux kernel version 2.6.19, September 2006.

[2]`http://support.microsoft.com/kb/949316`

many different end-to-end path scenarios. Whilst BIC and CUBIC have been examined under various conditions in the previous studies listed above, there have been no studies that examine specifically the use of OPB as a variable, and compare performance with Compound TCP.

We present our methodology and description of our testbed (Section 2), followed by a description of our evaluation criteria (Section 3). We discuss our results (Section 4) and then conclude (Section 5).

2. Methodology

We have used a simple testbed to measure end-to-end throughput for TCP flows to evaluate the relative performance of the TCP variant when operating together over a bottleneck link.

2.1. Testbed

Our testbed was the well-known dumbbell arrangement (Figure 1), as used in previous similar studies [5, 9, 12, 16]. This simple testbed helps to reduce the factors of error that may affect the results and concentrate on the protocol behaviour. As noted in [7], "Simple topologies, like a 'dumbbell' topology with one congested link, are sufficient to study many traffic properties." Indeed, this simple testbed helps us to reduce the factors of error or unknown behaviour that may affect the results. The IRTF Transport Modelling Research Group[3] are also looking to define mechanisms and metrics for evaluation of transport protocols, including fairness. Their ongoing work suggests that a dumbbell topology is one scenario that can be used to evaluate transport protocol performance [6].

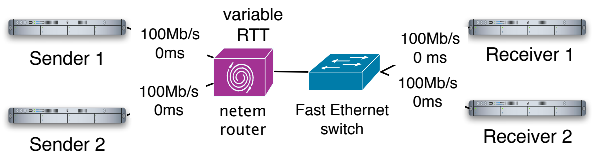


Figure 1. Testbed configuration

The dumbbell configuration, consists of two senders, two receivers and a router to provide the network delay. All the nodes were Intel Xeon Dual Core 1.6Ghz with 2GB of DDR2 (667Mhz) memory and a PCI-X (64-bit, 133Mhz) bus. All network connections were 100Mb/s full-duplex Ethernet, with no flow control (IEEE 802.3x not used). The senders and receivers were running Linux kernel version 2.6.22.6, while the router was running Linux kernel version 2.6.18. The package *netem*[4] was used at the router to introduce delay to the packet flows, with the delay split equally between the forward and reverse paths. The two flows are sent from Sender 1 to Receiver 1 and from Sender 2 to Receiver 2.

[3]`http://www.icir.org/tmrg/`
[4]`http://www.linux-foundation.org/en/Net:Netem`

The senders and receivers ran Linux kernel version 2.6.22.6, and we used *out-of-the-box* configuration for the end-systems, rather than tuning the stack for high-speed operation, in order to gauge the performance under (arguably) the most likely configuration of the end-system. TCP Selective Acknowledgements (SACK, RFC2018) was enabled; the Window Scale Option, Protection Against Wrapped Sequence Numbers, and Round-Trip Time Measurement (RFC1323) were enabled; and MTU size was 1500 bytes (no IP fragmentation): all these being default settings.

The implementation of Compound TCP used in our study is Caltech's Linux patch[5], which is written to the same specification used for the Windows implementation. As NewReno, BIC and CUBIC are all available in Linux, the Caltech implementation lends itself for easy use within our testbed, and we do not have to factor into our analysis any differences due to the behaviour of operating system if running Compound TCP under Windows.

2.2. Testbed calibration and validation

The router buffer size – OPB – was set to a given percentage of the BDP at each RTT value, the BDP being evaluated as 100Mb/s multiplied by the RTT value. The OPB and RTT values used are given in Table 1. This means, for each TCP variant a total of 80 data points are produced for each plot in Figures 3, 4, and 5. Whilst router buffer sizing remains an open issue, it is clear that buffer sizing does affect TCP throughput [14], and so it would seem appropriate to consider it as a potential factor to test. Of course, packet loss (due to congestion or transmission errors) can also affect TCP throughput [13]: we leave that for future work.

