A rough comparison of NewReno, CUBIC, Vegas and ‘CAIA Delay Gradient’ TCP (v0.1)

Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 110729A
Swinburne University of Technology
Melbourne, Australia
garmitage@swin.edu.au

***Abstract*—This report presents a preliminary, non-exhaustive experimental comparison of the relative latency induced by NewReno, CUBIC, Vegas and ‘CAIA Delay-Gradient’ (CDG) TCP algorithms under FreeBSD when run over a variety of home network links. While inducing less additional latency, CDG generally performs as well as NewReno and CUBIC over ADSL1, 802.11g WiFi and HomePlug AV links. All tests utilise FreeBSD-CURRENT as of May 2011.**

***Index Terms*—TCP, NewReno, CUBIC, Vegas, Delay-Gradient**

I. Introduction

Loss-based TCP congestion control (CC) algorithms (such as NewReno [1] and CUBIC [2]) tend to cause cyclical filling and draining of queues along network paths[1], adding to the latency experienced by *all* traffic sharing the path [3], [4]. This is particularly problematic given the growth of arguably-gratuitous buffering in all sorts of network devices, interface cards and network software stacks[2].

In contrast, delay-based TCP algorithms tend to minimise queuing delays induced at congestion points along a path (because they monitor delay measurements rather than packet loss to infer the onset of network congestion). This makes delay-based TCP an attractive proposition where real-time application flows (such as VoIP and online games) must share congestion points with TCP flows. However, delay-based algorithms have often been seen to perform worse than loss-based algorithms for straightforward data transfer.

As part of CAIA's NewTCP project [5] we developed a new, delay-based variant of TCP known as “CAIA Delay-Gradient” (CDG[3]) [6], [7], and implemented it as a patch to FreeBSD-CURRENT (FreeBSD 9 at the time of development). Unlike loss-based TCP variants, CDG adjusts its flow control ‘window’ (and hence transmission rate) in response to variations in the round trip time (RTT) observed between source and destination.

With support from the FreeBSD Foundation in late 2010, versions of CUBIC and Vegas (an earlier delay-based TCP) are now available in FreeBSD-CURRENT (FreeBSD 9, February 2011) and FreeBSD 8-STABLE (May 2011). This enables easy comparison between the ‘official’ FreeBSD versions of NewReno, CUBIC and Vegas against CDG version 0.1.

The rest of this brief report is structured as follows. Section II begins by highlighting what this report does not achieve. Section III describes the testing methodology, and Section IV summarises results. The report concludes in Section V.

II. A major caveat

It is important to note what this report does ***not*** do.

TCP congestion control research will often exhaustively evaluate fairness, responsiveness and convergence of (and between) flows using different CC algorithms. That is not the goal of this particular technical report.

Instead, we focus on a simple (but not-uncommon) home use case – a single stream of data briefly traversing a consumer-grade network link for some tens of seconds. We evaluate the throughput and increase in RTT experienced with each TCP variant.

You could drive a truck through the holes in some of our assumptions and/or lack of rigour. The tests are simplistic, and intended primarily to motivate additional exploration of our CDG TCP variant. We hope someone does produce a more detailed followup piece of work.

[1]Filling until packet loss is induced, draining when the sender temporarily reduces their transmission window.

[2]Aka ‘buffer bloat’, http://gettys.wordpress.com/category/bufferbloat/

[3]We ran out of budget for a cool name.

Critical evaluation of CDG and other TCP variants under more complex heterogeneous and multi-flow scenarios is definitely a matter for future work.

III. Experimental Methodology

Three different scenarios are tested: a remote Internet site sending to a home connected by ADSL1, in-house host-to-host over a 'HomePlug AV' link and in-house host-to-host over 802.11g WiFi link.

A. Repeated downloads, alternating algorithms

In each scenario we execute an *rsync*[4] file transfer using one of the four TCP variants (NewReno, CUBIC, CDG and Vegas) in turn. This process is then repeated six times. Between each file transfer the path is left idle for between 5 and 7 seconds.

Each file transfer takes roughly 30 to 50 seconds, during which we measure the RTT over the link and how much data arrives at the destination host per unit time (throughput, roughly speaking). The RTT measurements reveal how much *collateral damage* would be caused to any delay-sensitive application flows (such as VoIP or online games) that might have been sharing the link [3]. The throughput gives us a sense of how regular end users might perceive the performance of each TCP variant.

