Fairness of High-Speed TCP Stacks

Dimitrios Miras
Department of Computer Science
University College London (UCL)
London, WC1E 6BT, UK
d.miras@cs.ucl.ac.uk

Martin Bateman, Saleem Bhatti
School of Computer Science
University of St Andrews
St Andrews, KY16 9SX, UK
{mb,saleem}@cs.st-andrews.ac.uk

Abstract

We present experimental results evaluating fairness of several proposals to change the TCP congestion control algorithm, in support of operation on high bandwidth-delay-product (BDP) network paths. We examine and compare the fairness of New Reno TCP, BIC, Cubic, Hamilton-TCP, Highspeed-TCP and Scalable-TCP. We focus on four different views of fairness: TCP-friendliness, RTT-fairness, intra- and inter-protocol fairness.

1. Introduction

The ability of the congestion control algorithm of the Transmission Control Protocol (TCP) to react to periods of resource overloading, is key to the Internet's stability. However, the appearance of high-speed connectivity to the access and site networks, coupled with bandwidth-hungry applications (e.g. high data volume scientific computation) and high-performance workstations results in a desire to consume large amounts of capacity. TCP, however, is known to behave poorly in long-distance, high-speed networks – using paths with large bandwidth-delay-product (BDP) values – due to the slow growth of its congestion window, leading to significant under-utilisation of the path capacity.

To alleviate this problem, proposals for changes to the TCP congestion control have been suggested: BIC TCP [11], Cubic [12] TCP, Hamilton TCP [10], Highspeed-TCP [3], Scalable-TCP [8], and others. These so-called *high-speed TCP variants*, have attracted considerable interest from the research community to evaluate their performance, identify the need for modifications, and establish whether they are suitable for wider use on the Internet.

Such performance evaluation is not an easy task. Congestion control algorithms are particularly sensitive to a range of factors, including (but not limited to): the path bottleneck capacity and bottleneck queue size; the round-trip time (RTT) experienced by a flow; and the pattern and level of other traffic that concurrently shares the same link(s). Also of significance is the behaviour due to end-system specifics (kernel parameters, scheduling, socket send/receive buffers, interface send/receive buffers, the specific configuration of the network interface driver, etc.), which may have more profound effects than the aforementioned network-centric parameters.

In our work we use an experimental testbed that attempts to control as many of the salient end-system factors as possible in order to evaluate the following set of high-speed TCP variants: BIC TCP, Cubic TCP, Hamilton TCP (HTCP), Highspeed-TCP, and Scalable-TCP. Of course, we do not claim that this is a completely realistic environment but we believe it is sufficient to determine areas where these protocols exhibit beneficial behaviour as well as behaviour that may have a negative impact. The specific selection of protocols to examine was made using the following criteria: (i) protocols that have attracted considerable interest by the research community and (ii) are readily available for use (and so could be used) in the latest Linux kernels. In this paper, we exclusively focus on the *fairness* properties of high-speed protocols, in order to obtain a thorough understanding on how well these protocols are capable of sharing a common capacity resource (a shared bottleneck), and what properties they exhibit when they co-exist on a shared network path. This is important behaviour to understand, as many of the TCP variants that are available in the Linux kernel are being used across private networks and across the Internet.

The fairness criterion is investigated by considering four aspects of flow interaction, as follows:

- *TCP-Friendliness*: evaluates the fairness that a new high-speed TCP variant exhibits towards 'standard' TCP (New Reno) flows.
- *RTT Fairness*: evaluates the fairness of TCP flows which share the same bottleneck but traverse network paths with different round-trip times.

- *Intra-protocol Fairness*: evaluates how flows of the *same* TCP variant interact with each other.
- *Inter-protocol Fairness*: evaluates how flows of *different* TCP variants interact with each other.

2. High-speed TCP variants

This section presents a short overview of the congestion control definitions of each of the high-speed protocols evaluated in this work. In order to devote more space to our results, we assume the reader is familiar with the basic principle of congestion control in TCP – the notion of the *congestion window (cwnd)*, and the Additive Increase Multiplicative Decrease (AIMD) behaviour [6]:

$$\begin{aligned} OnACK : cwnd &\leftarrow cwnd + \alpha \\ OnLoss : cwnd &\leftarrow \beta.cwnd \end{aligned}$$

with $\alpha = 1$ and $\beta = 0.5$.

2.1. BIC TCP

Binary Increase Congestion control TCP – *BIC* TCP [11] – uses a binary search algorithm between the window size just before a reduction (W_{max}) and the window size after the reduction (W_{min}). If w_1 is the midpoint between W_{min} and W_{max}, then the window is rapidly increased when it is less than a specified distance S_{max} from w_1 and grows more slowly when it is near w_1. If the distance between the minimum window and the midpoint is more then S_{max}, the window is increased by S_{max}, following a linear increase. BIC reduces its window by a multiplicative factor β. If no loss occurs, the updated window size becomes the current minimum, otherwise, the window size becomes the new maximum. If the window grows beyond the current maximum, then it employs an exponential and then a linear increase to probe faster for the new equilibrium window size.