Parameter	Range	Increments
OPB (% of BDP)	20% – 100%	20%
RTT	25ms – 400ms	25ms

Table 1. Experimental parameters for the testbed

For each RTT value, the testbed was validated:

- by testing the RTT by the use of *ping* from sender to receiver: *ping* reported the delay was accurately configured (to within $\sim$1ms).
- by the use of *iperf* with UDP flows to ensure that a packet-level throughput of 100Mb/s was possible.

[5]`http://netlab.caltech.edu/lachlan/ctcp/`

2.3. Flows and measurement

For Figure 1, our measurement runs consisted of generating two flows, using *iperf*, for a pair-wise comparison: Flow 1 was from Sender 1 to Receiver 1, and Flow 2 was from Sender 2 to Receiver 2. The duration of each measurement run was 300s.

We make discrete measurements of end-to-end throughput for each flow, at intervals of time, t. We use the default for *iperf*, which reports throughput at intervals of 1s.

The TCP flows were generated using a modified version of *iperf* v2.0.2[6]: the modification allowed us to switch easily between the TCP variants used on a per-flow basis. Each experimental run was 300 seconds. The first flow, Flow 1, was started at $t = 0s$ and the second flow, Flow 2, was started at $t = 60s$ to help avoid initial synchronisation effects. The calculation in our evaluation used the throughput reported by *iperf* from 120s to 300s at 1s intervals, i.e. the first 119 measurements reported were ignored, to give the flows time to reach steady-state. Each run was repeated 5 times, with the mean of the 5 values taken as the measured data point value for that RTT-OPB combination. As explained in Section 2.2, 80 data points are used. For each data point, as we take the mean value over 5 runs, 400 flows are used for each data point, and:

- Each pair-wise experiment generated a total of 240000 measurements (300s), of which 144800 (180s) were used.
- For Figure 3 and Figure 4, only a single flows is considered, so 72400 measurements were used (from a total of 120000) for each plot.
- For Figure 5, we have 2 flows, so 144800 measurements were used (from a total of 240000) for each plot.

For the self-fairness tests, Flow 1 and Flow 2 were both the same TCP variant. For the inter-protocol testes, Flow 1 was always TCP NewReno (as the reference TCP implementation and algorithm) and Flow 2 was, in turn, BIC, CUBIC, and Compound TCP.

3. Evaluation

We choose to see how *fair* the TCP flows are to each other when competing across the bottleneck link, i.e., the relative share of the available end-to-end capacity each flow receives. Our end-to-end throughput measurements are used to with a metric, the *Normalised Resource Usage* to produce a comparison of the flows' performance.

[6]The modified version of *iperf* is available, upon request, from the authors.

3.1. Jain's Fairness Index (JFI)

Jain's Fairness Index (JFI) [10] is widely used for assessing *system-wide* fairness, as in Equation (1), where, $0 \leq J \leq 1$, N is the number of flows, r_n is the value of the resource attribute being assessed for flow n, e.g. r_n is the measured end-to-end throughput. $J = 1$ means there is fairness across all flows; $J = 0$ indicates no fairness.

$$J = \frac{\left(\sum_{n=1}^{N} r_n\right)^2}{N \sum_{n=1}^{N} r_n^2} \tag{1}$$

JFI has been used in previous studies, e.g., [9, 16], including our own, e.g. [3, 12]. However, the JFI has some properties that means it may not provide a suitable evaluation metric [4]:

- JFI may be difficult to interpret. For example, in Figure 2 we have created an artificial situation with two flows. Flow 2 is held constant at 100 and the value of Flow 1 is varied from 1 to 10000: as the ratio Flow 1 / Flow 2 changes over four order of magnitude, the value of JFI (Equation 1) has the range $[0.51, 1.00]$. Comparisons between different values of JFI are difficult because the difference in value has different significance at different points across this range of values. For example, a difference of 0.1 in the JFI value between two experiments has different significance if the absolute vales being compared are close to 1.0 or if they are close to 0.5
- The definition of the JFI assumes that each process being evaluated is equally capable of consuming a resource. This may not be true. For example, the TCP variants BIC, CUBIC and Compound TCP have all been designed to have greater throughput than TCP NewReno in environments with high BDP, i.e., they are designed not be of equal capability to NewReno.
- In practise, as long as at least one of the N flows is non-zero, the value of the JFI has the range $1/N \leq J \leq 1$ and not $0 \leq J \leq 1$. This means it may be difficult to compare JFI values between experiments with different numbers of flows.

