Interactions between TCP and Ethernet flow control over Netgear XAVB2001 HomePlug AV links

Radika Veera Valli, Grenville Armitage, Jason But, Thuy Nguyen
Centre for Advanced Internet Architectures, Technical Report 130121A
Swinburne University of Technology
Melbourne, Australia
rveeravalli@swin.edu.au, garmitage@swin.edu.au, jbut@swin.edu.au, tnguyen@swin.edu.au

***Abstract*—HomePlug AV links are usually slower than the surrounding wired LAN links, creating a bottleneck where traffic may be buffered and delayed. To investigate the interactions between TCP congestion control and Ethernet flow control we trialled four TCP variants over a HomePlug AV link using two Netgear XAVB2001 devices. We observed that the XAVB2001's use of Ethernet flow control effectively concatenates the buffers in both the XAVB2001 and the directly attached sending host. This led to multi-second RTTs when using NewReno and CUBIC (loss-based) TCPs, which is significantly troublesome for multimedia traffic on home LANs. In contrast, Vegas and CDG (delay-based) TCPs kept buffer utilisation low, and induced five to ten milliseconds RTT. While keeping RTT low, CDG achieved similar goodput to NewReno and CUBIC. Vegas achieved roughly one third the goodput of the other three TCPs.**

I. Introduction

HomePlug AV is a consumer-oriented technology for creating bridged Ethernet-like services across electrical mains-power lines already present in most homes [1]. HomePlug AV devices are often marketed using the term "Powerline AV 200" and have 100Mbps wired LAN ports. Achievable Ethernet-layer bitrates depend heavily on actual traffic patterns and number of HomePlug AV devices active at a given time. With previously observed 'best case' rates in the 40 to 70Mbps range [2], HomePlug AV links tend to be a bottleneck between 100Mbps wired LANs.

Bottlenecks usually implement internal buffering to minimise packet losses during transient overloads. Unfortunately, TCP traffic and interactive (e.g. multimedia) traffic generally do not interact well when sharing bottleneck buffers.

Common TCP congestion control (CC) algorithms (such as NewReno [3] and CUBIC [4]) tend to cause cyclical filling (until packet loss occurs) and draining (while the sender temporarily slows) of bottleneck buffers along network paths. A side-effect of such *loss-based* CC behaviour is that all traffic sharing the bottleneck will experience additional latency due to increased average *queueing delays* [5]. This is particularly problematic for real-time application flows (such as VoIP and online games) given the growth of arguably-gratuitous buffering ("buffer bloat") in network devices, interface cards and network software stacks [6].

In this report we borrow from, and extend, previous work by one of the current authors [7]. Our new work explores in greater detail the latency and flow control issues that emerge when end-to-end TCP is allowed to make use of all available buffering along the path. Our bottleneck HomePlug AV link consists of two Netgear XAVB2001 (Powerline AV 200) devices with 100Mbps LAN ports. We create long-lived data flows between FreeBSD hosts using two loss-based CC algorithms (NewReno and CUBIC) and two *delay-based* CC algorithms (CDG [8] version 0.1 and Vegas [9]).

We identify two key characteristics of the Netgear XAVB2001 devices:

- They can buffer roughly 2720 Ethernet frames
- They use Ethernet flow control to push back on the sending NIC when their own buffer is full

Ethernet flow control effectively concatenates the sending host's NIC buffers and the XAVB2001's buffers when the HomePlug AV link becomes overloaded. Consequently we observed RTTs (round trip times) in excess of two seconds when NewReno and CUBIC were allowed to 'fill the pipe'. Under similar conditions CDG and Vegas were observed to induce RTTs between five and ten milliseconds. While keeping RTT low, CDG achieved a substantial fraction of NewReno and CUBIC throughput and Vegas achieved roughly one third the throughput of the other three TCPs.

In environments where real-time/interactive multimedia applications share a HomePlug AV link, CDG would

appear a good choice of TCP algorithm, avoiding excessive RTTs that will be caused by NewReno and CUBIC.

The rest of this report is organised as follows. Simplified background on HomePlug AV, Ethernet flow control and TCP is provided in Section II and our experimental methodology is outlined in Section III. Section IV explores the potential performance of an XAVB2001 link, and the XAVB2001's internal buffer size is estimated in Section V. The imapct of Ethernet Flow control on FreeBSD's TCP behaviour is discussed in Section VI. Our measurements of TCP performance and induced RTTs are presented in Section VII. Sections VIII and IX describe future work and conclusions respectively.