2.2. Cubic TCP

Cubic TCP [12] uses a cubic function to control its congestion window growth. If W_{max} is the congestion window before a loss event, then after a window reduction, the window grows fast and then slows down as it approaches W_{max}. Around W_{max}, the window grows slowly, again accelerating as it moves away from W_{max}. The following formula determines the congestion window size (*cwnd*):

$$cwnd = C(T-K)^3 + W_{max}$$

where C is a scaling constant, T is the time since the last loss event and $K = \sqrt[3]{W_{max}\frac{\beta}{C}}$, where β is the multiplicative decrease factor after a loss event. C and β are set to 0.4 and 0.2 respectively. To increase fairness and stability, the window is clamped to grow linearly when it is far from W_{max}.

2.3. Hamilton TCP (HTCP)

Hamilton-TCP (HTCP) [10] modulates the normal AIMD parameters as a function of the elapsed time Δ since the last congestion event.

$$\begin{aligned} OnACK : cwnd &\leftarrow cwnd + \frac{2(1-\beta)f_\alpha(\Delta)}{cwnd} \\ OnLoss : cwnd &\leftarrow g_B(B).cwnd \end{aligned}$$

where:

$$f_\alpha(\Delta) = \begin{cases} 1 & \Delta \le \Delta_L \\ max(\bar{f}(\Delta)T_{min}, 1) & \Delta > \Delta_L \end{cases}$$

$$g_B(B) = \begin{cases} 0.5 & |\frac{B(k+1)-B(k)}{B(k)}| > \Delta_B \\ min(\frac{T_{min}}{T_{max}}, 0.8) & otherwise \end{cases}$$

where, Δ_L is a threshold such that the standard TCP algorithm is applied if $\Delta \le \Delta_L$, and a quadratic function f_α is used: $\bar{f}_\alpha = 1 + 10(\Delta - \Delta_L) + 0.25(\Delta - \Delta_L)^2$. T_{min} and T_{max} are, respectively, the minimum and maximum round-trip times seen by the flow, and $B(k+1)$ is the maximum bandwidth measurement during the last congestion epoch.

2.4. Highspeed-TCP

Highspeed-TCP [3] modulates its increase and decrease parameters as a function of the current congestion window (*cwnd*). The higher the current value of *cwnd*, the higher the additive increase for a lossless RTT time interval. Also smaller multiplicative decrease coefficient is used in response to a loss event.

$$\begin{aligned} OnACK : cwnd &\leftarrow cwnd + a(cwnd)/cwnd \\ OnLoss : cwnd &\leftarrow b(cwnd).cwnd \end{aligned}$$

Highspeed-TCP employs an increasing logarithmic function for $a(cwnd)$ and a decreasing logarithmic function for $b(cwnd)$. The AIMD parameters revert to the standard TCP values (1 and 0.5 respectively) when *cwnd* is below a low threshold.

2.5. Scalable-TCP

Scalable-TCP [8] tackles the problem of the long recovery time of standard TCP over large BDP links by making the process of updating its congestion window independent of the congestion window value at any time. The generalised Scalable-TCP modifies the congestion window as:

$$\begin{aligned} OnACK : cwnd &\leftarrow cwnd + \alpha \\ OnLoss : cwnd &\leftarrow \beta.cwnd \end{aligned}$$

where α and β are constants, with $\alpha, \beta \varepsilon (0,1)$ (the recommended values are $\alpha = 0.01$ and $\beta = 0.875$). Scalable-TCP reverts to the standard TCP congestion window update algorithm when its congestion window is smaller than a certain threshold, the *legacy congestion window* (*lcwnd*).

3. Experimental Setup

The network setup used in our high-speed TCP experiments is shown in Figure 1; although the dumb-bell topology is a simplified approach, it has the advantage of being widely used in similar scenarios, thus providing a known configuration for comparison of results. The testbed consisted of five identical Sun workstations with dual AMD Athlon64 2GHz CPUs, 2GB of memory and 1Gb/s PCI-X (133MHz) Intel PRO/1000 NICs, all running Linux 2.6.20. Two of these machines were running as senders, two as receivers and one as a network router, the latter providing the bottleneck between the senders and receivers. Linux kernel and ethernet controller parameters have been appropriately tuned for high-speed transfers (see Table 1). The TCP send and receive socket buffer sizes were also tuned to appropriate values (depending on the BDP) so that the TCP congestion window is not unduly constrained due to lack of buffer space at the hosting machine.

Table 1. Host kernel parameters (all tests)

Parameter	Value
txqueuelen	10000
net.core.netdev_max_backlog	2500
net.ipv4.tcp_timestamps net.ipv4.tcp_window_scaling net.ipv4.tcp_sack net.ipv4.tcp_no_metrics_save	enabled

In order to emulate properties of a wide area network (varying bandwidth, round-trip times and router buffer sizes), the *Netem* utility of Linux is used. Netem is an enhancement of Linux's traffic control (*tc*) facilities that allows the emulation of network delay, packet loss and other scenarios through buffer control mechanisms. Although the majority of related work has mainly used FreeBSD's *DummyNet* to configure such network path properties, NetEm is now a mature platform and is being actively used by many researchers for testing protocols and applications. We believe that presenting results from an alternative network emulation system is beneficial to the community, not only for cross-validating experimental results, but also to investigate whether the specific choice of emulation platform has an impact on the performance of the monitored TCP connections, and, therefore, the validity of such tests.