3.2. Normalised Resource Usage (NRU)

We propose a different metric when considering fairness, one that is designed to be simple but allows:

- weights to be applied that reflect relative capability, given specific resource provisioning.

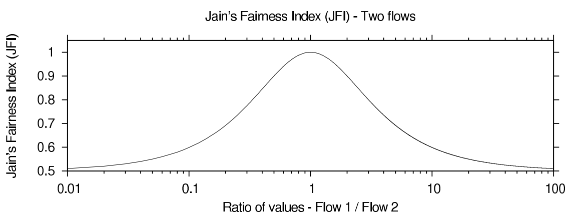


Figure 2. The range of JFI for two flows

- comparative assessments to be made, based on relative capability.

We choose to reflect the following characteristics in the output of our metric, in comparison to JFI:

- to be able to make comparisons of fairness on a *per-flow* basis, not just a *system-wide* basis.
- to enable an assessment of fairness over time (as well as a summary statistic), allowing observation of *per-flow* and *system-wide* dynamics.

In order to provide a richer view of the fairness information, we introduce a metric which is based on the ratio of resource usage of an individual flow with respect to its expected capability: the *Normalised Resource Usage (NRU)* [4]. The NRU metric, $U_n(t)$, for a flow n with throughput $r_n(t)$ at time t is defined in terms of the *Resource Share Ratio (RSR)*, $R_n(t)$:

$$U_n(t) = 10\log_{10}(R_n(t)) \quad (2)$$
$$R_n(t) = w_n(t)r_n(t) \quad (3)$$

where w_n is a weight which reflects the relative capability of the flow under the conditions being examined. Key to this metric is the evaluation of $w_n(t)$, which we address in due course. The use of the $10\log_{10}()$ deciBel term is for convenience of representing large and small values. When $U_n(t) = 0$, then flow n is receiving a fair share of the available resource. When $U_n(t) < 0$, then flow n is receiving less than its fair share of the resource. When $U_n(t) > 0$, flow n is receiving more than its fair share.

For a flow n with a set of values $U_n(t)$, over a given time-period, we can generate a summary by taking the mean, U_n, of the values in $U_n(t)$. We call this the *flow-NRU* (Equation 4). Assuming a system has N flows, $1 \leq n \leq N$, a system-wide summary is obtained by the mean *system-NRU* (Equation 5), U_N, the mean of U_n for all N flows. A mean could hide unfairness, as positive and negative values of U_n would cancel out, but that is not a concern here, as we have only two flows and for one of them we show the flow-NRU directly. (A more detailed account of the NRU and its use can be found in [4].)

$$U_n = \overline{\{U_n(t)\}} \quad (4)$$
$$U_N = \overline{\{U_n\}} \quad (5)$$

The meanings of values for U_n and U_N are summarised in Table 2. Note that the U_N values are listed as 'likely': remember that the system-NRU is a mean, so negative and positive values could cancel out. However, in this study, the system-NRU, in Figure 5, will be considered in the light of the flow-NRU for Flow 2, in Figure 4, to prevent misinterpretation.

Value	U_n (flow-NRU)	U_N (system-NRU)
0 (zero)	fair share	(likely) fair
< 0	unfair/lower share	(likely) unfair system
> 0	higher share	(likley) fair

Table 2. Summary of NRU semantics.