Traffic at both source and destination hosts is captured using tcpdump. RTT over time is calculated by running SPP[5] across both tcpdump files. Throughput is calculated using tcpstat[6] on the destination host's tcpdump file (counting all packet received, including retransmissions). The source host was also pinged two or five times per second by the destination host during each file transfer, to validate SPP's more finely-grained RTT measurements.

B. Destination and source hosts

In all scenarios the destination host is a Core2Duo machine on my home LAN running PC-BSD 8.2 (based on FreeBSD 8.2-RELEASE). The source host is a snapshot[7] of 64-bit FreeBSD-CURRENT from May 2011, with its kernel rebuilt from source after applying the CDG version 0.1 patches.

For ease of deployment, we instantiated the source host under VirtualBox (rather than wiping and re-installing FreeBSD on physical machines). The guest OS (source host) network interface was bridged to the physical host's network interface in each case.

For the ADSL1 scenario the source ran inside VirtualBox 3.1.12 under 64-bit PC-BSD 8.2 on a HP Compaq 8000 desktop PC with 8G RAM. For the PowerlineAV and 802.11g scenarios the source ran inside VirtualBox 4.0.6 under 64-bit Windows 7 Enterprise on a Toshiba R700 laptop with 8G RAM.

To ensure both PowerlineAV and 802.11g would be bottlenecks we confirmed that across a wired 1Gbps ethernet link the R700 laptop's source host could push over 100Mbit/sec to our destination host using any of the four CC algorithms.

C. Selecting source CC algorithms

FreeBSD-CURRENT allows new congestion control modules to be added, selected or de-selected on-the-fly. NewReno is always available immediately after rebooting:

```
# sysctl net.inet.tcp.cc
net.inet.tcp.cc.available: newreno
net.inet.tcp.cc.algorithm: newreno
#
```

Add additional dynamically-loadable CC modules:

```
# kldload cccubic
# kldload ccvegas
# kldload cccdg
```

And now the kernel reports multiple available choices:

```
# sysctl net.inet.tcp.cc
net.inet.tcp.cc.available: newreno, cubic,
vegas, cdg
net.inet.tcp.cc.algorithm: newreno
#
```

NewReno is the default choice. We can easily change the active CC algorithm to another of the available algorithms using the sysctl command. For example, the following makes all future TCP connections use CUBIC:

```
# sysctl net.inet.tcp.cc.algorithm=cubic
net.inet.tcp.cc.algorithm: newreno ->
cubic
#
```

D. Internet to home over ADSL1

The home broadband service is ADSL1 with a physical layer synchronised at 1536Kbps downstream (from ISP to home), and 256KBps upstream. The IP link between home and Internet is PPPoE.

The source host was located at the University, on CAIA's own 136.186.229/24 subnet. An IP-over-TCP/IP

[4]Representative of regular applications that people might find themselves using. A future study should probably utilise iperf.

[5]http://www.caia.swin.edu.au/tools/spp

[6]http://www.frenchfries.net/paul/tcpstat/

[7]FreeBSD-9.0-CURRENT-201105-amd64-dvd1.iso under ftp://ftp.freebsd.org/pub/FreeBSD/snapshots/201105

tunnel over the public Internet (using VTUN[8]) links the home LAN with the 136.186.229/24 subnet.

This tunnel means the source and destination hosts appeared to be one hop from each other when in reality a separate, underlying NewReno-based TCP connection was carrying their packets over the public Internet. This 'hidden' layer of NewReno TCP introduces artifacts of its own (such as retransmitting packets lost over the public Internet). Nevertheless, since this sort of tunnelling is not uncommon, we felt it would be interesting to see what differences appeared when running each of the four CC algorithms across an IP-over-TCP/IP tunnel[9].

The idle path between source and destination host has an RTT of roughly 50ms. Each rsync transferred 4Mbyte of data and the destination host ping'd the source host five times per second for the duration of each transfer.

E. HomePlug AV

HomePlug AV is a consumer-oriented technology for creating bridged Ethernet-like services across electrical mains-power lines already present in most homes[10].