We consider a range of round-trip time (RTT) values, from low (16ms) to high values (324ms). Previous similar studies [9, 5] have considered several values of bottleneck

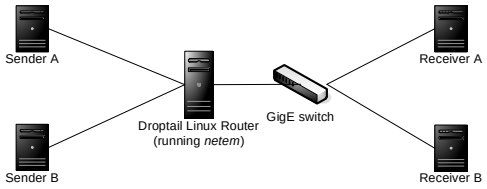


Figure 1. Experimental network testbed setup

capacity but only up to data rates of 400-500Mb/s. Our results fill the 'gap' in relevant experimental results by considering even higher values: in our emulation tests, the bottleneck capacity was set to 900Mb/s, a data rate which is now easily achievable by commodity server and workstation end systems. We choose two different configurations for the router's buffer sizing: one with a constant size of 3MB and one where the buffer size is proportional to the bandwidth delay product (BDP); in the latter case, the buffer size is set at 20% of the BDP.

In this work, we consider high-speed TCP congestion algorithms that are readily available in the latest Linux kernels: *BIC*, *Cubic*, *HTCP* (Hamilton), *Highspeed*[1] and *Scalable* TCP. The standard Linux TCP implementation of New Reno was used as the 'standard' TCP. TCP flows were generated using a modified version of the *iperf*[2] utility to allow for easy selection of congestion control algorithms and to record the values of various TCP variables (*cnwd*, *ssthresh*, *rtt*). Each individual test is run for 10mins; to account for variability among different runs, each test is run at least five times and the mean value of throughput and fairness over all runs is calculated.

In the experiments, the first *iperf* TCP flow starts at time 0s and the second at time 60s.

4. Experimental results

We conducted a set of experiments where the RTT for two TCP flows (from the 6 variants listed above) was set to 16, 42, 82, 162 and 324ms. We emulated scenarios where the two TCP flows have the same and then different end-to-end delays. We conducted transmissions in the absence of any background traffic, to establish a 'baseline' performance[3].

The following section presents results on fairness, including *TCP-friendliness*, *intra-protocol fairness* and *RTT-fairness*. We also give attention to *inter-protocol fairness* (the fairness between different instances of the high-speed

[1] The Linux kernels used implement the limited slow-start option [4] for Hisghspeed-TCP connections with large congestion windows; this limits the large increase of the congestion window during the slow-start phase.

[2] Available on request from the authors.

[3] We also have results conducted using synthetic background traffic.

TCP variants), something that has not been widely reported in related work. Our experiments produced a significant volume of data and a large number of graphs – we report here the most important findings.

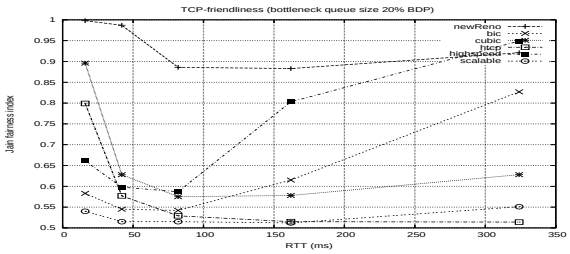
4.1. TCP Friendliness

TCP-friendliness measurements were produced by running a 'standard' (New Reno) TCP flow and one of each of the high-speed variants over the same bottleneck link. Both flows experience the same RTT. We measure the fairness between normal TCP and a high-speed variant by calculating Jain's fairness index [7]:

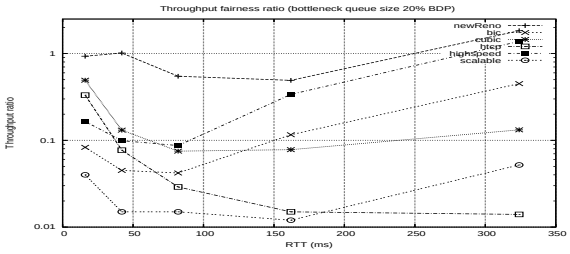
$$F(b) = \frac{(\sum b_n)^2}{(N \sum b_n^2)}$$

where b_n is the bit-rate measured for flow n, $n = 1 \ldots N$. However, Jain's fairness index does not show how aggressive a protocol instance is towards other protocol instances, only how 'fairly' the link's bandwidth is shared across the complete duration of the flow overall. We also measure the *ratio of average throughput* as an indicator of how gentle or aggressive a protocol is towards other cross-traffic. A ratio value near one indicates that both protocol instances are fair to each other, while the closer this value is to zero, the more aggressive the high-speed variant is towards the competing standard TCP flow. TCP friendliness fairness indices for the high-speed TCP variants in relation to varying RTT values are shown in Figure 2, for two configurations of router queue size (3MB and 20% of BDP) with no background traffic.