3.3. Weights for the NRU

A distinguishing feature in our methodology and fairness metric is that we assign weights that reflect each flow's capability to consume the available resource. From Equation (3), we define:

$$w_n(t) = 1/_R r_n(t) \quad (6)$$
$$w_n = 1/\overline{_R r_n} \quad (7)$$

where $_R r_n(t)$ is the expected throughput of that flow under the conditions being examined. Equation (6) is a general expression, and we simplify this for our needs by using Equation (7), representing the weight as a scaling factor, evaluated from the mean throughput. In our case, the value of w_n for Flow 2 is taken from the mean of the calibration test for each RTT–OPB combination that we measure when each flow is run on the testbed against itself, i.e. *self-fairness*.

Given our use of *iperf*, as explained in Section 2.3, $r_n(t_j)$ is then an approximation of throughput as determined at time interval t_j, and evaluated over the period $(t_j, t_{j-1}), t_j > t_{j-1}$, where t_{j-1} is the previous time at which an approximation was determined. For our experiments, *t* was every second. So, it is easy to evaluate, $U_n(t)$, U_n and U_N directly from the end-to-end throughput values reported by *iperf*.

4. Results

We describe first the results for our determination of weights for each variant, w_N, and then consider the *inter-flow fairness* across the TCP variants.

4.1. Self-fairness and weights, w_n

By measuring how two flows of the *same* TCP variant behave when competing for the bottleneck link we can establish the baseline capability under the conditions being tested, and establish values for w_n for our set of experiments. Effectively, we are examining *self-fairness*: how well the TCP variant behaves with another flow of the same variant. For our comparative evaluation of *inter-flow fairness*, the new variant being introduced will be Flow 2. So, for the case where both flows are of the same variant, we show in Figure 3, the mean throughput for Flow 2 for each of the RTT–OPB combinations when the *same* variant is used for both flows. Considering Equation 7, the mean value of throughput recorded for Flow 2 at a particular RTT–OPB combination can be taken as the expected capability of that TCP variant, ${}_{R}r_n$, and so we have our values of ${}_{R}r_n$ for evaluating w_n (Equation 7) as required for the flow-NRU (Equation 4).

Before examining the NRU analysis, it is interesting to note the individual behaviour of Flow 2 in our testbed, as shown in Figure 3. Considering first NewReno, Figure 3(a), we see that the end-to-end throughput of NewReno is highly variable, ranging from as low as 8.10Mb/s to 49.41Mb/s. We see also that as the RTT on the path increases, the throughput of Flow 2 decreases. This is consistent with observed behaviour of TCP in other studies. We note that increased OPB improves end-to-end throughput at lower RTTs (below 150ms), but not at higher RTTs.

For BIC, CUBIC and Compound TCP, we note that they exhibit some similarities in overall behaviour. They have very similar minimum, mean and maximum throughputs. Also, the plots of Figures 3(b), 3(c) and 3(d) are very similar in shape, including peaks an troughs. This is to be expected of BIC and CUBIC, as other studies have showed that they behave similarly [11, 12]. Compound TCP's similarity in behaviour is not considered surprising: although BIC and CUBIC respond to loss, loss was not a parameter in our experiments. The only loss that occurred in our experiments was due to congestion when both flows 'fill' the end-to-end provisioning.

In all cases, we see lower throughput for Flow 2 at the lower OPB values, and that throughput improves as OPB increases. This is because Flow 2 starts sometime after Flow 1, and so Flow 1 has already occupied much of the OPB provision for the path. As both flows are high-rate and long-lived (1500-byte streams of packets from Sender to Receiver), Flow 2 does not always get a chance to occupy an equal proportion of the buffer space as Flow 1, so achieves lower throughput. This behaviour is accentuated at the low OPB values: Flow 1 'grabs' most of the available buffer space, hence the troughs at the lower OPB values. It manages to occupy more buffer space approximately half of the time, across our range of experiments, hence the regular peaks and troughs observed in Figures 3(b), 3(c) and 3(d).