Normal HomePlug AV nodes have a peak physical layer (PHY) rate of 200Mbps across the powerlines, and many HomePlug AV devices are marketed using the term "Powerline AV 200"[11], with 100Mbps wired Ethernet connections to external devices. The bitrate achieved by Ethernet frames on a HomePlug AV link have been observed in the 40 - 70Mbps range[12], and in practice depends heavily on the traffic patterns and number of HomePlug AV devices active at a given time.

The in-house host-to-host link utilised two Netgear XAVB 2001 "Powerline AV 200" devices in different rooms to connect the source (Toshiba R700 laptop) and destination hosts. The Netgear management tool reported the PHY synchronised at 107 Mbps from source to destination and 143 Mbps from destination to source.

The idle path between source and destination host has an RTT of roughly 4ms. Each rsync transferred 40Mbyte of data. In addition, the destination host ping'd the source host five times per second for the duration of each rsync transfer.

[8]http://vtun.sourceforge.net/

[9]Similar tunnels also occur when e.g. port-forwarding TCP over SSH (which itself runs over TCP).

[10]https://www.homeplug.org/tech/whitepapers/HPAV-White-Paper_050818.pdf

[11]To differentiate them from both the original 85Mbps HomePlug 1.0 specification and new 500Mbps HomePlug AV devices that have started appearing on the market in 2011

[12]http://www.smallnetbuilder.com/lanwan/lanwan-reviews/31241-homeplug-av-adapter-roundup

F. 802.11g Wireless LAN

The in-house host-to-host WiFi link consisted of a FreeBSD 8.0 host with Ralink Technology RT2560 PCI 802.11g adaptor as access point, and the Toshiba R700 laptop as 802.11g client. Disconnecting the laptop's wired ethernet port ensured all traffic ran over the WiFi interface. Wired 1Gbps Ethernet connected the destination host and access point host. The 802.11g link's PHY speed was not recorded, but Windows 7's link quality indicator suggested "good" connectivity.

The idle path between source and destination host has an RTT of roughly 1.5ms. Each rsync transferred 80Mbyte of data. In addition, the destination host ping'd the source host two times per second for the duration of each rsync transfer.

IV. Results

Here we summarise the RTT and throughput (bandwidth) results for each trial. The median values (RTT or bandwidth) are shown in each Figure's legend.

A. Internet to home over ADSL1

Pushing data over a noisy, uncontrolled Internet path to home reveals a distinct difference between the delay-based and loss-based algorithms.

Figure 1 shows that both Vegas (median ~73ms) and CDG (median ~134ms) induced noticeably lower levels of additional queuing delay than NewReno (median ~492ms) and CUBIC (median ~613ms). Both NewReno and CUBIC seem quite happy to fill excess buffering available along the path, with CUBIC being noticeably worse than NewReno (consistent with previous work under controlled conditions [3]).

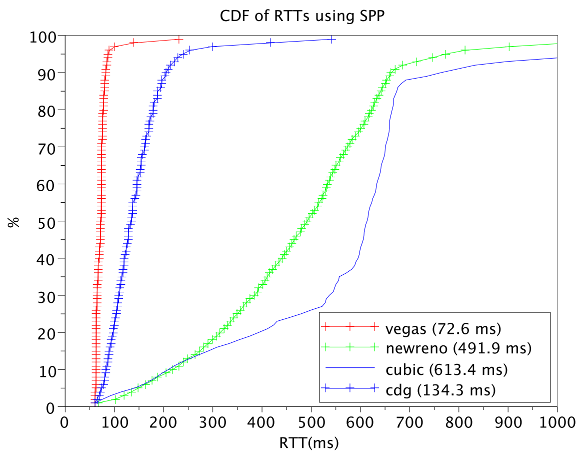

Figure 1. RTT distribution – ADSL1 path

Although Vegas induces the smallest additional queuing delay, there is a definite cost. Figure 2 shows that using Vegas (median ~0.96Mbps) incurs something in the range of a 20% - 25% performance penalty relative to NewReno, CUBIC *and* CDG (medians from 1.2 to 1.25Mbps). In this scenario CDG achieves quite creditable latency figures without giving away much by way of performance.