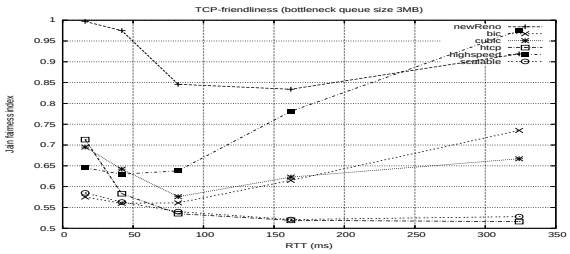
From our tests, it appears that *Cubic* and *HTCP* exhibit better fairness with standard TCP at very low RTT (Figure 2). For higher RTT networks, almost all high-speed variants display similar unfairness towards standard TCP (*Scalable* TCP seems to consistently be the most aggressive towards standard TCP traffic, closely followed by *HTCP*). This is to be expected, since it is in such high BDP paths that the inability of TCP to quickly utilise available capacity after a congestion event becomes more evident. However, it appears that *Highspeed-TCP* exhibits superior TCP-friendliness at medium to high RTTs (RTT $>$ 42ms) in comparison to the other high-speed variants: we attribute this to the adoption of the limited slow-start option [4] in the algorithm's implementation in the latest Linux kernels.



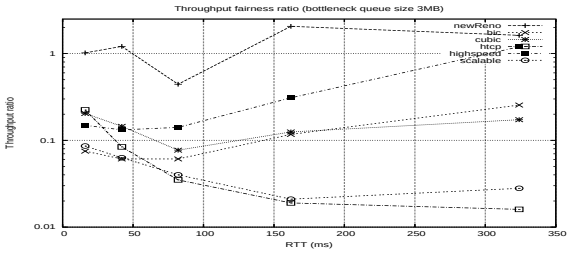
(a) Intra-protocol fairness, bottleneck queue 20% BDP



(b) Throughput fairness, bottleneck queue 20% BDP



(c) Intra-protocol fairness, bottleneck queue 3MB

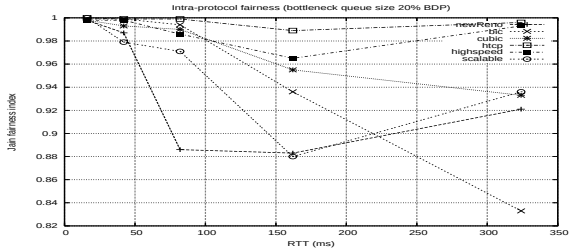


(d) Throughput fairness, bottleneck queue 3MB

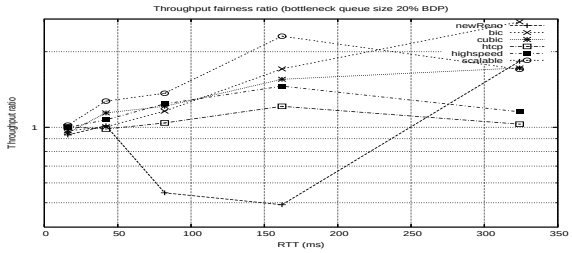
Figure 2. TCP-friendliness

4.2. Intra-protocol Fairness

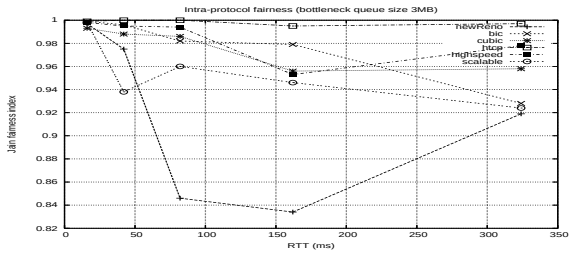
Intra-protocol fairness is measured in a similar fashion to TCP friendliness, by considering two flows of the *same* TCP variant with the same RTT (varied between 16 and 324 ms). Intra-protocol fairness is shown in Figure 3. Subfigures (a) and (c) show Jain's fairness index of the system, while (those)b) and (d) show (on a logarithmic y-axis) the ratio of throughput of the two competing TCP flows. Jain's fairness index for standard TCP tests is also plotted as a reference. From these graphs, we observe that all high-speed TCP variants exhibit good intra-fairness properties, with fairness indices over 0.9 in the majority of cases. *HTCP* in particular, has a consistently high fairness index at all RTTs, closely followed by *Highspeed-TCP*.