II. Background Information

A. HomePlug AV Technology

The HomePlug Powerline Alliance developed the HomePlug AV (HPAV) technology to communicate over the powerline infrastructure. Between a pair of adapters HomePlug AV technology provides a peak unidirectional raw physical (PHY) layer rate of 200 Mbps and a peak information rate of 150 Mbps [1]. These are often marketed using the term "Powerline AV 200" to differentiate them from both the original 85Mbps HomePlug 1.0 specification and newer 500Mbps HomePlug AV devices (which use proprietary extensions for significantly faster physical layer rates [10], [11]).

HomePlug AV uses orthogonal frequency-division multiplexing (OFDM) to encode data on carrier frequencies. Levels of electrical noise on the line influence achievable PHY rates, and devices may synchronise at different physical rates in each direction [12]. Adapters use "variable bit loading", adapting the encoding rate (number of bits per carrier signal) at regular intervals to achieve the highest possible throughput given channel conditions at the time [13].

HomePlug AV links are half-duplex and contention based (somewhat like WiFi), with additional overheads introduced by error detection, error correction, carrier sensing, channel access mechanisms and so forth.

The Ethernet bridging rate and the IP layer rate over the powerline link is lower than the PHY rate due to encapsulation overheads and the MAC protocols used to resolve contention between devices trying to transmit at the same time.

B. The role and impact of buffering

A node in a network path becomes a bottleneck (congestion point) when the packet arrival (ingress) rate at the node exceeds the departure (egress) rate. In such cases, packets need to be temporarily queued inside the node (in a buffer) to avoid losses during bursts of traffic. While the queue (buffer) is close to empty, packets take minimal time transiting through the node. However, during periods where the queue is significantly full, all packets transiting the node experience additional queuing delays as they wait for packets ahead of them to depart. In the extreme, the queue fills up and new packets are dropped (lost).

Buffer size is a key consideration. Small buffers may over-flow easily and lose packets frequently, but impose a low bound on peak queuing delays. Large buffers will absorb more congestion before losing packets, but all traffic sharing the queue will experience high queueing delays when any one or more flows congest the bottleneck node. High queuing delays contribute to higher end to end RTTs, which negatively impacts on interactive applications (such as Voice over IP and online games) and the dynamic responsiveness of protocols that rely on timely feedback (such as TCP itself).

C. Ethernet flow control

When Ethernet traffic arrives on the Netgear XAVB2001's 100Mbps LAN port faster than it can be forwarded across the HomePlug AV link, the XAVB2001 uses a combination of internal buffering and Ethernet flow control [14] to manage the overload. Ethernet flow control works as follows:

Consider an host PC's NIC (network interface card) N_{host} directly connected to port P_{switch} of a congested Ethernet switch. Ethernet flow control allows P_{switch} to send PAUSE frames back to N_{host} requesting that the NIC temporarily pause transmission of Ethernet frames towards P_{switch} for a specific period of time. N_{host} responds to these PAUSE frames by pausing transmission for the time period requested by P_{switch}. The overwhelmed switch never has to lose any Ethernet frames.

In practical terms N_{host} honours these PAUSE frames by briefly buffering outbound Ethernet frames which the host's own network stack is still passing down for transmission. The PAUSE frames effectively pushes the bottleneck back into the NIC, briefly co-opting the NIC's own buffers as an extension of whatever buffers exist inside P_{switch}. From an end to end perspective, the bottleneck appears to have a buffer roughly the sum of the buffers in both N_{host} and P_{switch}.

The network stacks of different operating systems will have their own ways of reacting if N_{host}'s own buffer is exhausted while honouring a PAUSE frame from P_{switch}.

D. TCP Congestion Control Algorithms:

Following on from [7] we chose to utilise two loss-based and two delay-based TCP CC variants – New Reno, CUBIC, CDG (CAIA Delay Gradient) and Vegas. Here we summarise the key TCP mechanisms for maximising throughout while minimising congestion.