4.2. Inter-flow fairness

Let us now consider the flow-NRU, U_n for each variant, as shown in Figure 4. Again, we examine the flow-NRU for Flow 2, as one of the other TCP variants, while Flow 1 is always NewReno.

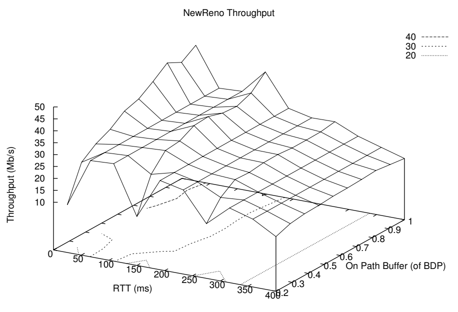
If we compare plots of throughputs in Figure 3 with the corresponding plots of flow-NRU in Figure 4, we can gain some insight into the relative performance of Flow 1 and Flow 2. Firstly we see that, even though the plots of Figure 3 show peaks and troughs in the throughput for Flow 2 over the range of experiments, the flow-NRU for Flow 2 in Figure 4 is relatively flat for all plots, showing that the behaviour does tend to a stable an consistent behaviour, relative to what each Flow 2 might achieve against another flow of the same kind. However, the behaviour is not always fair.

In Figure 4(a), we see that the mean flow-NRU for Flow 2 is -0.36, i.e. close to zero, and the plot is reasonably flat. This indicates the two NewReno Flows do converge to a stable situation with each flow getting roughly a fair share.

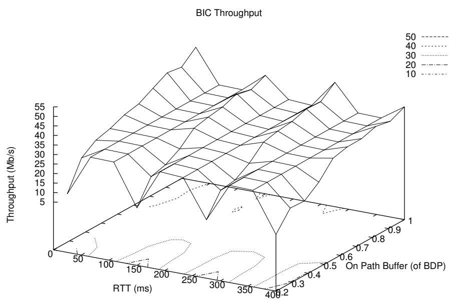
As might be expected from the throughput plots, we see some similarities between the behaviour of BIC, CUBIC and Compound TCP. Fairness with NewReno seems to improve as the RTT increases, in each of the cases BIC (Figure 4(b)), CUBIC (Figure 4(c)) and Compound TCP (Figure 4(d)). Interestingly, at low RTT values, the flow-NRU is negative, so the variant receives less than its fair share of capacity when running against NewReno, and this behaviour seems to be similar (if not exactly the same) across all OPB values. However, as the RTT increases to 50ms and beyond, we see that BIC, CUBIC and Compound TCP all consume more capacity than would be considered their 'fair share', based on the self-fairness experiments. As the mean flow-NRU values are greater than zero in each case, this means that BIC, CUBIC and Compound TCP manage to achieve greater throughput when running against NewReno than if they were running against Flow 1 of the same variant. To see if this is at the cost of NewReno, we need to look at the system-NRU, as shown in Figure 5[7].

If we consider first Figure 5(a), we see that the plot is flat and very close to zero for nearly all experiments. This means that, on average, the two NewReno flows are fair to each other in our experiments, then considering the system as a whole and the plot of Figure 4(a). This is as might be expected. Also, as we might expect form the other plots of Figure 4, we see that BIC, CUBIC and Compound TCP as Flow 2 are generally unfair to NewReno, on a system-wide basis.

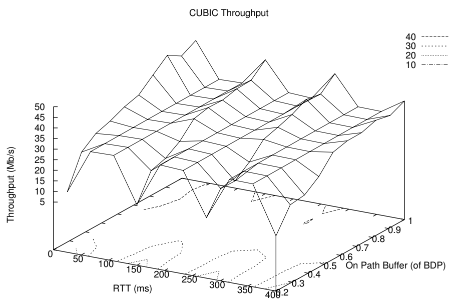
[7]We could also examine the flow-NRU for Flow 1, but we choose to use the system-NRU in this case.