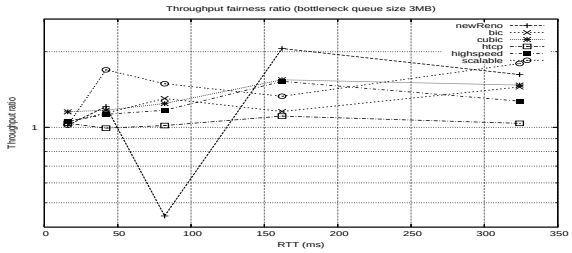
(a) Intra-protocol fairness, bottleneck queue 20% BDP



(b) Throughput fairness ratio, bottleneck queue 20% BDP



(c) Intra-protocol fairness, bottleneck queue 3MB



(d) Throughput fairness ratio, bottleneck queue 3MB

Figure 3. Intra-protocol fairness

4.3. RTT Fairness

To define what constitutes 'fair' sharing of a bottleneck resource among flows that traverse end-to-end paths of various lengths (distance, number of intermediate routers, etc.) is not straightforward. To desire that flows with large RTTs share equally a resource even though they utilise more network resources on their end-to-end path than other flows with shorter RTT may not always be considered a pragmatic approach. An alternative approach is to have 'proportional' fairness, where flows share bandwidth in a manner that is inversely proportional to their RTT, and to also require that flows with large RTTs are not starved of bandwidth.

We conducted experiments to test the pattern of sharing the bottleneck bandwidth between two competing flows with different RTTs. Two flows of the same TCP variant are tested: the RTT of the first flow is permanently set at 162ms, while the RTT of the second flow varies from 16 to 324 ms. The first flow's RTT of 162ms[4] was chosen because it is half of the highest RTT and therefore reports the behaviour at the midpoint of our RTT range. The results are presented in Figure 4. The RTT-fairness of standard TCP is also plotted as a reference.

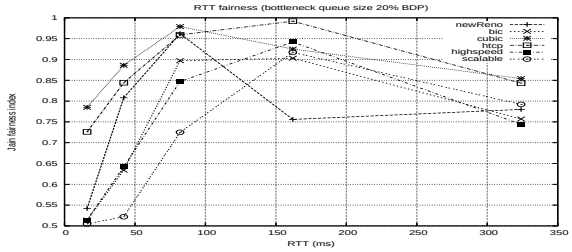
The sub-figures (graphs on the left in)a) and (c) Figure 4 show that *HTCP* and *Cubic* have a consistently superior Jain's fairness index than the other TCP variants at all RTT values and at both the bottleneck router queue size configurations (3MB and 20% of the BDP). The RTT-fairness indices of *BIC* and *Highspeed-TCP* are much lower, especially at lower RTTs. However, this should not be seen as a drawback of these protocols: it is not agreed that an equal bandwidth sharing among flows with different end-to-end delays is a desirable proposition. Finally, in our tests, *Scalable-TCP* consistently had the lower fairness index. These observations hold for all tested cases of router queue size.

We turn our attention to sub-figures (b) and (d) of Figure 4 which plots the throughput ratio of the two competing flows as the RTT of the second flow varies from 16 to 324 ms (reminder: that the RTT of the first flow is fixed at 162ms). In addition, the RTT ratio between the two competing flows is plotted. This line represents a throughput ratio proportional to the RTT ratio with unit gradient. Some interesting observations can be deduced from these plots:

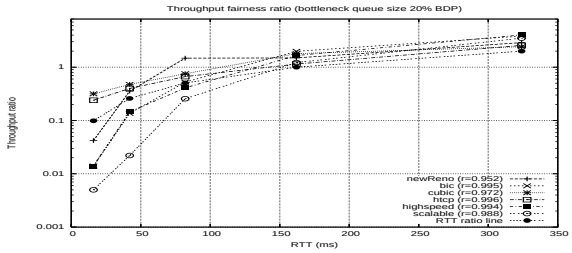
- All TCP variants share the bandwidth in an 'RTT-proportional' manner; this is evident by the high correlation between the throughput and RTT ratio values (the corresponding linear correlation coefficient r for each TCP variants is also shown in these graphs); the difference among different flows of the same protocol is shown in the gradient of their linear relationship with their corresponding RTT ratio.

- When the RTT of the second flow is smaller than that of first flow, the throughput ratios for *Cubic* and *HTCP* are above the 'reference' RTT ratio line; this means that instances of these protocols with longer RTTs yield, on average, higher bandwidth shares in the presence of flows with smaller RTT; this confirms their higher RTT fairness, as mentioned above. The rest of the high-speed TCP variants have a smaller slope than the 'reference' RTT ratio line, which means that

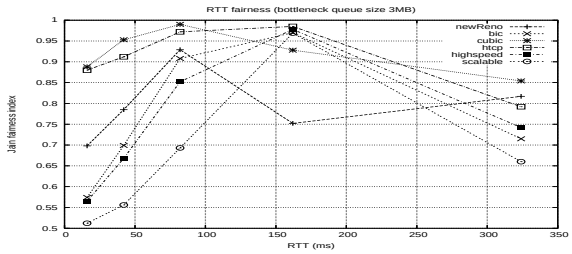
[4]This happens to be approximately the RTT, as measured by ping, from the University of St Andrews to the central states of the USA



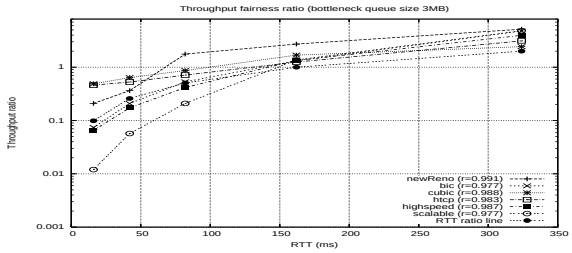
(a) RTT fairness, bottleneck queue 20% BDP



(b) Throughput fairness ratio, bottleneck queue 20% BDP



(c) RTT fairness, bottleneck queue 3MB



(d) Throughput fairness ratio, bottleneck queue 3MB

Figure 4. RTT fairness

on average, higher-RTT flows are permitted less – in comparison – of the bandwidth share.