Every TCP sender maintains a congestion window (`cwnd`) representing the sender's current best estimate of the number of unacknowledged bytes that the network can handle without causing further congestion. Every TCP receiver advertises a receiver window (`rwnd`) representing the amount of data the receiver can handle at any given point in time. The sender should allow more than either `cwnd` or `rwnd` packets to be transmitted (and unacknowledged) at any given point in time, to avoid overwhelming the network or the receiver respectively.

We broadly classify CC algorithms as either loss-based or delay-based to reflect the type of network feedback used to infer the existence or onset of congestion.

Loss-based TCP CC algorithms (such as NewReno and CUBIC) use the loss of a packet as an indicator of network congestion. They grow `cwnd` as packets are successfully transferred, and reduce `cwnd` when packet loss is identified. Consequently, such CC algorithms cause cyclical filling (until packet loss occurs) and draining (while the sender temporarily slows) of bottleneck buffers along network paths. A side-effect of such behaviour is that all data traffic sharing the bottleneck will experience additional latency variations as bottleneck buffers cycle between non-full and full.

Delay-based algorithms (such as CDG and Vegas) track RTT variations to infer when a bottleneck queue starts filling up somewhere along the path (and hence infer the onset of congestion). They grow `cwnd` as packets are successfully transferred, and reduce `cwnd` in response to the delay-based indication of congestion. The bottleneck queue(s) tend to remain closer to empty and queueing delays are minimised for all traffic sharing the bottleneck queue.

More specific details of NewReno, CUBIC, CDG and Vegas are outside the scope of this report.

III. Experimental Methodology

We perform a number of experiments utilising variations of the simple topology shown in Figure 1. The source host runs 32-bit FreeBSD 9.0-STABLE (svn version r229848M) and the destination host runs 64-bit FreeBSD 8.2-RELEASE. Both the PCs use an Intel(R) PRO/1000 NIC with FreeBSD's `em` driver [15] to access the 100 Mbps LAN links. The source host supports NewReno, CUBIC, Vegas and CDG version 0.1 algorithms[1].

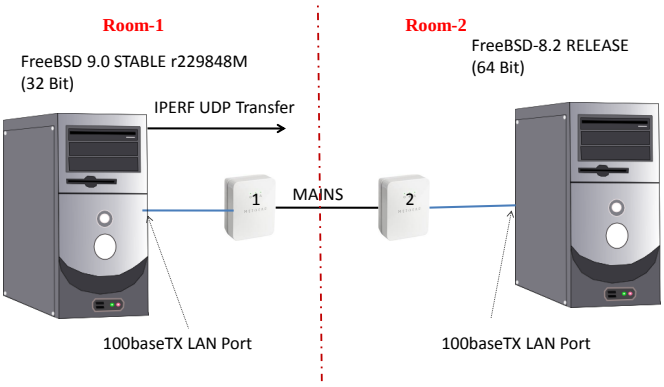

Figure 1. Hosts connected by an XAVB2001 HomePlug AV link

A. Experiments

Our experiments include:

1) Baseline TCP performance over 100Mbps LAN
2) 'Best-case' TCP performance over XAVB2001 link with varying `rwnd`
3) Estimating the XAVB2001 buffer size
4) Distributed buffering via Ethernet flow control
5) Impact of CC algorithms on RTT and TCP goodput over excessively-buffered paths

Experiments 1 and 2 are described further in Section IV. Experiment 3 is in Section V, and experiments 4 and 5 are described in Sections VI and VII respectively.

B. Traffic generation and analysis

Our TCP performance tests utilised nttcp [16] to generate 1GB of TCP traffic for each TCP CC variant and with `rwnd` set to {50 KB, 100 KB, 200 KB, 600 KB, 1 MB, 5 MB, 10 MB[2]} as required to explore the impact of `rwnd` on TCP behaviour. We repeated each of the 28 combinations five times for the TCP over LAN experiment and six times for the TCP over HomePlugAV experiments.

Our buffer size estimation experiments utilised IPERF [17]. We generated UDP traffic at four different rates (50 Mbps, 60 Mbps, 70 Mbps and 80 Mbps) and for each rate transferred four different amounts of data (70 MB, 84 MB, 98 MB, 126 MB). Each combination is repeated 10 times with each of three MTU values {600,1200,1500} bytes, resulting in 480 different combinations.

[1] All four are sender-side CC algorithms so nothing special is requried on the destination

[2] Due an nttcp limitation, we actually used 9999KB for the `rwnd`=10MB trials.