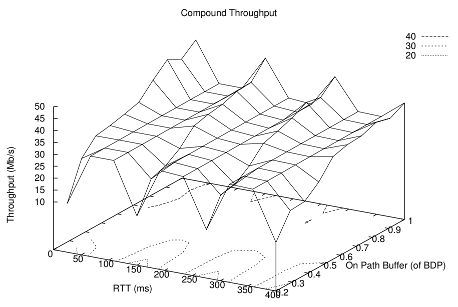
(a) NewReno (min 8.1, mean 25.7, max 49.4)



(b) BIC (min 8.6, mean 31.5, max 50.1)

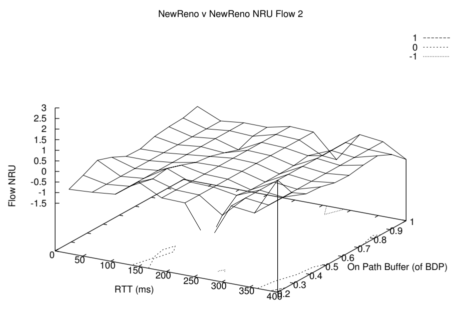


(c) CUBIC (min 8.6, mean 31.8, max 53.1)

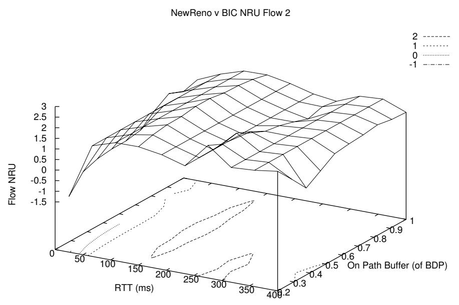


(d) Compound (min 7.1, mean 31.6, max 50.9)

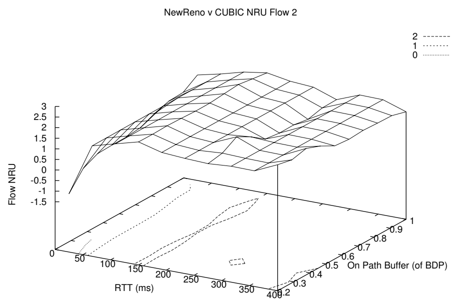
Figure 3. Throughput of Flow 2 for two flows of the same TCP variant.



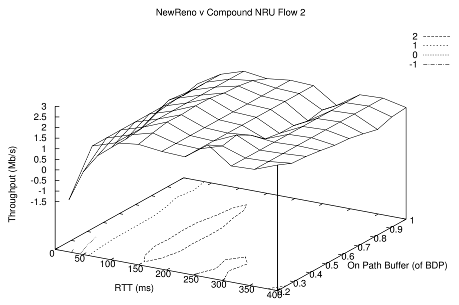
(a) NewReno (min -3.14, mean -0.36, max 1.55, stddev 0.75)



(b) BIC (min -1.50, mean 1.55, max 2.58, stddev 0.88)

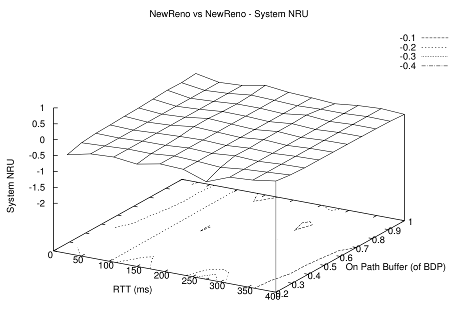


(c) CUBIC (min -1.00, mean 1.63, max 2.56, stddev 0.70)

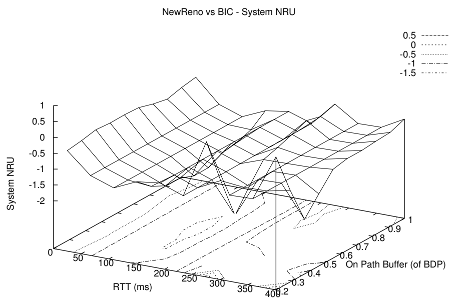


(d) Compound (min -1.28, mean 1.66, max 2.60, stddev 0.75)