- In agreement with the fairness measurement results above, the TCP throughput ratio graphs show that *Scalable-TCP* flows with lower RTTs are more aggressive towards higher RTT flows in comparison with the other high-speed variants that we tested.

4.4. Inter-protocol Friendliness

In this section we present results on the fairness properties when different instances of high-speed TCP variants coexist on the same network and, specifically, how aggressive (or not) an instance of a high-speed variant is in the presence of other high-speed flows. This aspect of fairness has not been examined as thoroughly in recent studies, but we consider this especially important: it is possible that in the near future, there will not be one TCP algorithm which has dominant deployment, rather, various TCP flavours will coexist and concurrently share bandwidth. A desirable property would be that a protocol would be neither too aggressive nor too gentle towards other protocol instances. Of course, the particulars of this proposition can not be asserted in the absence of a well-agreed definition of fairness, but the adoption of 'fuzzy' labels such as 'not-aggressive' or 'not-gentle' currently suffice within the community to give meaningful descriptions, coupled with measurements from which conclusions can be drawn.

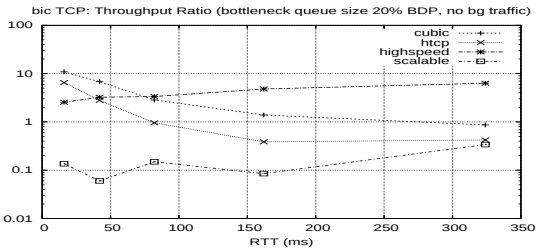
To measure inter-protocol fairness, two high-speed TCP flows are transmitted from two different machines. We measure the ratio of average throughput obtained between the two competing protocol flows over the duration of the experiment (600 seconds). In contrast to Jain's fairness index, this metric reveals how aggressive or not is a flow in comparison to its competitor. Figures 5–9 plot this metric over different RTT values for all five high-speed variants considered in our experiments. For each TCP variant, four graphs are shown for the four combinations of router queue size.

Some interesting observations can be deduced from the inter-protocol fairness graphs. *BIC* TCP (Figure 5) tended to be more aggressive to all other protocols (except *Scalable-TCP*) at low RTT values. At higher RTTs, *BIC* still received a higher share of bandwidth than *Highspeed-TCP*, but shared bandwidth equally with *Cubic*, and became more gentle towards *HTCP*. *BIC* consistently achieved lower bandwidth than *Scalable-TCP* in any of the stated configurations of the tested.

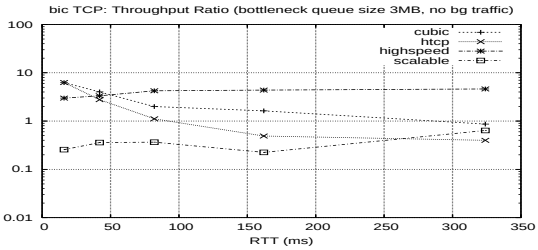
The graphs in Figure 6 show that *Cubic* TCP tended to be gentle (i.e. yielded lower throughputs) towards all TCP variants at low RTTs, but tended to exhibit better inter-fairness at higher RTT values: in particular, *Cubic* is relatively more aggressive towards *Highspeed-TCP* and *BIC*.

Figure 7 shows that *HTCP* is capable of grabbing more bandwidth as the end-to-end network delay increases, and is therefore quite aggressive towards *Highspeed-TCP* flows, and to a lesser degree, towards *BIC* and *Cubic* flows. Again, higher statistical multiplexing improves the fairness amongst competing flows.

Highspeed-TCP (Figure 8) was consistently gentle towards the other high-speed protocols, as it obtained a considerably lower share of the bottleneck bandwidth (with the