Traffic at both source and destination hosts was captured with tcpdump [18]. FreeBSD's SIFTR [19] logged `cwnd`, RTT and number of bytes in flight as recorded in the TCP control block on a per-packet basis.

C. Variations in PHY rate

The XAVB2001 adapters offer a maximum raw PHY layer rate of 200 Mbps (corresponding to a peak information rate of 150 Mbps) in each direction. The PHY rate can vary during the course of the experiments depending on variations in electrical noise. As noted in Figure 2, the PHY rates can also be different in each direction.

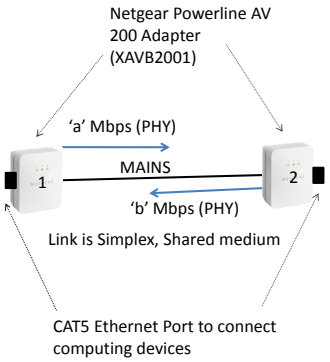

Figure 2. HomePlugAV links are simplex, with PHY rates established separately in each direction

Variations in PHY rates are common when adapters are used in a typical home setup with other electrical equipments sharing the AC lines. We used the "Netgear powerline utility" to observe the PHY rates [20] negotiated from time to time. However, we did not keep detailed records of PHY rates throughout each experiment – it was sufficient that we observe reasonable consistency over time. (The specific relationship between PHY rates and achievable IP layer rates is complicated by the simplex nature of the shared medium, and is outside the scope of this report.)

D. Calibrating time stamps

Our buffer size estimation in Section V assumes we can accurately compare packet-arrival timestamps at both source and destination hosts. Although not shown in Figure 1, each host had a second, separate LAN connection to our lab's corporate network. Although basic time synchronisation was provided using `ntp` on both hosts, the respective clocks could still differ by 10s or 100s of millisecond.

Our solution was to capture and utilise unrelated multicast packets[3] seen on the corporate LAN ports as beacon packets. Copies of each beacon packet are observed arriving at the source and destination hosts at times $TBeacon_{src}$ and $TBeacon_{dst}$ respectively. In reality each copy arrives at both hosts no more than a few microseconds apart[4]. So the offset between the source and destination clocks is given by the difference in apparent arrival times:

$$T_{offset} = TBeacon_{src} - TBeacon_{dst} \quad (1)$$

In Section V we then adjusted all packet-arrival timestamps at one end of the HomePlugAV link by T_{offset} to correct for the offset clocks.

IV. Baseline LAN and HomePlugAV performance

Our first experiments establish a baseline of TCP performance over a basic LAN link and over a 'best case' HomePlug AV link, and explain why some previously published discussion saw TCP performance improve as the number of concurrent flows increased.

A. TCP over 100 Mbps Ethernet

We modified Figure 1 by removing the XAVB2001 adapters and connecting the source and destination hosts directly over LAN cable. TCP performance was tested by running nttcp multiple times as discussed in Section III-B. Across all chosen combinations of `rwnd` and TCP CC variants we observed throughput of ~94 Mbps – very close to the LAN's intrinsic 100 Mbps rate.

B. TCP over 'best case' HomePlug AV link

Measuring the 'best case' performance of TCP over a HomePlug AV link involved using Figure 1 with both XAVB2001 adapters plugged into the same powerboard (to maximise signal strength and minimise external electrical interference). The best case IP layer throughput achieved was ~70 Mbps, due to both the PHY rates negotiated in each direction and the simplex nature of the link (Data packets in one direction contending with ACK packets in the reverse direction). Even under good conditions a HomePlug AV link is likely to be a bottleneck between 100Mbps LANs.

[3] HSRPv2 packets regularly emitted by our corporate network's Cisco switches

[4] Allowing for possibly different propagation times through our Lab switch's ports

C. The impact of receiver window size on throughput

Some early discussion of HomePlugAV devices [2] had reported improved aggregate throughput when using multiple simultaneous TCP flows. By exploring the impact of `rwnd` on throughput over our 'best case' scenario, we conclude that the tests in [2] were simply using an `rwnd` that was too small.