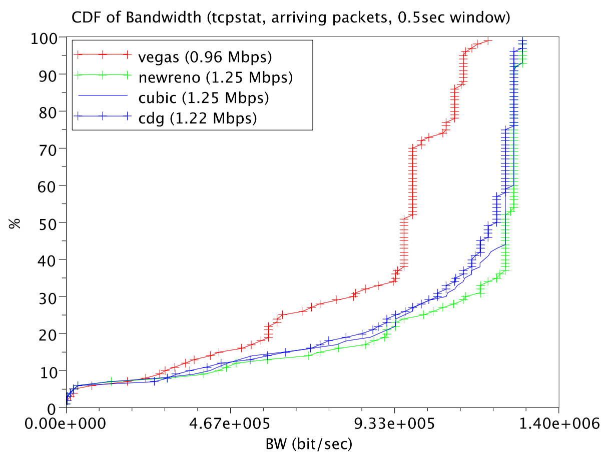

Figure 2. Bandwidth distribution – ADSL1 path

B. HomePlug AV

The results using HomePlug AV are intriguing. Figure 3 shows that none of the four CC algorithms really cause a significant absolute growth in RTT over the link's idle RTT of 4ms, with Vegas again inducing the lowest RTTs.

However, Figure 4 reveals that Vegas suffers a dramatic performance hit, achieving ~14Mbps compared to the ~31 – 36Mbps of the other three CC algorithms. (Interestingly, Figure 4 shows CDG pushing the link slightly harder than NewReno or CUBIC, which might account for its slightly higher median RTT in Figure 3.)

C. 802.11g

Figures 5 and 6 reveal that Vegas appears exceedingly well adapted to the home 802.11g network's queuing and loss mechanisms. CDG induces somewhat less RTT than NewReno and CUBIC (median ~48ms versus ~53ms and ~55ms respectively), while Vegas barely nudges the RTT above the link's idle state. Despite this, all four CC algorithms achieve a a similar median throughput of ~18Mbps.

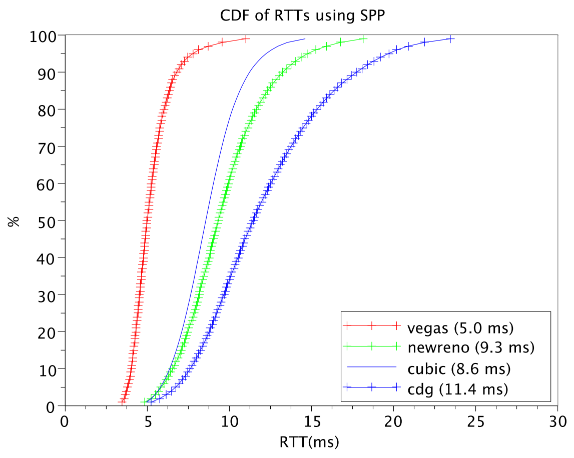

Figure 3. RTT distribution – HomePlug AV path

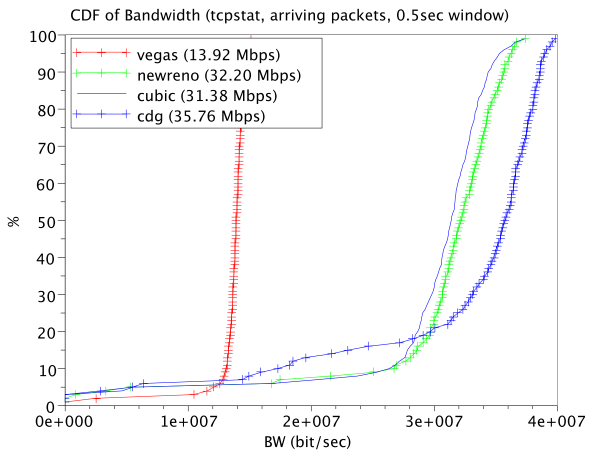

Figure 4. Bandwidth distribution – HomePlug AV path

V. Conclusions and Further work

Delay-based TCP algorithms have been tainted with a reputation for performing worse than loss-based algorithms for straightforward data transfer. In this fairly rudimentary series of experiments, we compare the behaviours of four TCP variants – NewReno, CUBIC, Vegas and "CAIA Delay-Gradient" (CDG) version 0.1 [6], [7] – under FreeBSD-CURRENT.