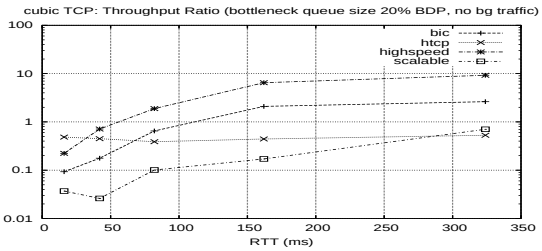


(a) Throughput ratio, bottleneck queue 20% BDP

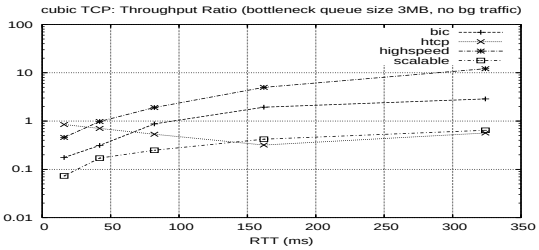


(b) Throughput ratio, bottleneck queue 3MB

Figure 5. Inter-protocol fairness: *BIC*



(a) Throughput ratio, bottleneck queue 20% BDP

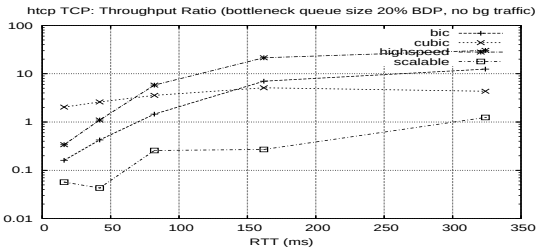


(b) Throughput ratio, bottleneck queue 3MB

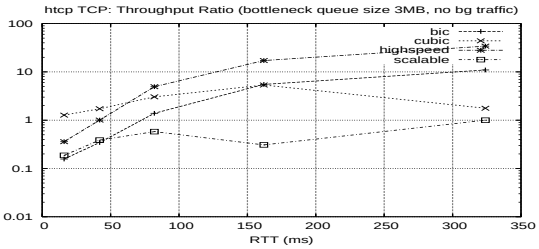
Figure 6. Inter-protocol fairness: *Cubic*

exception of *Cubic* and *HTCP* at very low RTTs). This is attributed to its adoption of the limited slow start option, as discussed in Section 3 above. Consistent to our previous observations, *Scalable-TCP* showed the greatest aggressiveness towards *Highspeed-TCP*. At high RTTs, *HTCP* is equally aggressive. *Scalable-TCP* was shown, from our experiments, to exhibit the highest aggressiveness in occupying more of the bandwidth share.

Finally, and in accordance with our findings in the previous sections, Figure 9 shows that *Scalable TCP* is the most aggressive protocol, at times obtaining at least an order of magnitude (and higher) more bandwidth in comparison to competing flows of other high-speed variants. This was more evident at lower RTTs. In this case, loss events are the results of the interaction between the two competing flows. *Scalable-TCP's* response curve dictates that the congestion window can grow quickly to a large value when the loss rate is kept low, thus outperforming other TCP protocols. At higher RTTs, fairness of *Scalable-TCP* is also improved, and with the exception of *Highspeed-TCP*, it performs fairly with other protocols.



(a) Throughput ratio, bottleneck queue 20% BDP



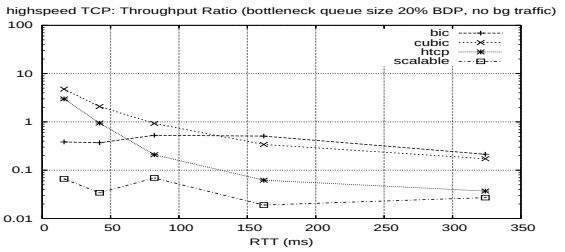
(b) Throughput ratio, bottleneck queue 3MB

Figure 7. Inter-protocol fairness: *HTCP*

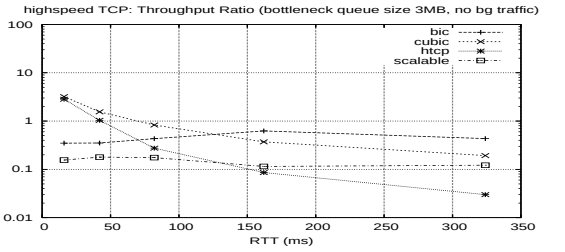
5. Related Work

Performance measurement of high-speed TCP networking has gained considerable interest recently as the limitations of standard TCP have been exposed on high BDP networks and new congestion control algorithms have emerged. Bullot et al [1] compared the performance of several TCP variants over three production networks with different RTTs. They confirmed the poor performance of standard TCP in those environments and reported good intra-fairness properties for high-speed variants and varying inter-protocol fairness (i.e. *Scalable-TCP* was very aggressive, and *Highspeed-TCP* relatively gentle). They used a simplistic sinusoidal UDP model of background traffic.

Researchers at the Hamilton Institute [2, 9] have performed performance experiments for standard TCP, *BIC*, *Highspeed-TCP*, *HTCP* (Hamilton) and *FAST* TCP (not *Cubic*) on a network emulation testbed using *DummyNet*. They