As noted in Section II-D, a TCP source cannot have more un-acknowledged data in flight than `min(cwnd, rwnd)`. Therefore, to utilise the maximum potential TCP goodput, one must ensure the maximum receiver window is not a bottleneck to TCP's window growth. The value of `rwnd` should be such that it allows TCP's window to grow at least up to the 'bandwidth delay product' (BDP) of the path.

No. of TCP Flows	Combined throughput in Mbps	
	`Rwnd` = 50 KB	`Rwnd` = 200 KB
1	53.2	69.5
2	69.0	69.4
3	70.0	70.3
4	69.6	69.4

Table I
THROUGHPUT OF CONCURRENT NEWRENO FLOWS OVER HOMEPLUG AV LINK

Table I shows the aggregate throughput when multiple simultaneous NewReno flows traverse our 'best case' powerline link at `rwnd` sizes 50 KB and 200 KB. When `rwnd`=200 KB, regardless of the number of simultaneous flows across the powerline link, the aggregate throughput (of all the parallel flows) is very similar.

However, when `rwnd`=50 KB, the single flow case is unable to fully utilise the available path capacity (TCP's window growth being limited by the `rwnd` value). Adding a second flow allows each flow to make use of the capacity not used by the other flow, and the aggregate throughput increases noticably. Adding more flows makes little difference – the aggregate appears equivalent to that of a single flow whose `rwnd` is 200 KB. We have essentially 'filled the pipe' (reached the path's BDP).

This explains the observed improvement of throughput with additional TCP flows in [2].

V. ESTIMATING THE XAVB2001'S BUFFER SIZE

The XAVB2001 adapter implements Ethernet flow contol (Section II-C) when overwhelmed, and emits PAUSE frames to the sender requesting a brief halt of transmissions. The XAVB2001's buffer never overflows and there is no occurrence of packet loss in the XAVB2001 itself.

Nevertheless, we are interested in estimating the potential contribution to RTT by an XAVB2001's internal buffering when it is a network bottleneck. We estimate the XAVB2001's buffer size as the amount of data that triggers the first PAUSE frame to be sent by the adapter[5].

A. Methodology

We connected the adapters in different rooms of our lab as shown in Figure 1, with Adapter 1 being the XAVB2001 attached to our source host. This is similar to a home environment where HomePlug AV adapters are used to provide connectivity between different rooms. Using IPERF we observed median UDP-layer throughputs of ~31 Mbps (MTU=600), ~28 Mbps (MTU=1200) and ~35Mbps (MTU=1500)[6]. This approximates the egress rate from Adapter 1 to Adapter 2 over the HomePlug AV link.

We used IPERF (as explained in Section III-B) to overload Adapter 1 by sending UDP packets at close to the LAN link's line rate. Adapter 1's buffer size is estimated using two different techniques, both based around traffic patterns leading up to the emission of a PAUSE frame. By repeating our experiments using Ethernet MTUs of {600,1200,1500} bytes at the sender we could generate UDP traffic with different packets sizes, and thus infer whether the XAVB2001 implements a packet-based or byte-based buffer.

B. Method-1

Figure 3 shows the key events used in Method 1. While IPERF is sending UDP packets towards Adapter 1, we use tcpdump to capture traffic in each direction at the source and destination hosts. The source host's tcpdump file also captures any PAUSE frames sent back to the source by Adapter 1.

Triggered by the receipt of the first PAUSE frame from Adapter 1, we identify time "t" as the transmission time of the UDP packet "x" that initially overwhelmed Adapter 1. We then identify UDP packet "a" that arrived at the destination host at this same time "t".

Adapter 1's buffer size is calculated as $(x - a)$, the number of packets in flight between the sender and the

[5]The actual internal buffer size might be higher, but in practical terms the XAVB2001 is full whenever it emits PAUSE frames.

[6]The MTU=1200 rates dropped slightly due to a temporary drop in PHY speed. Nevertheless, all rates are significantly < 100 Mbps, making Adapter 1 the bottleneck in Figure 1's topology