We focus on a simple (but not-uncommon) home use case – a single stream of data briefly traversing a consumer-grade network link for some tens of seconds – and evaluate the throughput and increase in RTT experienced with each TCP variant. Three different scenarios are tested: a remote Internet site sending to a home connected by ADSL1, in-house host-to-host over

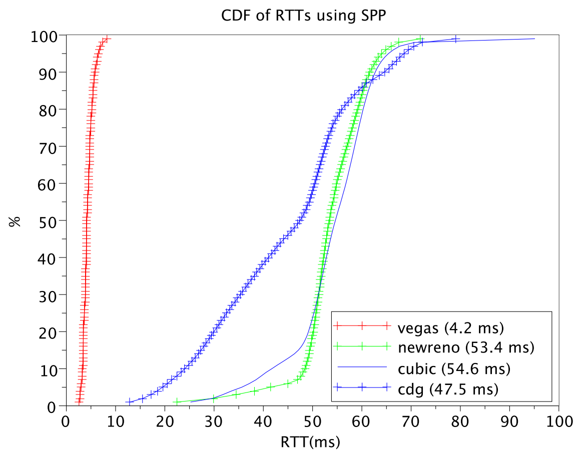

Figure 5. RTT distribution – 802.11g path

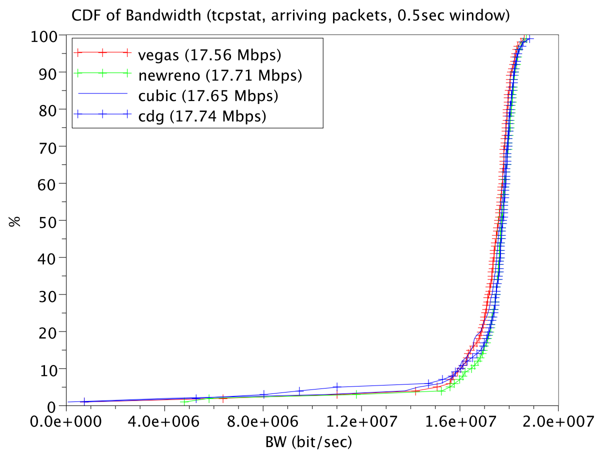

Figure 6. Bandwidth distribution – 802.11g path

a 'HomePlug AV' link and in-house host-to-host over 802.11g WiFi link.

CDG is observed to perform similarly to NewReno and CUBIC, while usually inducing noticably lower RTT. Vegas induces the lowest RTT, but has highly variable performance (often noticably poorer than the other three TCP variants). It would seem that CDG is probably safe to deploy for experimental purposes – you will experience little degradation in performance relative to using NewReno or CUBIC, yet potentially experience significantly less RTT-related collateral damage to other data streams sharing congestion points in your network.

These admittedly simplistic experiments are intended to stimulate further trials that more comprehensively evaluate CDG under a variety of controlled home-network conditions. For example, future work should explore the behaviour of multiple concurrent flows sharing the same home network path(s), using different and/or the same CC algorithms for each concurrent flow. It may also be interesting to consider how an end-to-end TCP session's congestion control interacts with the congestion control inside an IP over a TCP/IP tunnel, particularly when the outer and inner TCP control loops use different CC algorithms.

REFERENCES

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

[2] L. X. I. Rhee and S. Ha, "CUBIC for fast long-distance networks," North Carolina State University, Tech. Rep., Aug. 2008. [Online]. Available: http://tools.ietf.org/id/draft-rhee-tcpm-cubic-02.txt

[3] L. Stewart, G. Armitage, and A. Huebner, "Collateral damage: The impact of optimised TCP variants on real-time traffic latency in consumer broadband environments," in *Proceedings of IFIP/TC6 NETWORKING 2009*, Aachen, Germany, May 2009.

[4] L.Stewart, D. Hayes, G. Armitage, M. Welzl, and A. Petlund, "Multimedia-unfriendly TCP Congestion Control and Home Gateway Queue Management,," in *ACM Multimedia Systems Conference (MMSys 2011)*, San Jose, California, 23-25 February 2011. [Online]. Available: http://portal.acm.org/citation.cfm?id=1943558

[5] "The NewTCP project," Aug. 2008, accessed 8 Aug 2008. [Online]. Available: http://caia.swin.edu.au/urp/newtcp

[6] D. A. Hayes and G. Armitage, "Revisiting TCP congestion control using delay gradients," in *IFIP Networking 2011*, Valencia, Spain, May 2011.

[7] D. A. Hayes, "CAIA Delay Gradient (CDG) Congestion Control Module for TCP v0.1," Mar. 2011, accessed 29 March 2011. [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools.html