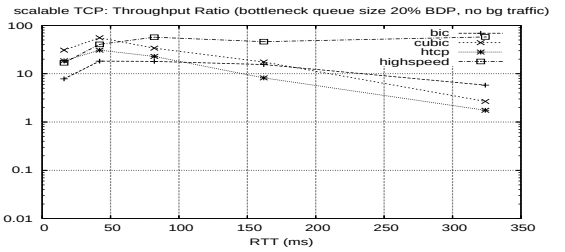


(a) Throughput ratio, bottleneck queue 20% BDP

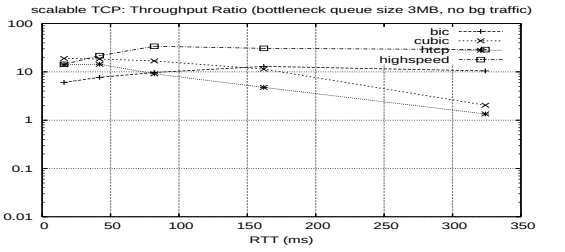


(b) Throughput ratio, bottleneck queue 3MB

Figure 8. Inter-protocol fairness: *Highspeed*



(a) Throughput ratio, bottleneck queue 20% BDP



(b) Throughput ratio, bottleneck queue 3MB

Figure 9. Inter-protocol fairness: *Scalable*

used network configuration similar to ours, but they experimented with bottleneck capacities ranging from 1Mb/s–250Mb/s, i.e. well below 1Gb/s. The most interesting result is that they found recent high-speed proposals to perform poorly in terms of fairness, which is contrary to our findings. No inter-protocol fairness results were reported.

Similar evaluation experiments were presented in [5]; results were produced using network emulation on a 400Mb/s bottleneck link with a 2MB queue size. They too reported good intra-protocol fairness properties, and lower fairness indices for *Scalable-TCP* and *Highspeed-TCP* during experiments without background traffic. Their RTT-fairness results broadly agree with ours, although there are some disparities, especially at lower RTTs. This can be explained by the use of different bottleneck capacity and queue size in their experimental testbed. Their work does not report on inter-protocol fairness.

6. Conclusion

In this paper, we presented experimental results to evaluate the fairness properties of emerging TCP protocols designed for high-speed IP networking. We presented a comprehensive set of results, based on network emulation on a testbed, that assess four aspects of protocol fairness: friendliness to standard TCP; intra- and inter-protocol fairness; and fairness between flows with heterogeneous round-trip times. Our experiments considered several combinations of round-trip times, bottleneck queue size and transmissions in the absence of background traffic. We do not claim that our experiments are exhaustive rather that they complement previous studies. However, we believe that we have highlighted the fairness properties of these protocols. In comparison to some of the recent relevant published work, we considered a high bottleneck capacity value (900Mb/s).

Our results show that the high-speed TCP protocols can be significantly unfriendly to existing standard TCP traffic, especially as the bandwidth-delay-product (BDP) increases. This is, however, derived from the requirements for these protocols to have been designed and implemented – to alleviate the low utilisation exhibited by standard TCP on high BDP paths. In such high BDP networks and with high utilisation in mind, it is arguable whether TCP-unfriendliness is indeed a negative characteristic. The addition of the limited slow-start option of *Highspeed* TCP in the recent Linux kernel does yield a significant improvement in TCP-friendliness, with respect to earlier incarnations in the Linux kernel studied in [5, 2, 9].

In contrast to [2, 9] but in agreement with [5], our experiments showed that all high-speed variants exhibit good intra-protocol fairness, even under the somewhat artificial conditions of no-background traffic. In our experiments, *HTCP* obtained the higher Jain's fairness index values at all tested configurations, closely matched by the other high-speed variants. Our results have also shown that the throughput between flows with heterogeneous round-trip times is indeed proportional to the ratio of their corresponding RTTs, albeit with different linear correlation. Specifically, *HTCP* and *Cubic TCP* showed higher RTT-fairness; on the other hand, *Scalable-TCP*, *Highspeed-TCP* and *BIC TCP* tend to favour flows with smaller RTTs. The examined high-speed protocols exhibited different inter-protocol fair-

ness, with *Scalable-TCP* being the most aggressive towards other protocols, followed by *HTCP*. On the other hand, *Highspeed-TCP* was the most gentle towards the other high-speed protocols examined.

We find that in intra-protocol tests there is a trend to increased fairness with increased RTT (Figure 2). With our RTT tests (Figure 4), we find that the best fairness is when all flows have the same RTT and fairness is lower for when the tested variant (second flow) has low or high RTT compares to the reference flow (first flow, RTT 162ms). However, no general summary of fairness behaviour can be tabulated.

References

[1] 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.

[2] B. Even, Y. Li, and D. Leith. Evaluating the performance of TCP stacks for high-speed networks. In *Fourth International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet06)*, Feb 2006.

[3] S. Floyd. Highspeed TCP for large congestion windows. RFC 3649, Dec 2003.

[4] S. Floyd. Limited slow-start for TCP with large congestion windows. RFC 3742, Mar 2004.

[5] 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.

[6] V. Jacobson and M. Karels. Congestion avoidance and control. In *ACM SIGCOMM*, pages 314–329, Aug 1988.

[7] 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.

[8] T. Kelly. Scalable TCP: Improving performance in high-speed wide area networks. *SIGCOMM Computer Communication Review*, 33(2):83–91, 2003.

[9] Y.-T. Li, D. Leith, and R. Shorten. Experimental evaluation of TCP protocols for high-speed networks. Hamilton Institute. To appear in Transactions on Networking.

[10] 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.

[11] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast long-distance networks. In *IEEE INFOCOM*, volume 4, pages 2514–2524, Mar 2004.

[12] 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.