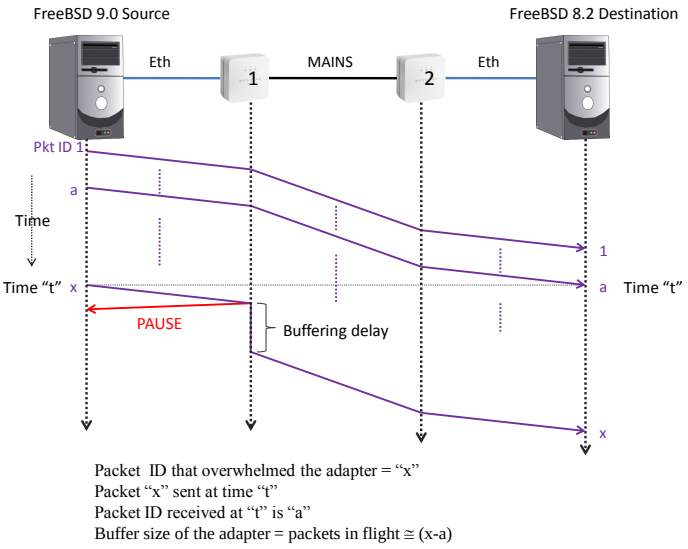

Figure 3. Buffer Estimation – Method-1

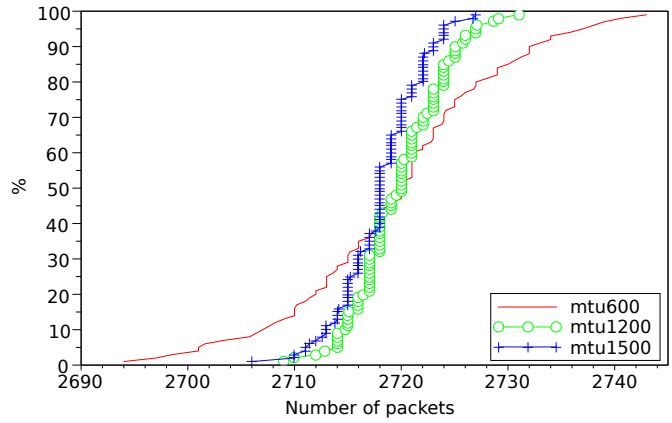

Figure 4. CDF of Buffer Size estimated by Method-1

receiver at time "t". (Since our HomePlug AV link is the significant bottleneck between source and destination, we assume the 100 Mbps LAN links hold a negligible number of these packets in flight.)

Figure 4 is a CDF of Adapter 1's estimated buffer size using Method-1 for each MTU. The median estimated buffer size is roughly 2720 packets, regardless of MTU (and hence UDP packet size). This strongly suggests the XAVB2001 implements a packet-based buffer (where 'full' depends on the number of packets, rather than the number of bytes, being temporarily stored).

C. Method-2

To support the results obtained by Method-1, we also estimated the buffer size using a the method illustrated in Figure 5. Here, we estimate the One Way Delay (OWD) of the first UDP packet as OWD_{idle} when there is no buffering in the powerline adapter. The OWD of the packet that triggered the first PAUSE frame is taken as the OWD_{load} when the adapter is overwhelmed. The additional delay caused by buffering is given by the formula:

$$OWD_{buffer} = OWD_{load} - OWD_{idle} \quad (2)$$

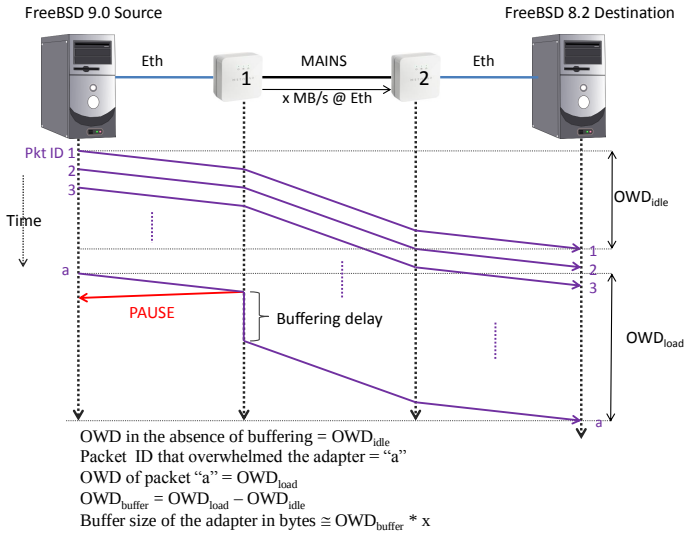

Figure 5. Buffer Estimation – Method-2

The buffer size of the adapter is calculated as the product of the additional delay caused by buffering and the bandwidth offered by the powerline link at the Ethernet layer. IPERF reports the average rate it achieved at the UDP layer, so we scale this (taking into account the UDP, IP and Ethernet headers) to calculate the equivalent Ethernet-layer throughput ($EthernetRate$) over the HomePlug AV link. The buffer size is calculated in bytes and packets as:

$$BufferSize(bytes) = OWD_{buffer} * EthernetRate \quad (3)$$

$$BufferSize(packets) = \frac{BufferSize(bytes)}{MTU} \quad (4)$$

Figure 6 shows the CDF of buffer size of the adapter with various MTU values estimated using Method-2. The estimated buffer sizes are close, with median buffer sizes ranging from 2708 to 2716 packets.

D. Buffer Estimation Results

Table-II shows the median values of buffer size estimated by both methods (in packets and MBytes). It seems reasonable to conclude that the Netgear XAVB2001 adapter can buffer up to 2720 packets before

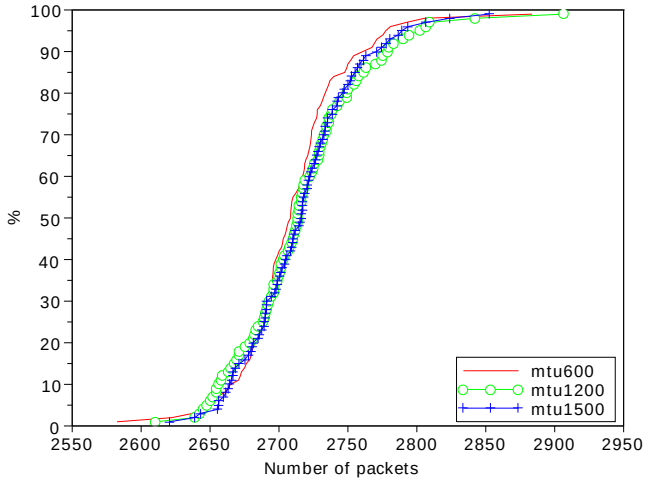

Figure 6. CDF of Buffer Size estimated by Method-2

issuing a PAUSE frame. This represents the usable buffer space – the actual buffer size could be higher depending on how full the buffers are when the adapter sends the first PAUSE frame.

	Buffer size in Packets (MBytes)	
MTU in bytes	Method-1	Method-2
600	2720 (1.5)	2708 (1.5)
1200	2720 (3.1)	2714 (3.1)
1500	2718 (3.9)	2716 (3.9)

Table II
ESTIMATED BUFFER SIZE – NETGEAR XAVB2001

Note that although Method-2 broadly confirms the results of Method-1, Method-2 is potentially less accurate because the IPERF rate used in the calculation is an overall rate throughout the entire transfer time and not just during the OWD_{idle} and OWD_{load} periods under consideration.

VI. DISTRIBUTED BUFFERING VIA ETHERNET FLOW CONTROL

The XAVB2001's buffers are only part of the equation. Ethernet Flow control exerts back-pressure on the device immediately upstream of the XAVB2001 (the source host in Figure 1). In the case of a regular host, its higher-layers (including TCP) are generally unaware of Ethernet-layer PAUSE frames and consequently keep creating more IP packets (and hence Ethernet frames) for transmission. Consequently, the sending NIC's device driver buffers the outbound Ethernet frames (and hence IP packets) when honouring PAUSE frames received from the XAVB2001.

When the HomePlug AV link becomes overloaded, the bottleneck queue is effectively the concatenation of the sending host's NIC buffers and the XAVB2001's internal buffers. In this section we describe how FreeBSD's `em` driver [15] in our testbed's hosts responds to PAUSE frames, and how the `em` driver's own buffering influences TCP traffic over an XAVB2001-based link.

A. FreeBSD's `em` driver and Pause frames

The `em` driver maintains a list of descriptors, one for each packet queued for transmission. Generally this queue will be close to empty unless a higher layer application is generating traffic faster than the attached LAN link speed. However, when a PAUSE frame arrives from a directly attached neighbor FreeBSD's `em` driver stops all transmissions and starts internally buffering new outbound frames until the requested PAUSE period expires.