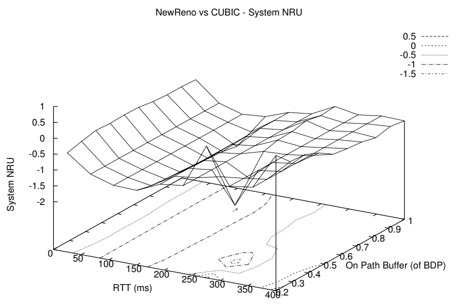
Figure 4. Flow-NRU, U_n for Flow 2, (Flow 1 is NewReno).



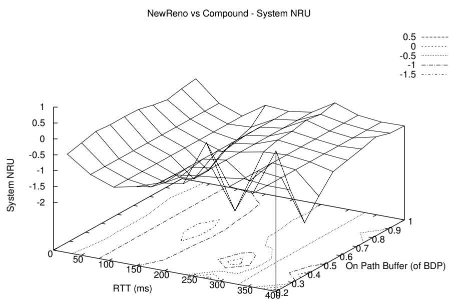
(a) NewReno (min -0.73, mean -0.16, max 0.18, stddev 0.14)



(b) BIC (min -2.91, mean -0.91, max 0.89, stddev 0.90)



(c) CUBIC (min -2.77, mean, -0.70, max 0.91, stddev 0.81)



(d) Compound (min -2.98, mean -0.74, max 0.99, stddev 0.86)

Figure 5. System-NRU, U_N.

5. Conclusion

Using a testbed, we have examined the relative performance of the TCP variants BIC, CUBIC and Compound TCP in pair-wise tests against TCP NewReno. We use our own metric, the normalised resource usage (NRU), which gives a per-flow assessment of the resource consumed by each flow, relative to its capability under the experimental conditions of our testbed.

We have found that, across a range of values of on-path buffering (OPB) and round-trip time (RTT) values, the TCP variants generally exhibit *unfairness* towards TCP NewReno. That is, they consume more of the capacity then they would against another flow of the same variant, at the cost of the TCP NewReno flow, which is forced into a behaviour resulting in it having a lower end-to-end throughput compared to running against another TCP NewReno flow.

However, this behaviour of unfairness is not consistent, and at low values of RTT (at 50ms and below), i.e., low values of bandwidth-delay product (BDP), we observe that the TCP variant flow gets less than its fair share. While this is not necessarily expected, the TCP variants all have a shared design goal, which is to be more effective specifically on paths with higher BDP.

We see that, in general, fairness improves as OPB values increase and RTT values increase, i.e., at higher BDPs. This is consistent with [14] which states that that when the output/input capacity is close to 1, and statistical multiplexing on a link is low (i.e. there are a lower number of flows), large amounts of buffering is beneficial for the performance of TCP.

5.1. Limitations and Future work

We state here, briefly, limitations of our study, which would all be suitable for investigation as future work, to expand on the experiments and analysis conducted here.

In this study, we have considered a wide range of OPB and RTT combinations. However, both flows in our experiment have shared a very similar end-to-end path, and this may not necessarily be the situation in all cases. However, certainly in environments with mixed operating systems (such as academic and corporate site networks), this may be the case at least for part of the network path.

Although we have staggered the starting of the flows to try and alleviate synchronisation effects between them, we cannot guarantee that the flows do not synchronise. Additionally, we have observed in previous similar experiments [3] that considering the flow under questions as Flow 1 (starting at $t = 0$) or as Flow 2 (starting at $t = 60$) could yield slightly different behaviour in pair-wise tests.

We have examined only flows in competition for a bottleneck link. The end-to-end behaviour observed may not be the same when considering the statistical multiplexing effect that results from many flows sharing the same link. Large-scale experiments may be impractical: we would need to investigate if simulation, e.g. using *ns2* [8], would yield accurate results [2].

We have not modelled loss or error behaviour for the network in our study. The TCP variants are designed for operation in high BDP paths, most of which are likely to have low error rates, e.g. fibre links, but may experience congestion loss that is not as simple as modelled in our testbed with a single bottleneck. Additionally, some high BDP paths may have high error rates, e.g., satellite links.

Like other studies, e.g., [4, 5, 11], we have used only two flows and assumed that both flows are like file-transfers: highly asymmetric in data-flow, are relatively long-lived, and send a continuous stream of packets. However, short-lived flows (e.g. WWW access and other interactive services) could produce different results [14].

References

[1] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4):281–292, 2004.

[2] M. Bateman, S. Bhatti, G. Bigwood, D. Rehunathan, C. Allison, T. Henderson, and D. Miras. A comparison of TCP behaviour at high speeds using ns-2 and Linux. In *Proc. 11th Communications and Networking Simulation Symposium (CNS'08)*, Ottawa, Canada, April 2008.

[3] S. Bhatti, M. Bateman, and D. Miras. A Comparative Performance Evaluation of DCCP. In *Proc. Int. Sym. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS2008)*, June 2008.

[4] S. Bhatti, M. Bateman, D. Rehunathan, T. Henderson, G. Bigwood, and D. Miras. Revisiting inter-flow fairness. In *Proc. BROADNETS2008 - 5th International Conference on Broadband Communications, Networks and Systems*, London, UK, September 2008.

[5] H. Bullot, R. L. Cottrell, and R. Hughes-Jones. Evaluation of advanced TCP stacks on fast long-distance production networks. In *First International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet03)*, Feb 2003.

[6] S. Floyd. Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166, Mar. 2008.

[7] S. Floyd and E. Kohler. Internet research needs better models. In *Proc. HotNets–I*, October 2002.

[8] Y. Ganjali and N. McKeown. Update on buffer sizing in Internet routers. *SIGCOMM Comput. Commun. Rev.*, 36(5):67–70, 2006.

[9] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu. A step toward realistic performance evaluation of high-speed TCP variants. In *Fourth International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet06)*, Feb 2006.

[10] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301, Sep 1984.

[11] Y.-T. Li, D. Leith, and R. N. Shorten. Experimental Evaluation of TCP Protocols for High-Speed Networks. *IEEE/ACM Transactions on Networking*, 15(5):1109–1122, Oct 2007.

[12] D. Miras, M. Bateman, and S. Bhatti. Fairness of High-Speed TCP Stacks. In *22nd IEEE International Conference on Advanced Information Networking and Applications (AINA 2008)*, Japan, March 2008.

[13] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *SIGCOMM Comput. Commun. Rev.*, 28(4):303–314, 1998.

[14] R. S. Prasad, C. Dovrolis, and M. Thottan. Router buffer sizing revisited: the role of the output/input capacity ratio. In *Proc ACM CoNEXT 2007*, pages 1–12, New York, NY, USA, 2007. ACM.

[15] G. Raina and D. Wischik. Buffer sizes for large multiplexers: TCP queueing theory and instability analysis. In *Proc. EuroNGI 2005*, April 2005.

[16] R. N. Shorten and D. J. Leith. H-TCP: TCP for high-speed and long-distance networks. In *Fourth International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet06)*, Feb 2006.

[17] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP approach for high-speed and long distance networks. In *Proc. IEEE INFOCOM 2006*, Spain, April 2006.

[18] C. Villamizar and C. Song. High performance TCP in ANSNET. *SIGCOMM Comput. Commun. Rev.*, 24(5):45–60, 1994.

[19] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, and R. N. Shorten. A critique of recently proposed buffer-sizing strategies. *SIGCOMM Comput. Commun. Rev.*, 37(1):43–48, 2007.

[20] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast long-distance networks. In *Proc. IEEE INFOCOM 2004*, Mar 2004.

[21] L. Xu and I. Rhee. CUBIC: A new TCP-Friendly high-speed TCP variant. In *Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet06)*, Feb 2005.

[8] `http://www.isi.edu/nsnam/ns/`