By default the `em` driver supports up to 256 descriptors, and hence 256 full-sized Ethernet frames. The actual number of descriptors can be altered via the `hw.em.txd` kernel variable – our particular NICs could be configured to support up to 4096 descriptors (and hence buffer up to 4096 frames).

B. FreeBSD TCP's response to NIC buffer exhaustion

If the `em` driver runs out of descriptors, and cannot accept any further Ethernet frames, it returns ENOBUFS up the network stack to exert back pressure on the higher layer(s). This can happen when higher layer(s) generate more than `hw.em.txd` packets for transmission while the `em` driver is honouring a PAUSE frame.

FreeBSD's current TCP stack[7] responds to an ENOBUFS by reducing `cwnd` to one packet and reverting to Slow Start growth (even though FreeBSD's Slow Start usually begins with `cwnd` set to three packets). The TCP stack is correctly treating ENOBUFS as a congestion indication, but is reacting more conservatively than if it had detected actual packet loss.

Figure 7 shows a close-up of `cwnd` over time (in red) and the arrival of PAUSE frames (blue) during a NewReno TCP flow over our HomePlug AV link[8]. At $t = 0$ sec `cwnd` begins growing, and at roughly $t = 1$ sec the XAVB2001 begins emitting PAUSE frames towards the source. Unaware that the `em` driver has started locally buffering outbound packets, `cwnd` continues to grow. At roughly $t = 1.15$ sec the `em` driver exhausts its descriptor

[7]Including FreeBSD 9.1-RELEASE at the time of writing

[8]With an `rwnd` of 10 MB to ensure TCP wasn't receiver-limited, and the `em` driver configured with 256 descriptors

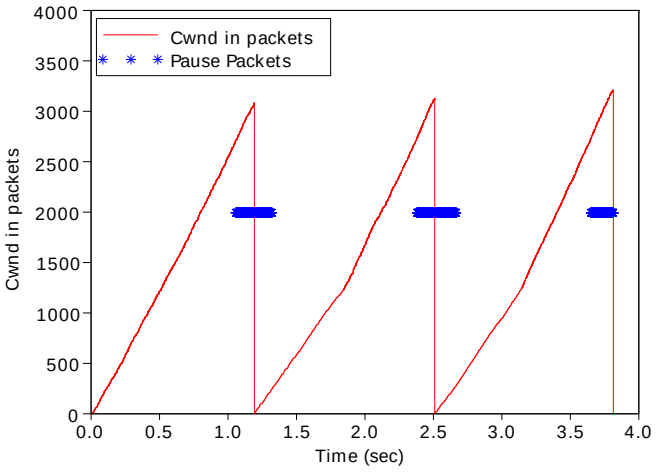

Figure 7. Cwnd fluctuations in response to Ethernet flow control and ENOBUFS in loss-based algorithms

list and returns ENOBUFS to the TCP layer, which immediately resets `cwnd` to one packet and restarts Slow Start. This cycle continues while the TCP connection has data to send faster than the XAVB2001 can forward it.

Figure 7 also reveals that the XAVB2001 is sending a stream of PAUSE frames up to and slighly beyond each ENOBUFS event. This is due to the `em` driver continuing to clear out the backlog of frames caused by the initial PAUSE frames. Recall that PAUSE frames request only short periods of suppressed transmission, during which time the XAVB2001's own buffer will only partially drain. When a PAUSE frame expires the `em` driver resumes sending packets towards the XAVB2001 at line rate (100 Mbps), which quickly re-fills the XAVB2001's own buffer and triggers another PAUSE frame. The resulting rapid succession of PAUSE frames means the NIC is effectively rate limited to the speed at which the XAVB2001 is draining its own buffer (i.e., the rate at which it is transmitting over the HomePlug AV link).

Ultimately the drop in TCP transmission rate after ENOBUFS allows both the `em` and XAVB2001 buffers to drain, and eventually the XAVB2001 ceases sending PAUSE frames (until TCP ramps up its transmission speed again and the cycle repeats).

C. Impact of increased NIC buffering

We ran multiple trials for `hw.em.txd = {256,512,1024,2048}`, resulting in hundreds of ENOBUFS events. The times between the first PAUSE frame and subsequent ENOBUFS event for each cycle were calculated and plotted as cumulative distributions in Figure 8. Not surprisingly, increasing `hw.em.txd` allows the `em` driver to buffer more packets before signalling ENOBUFS.

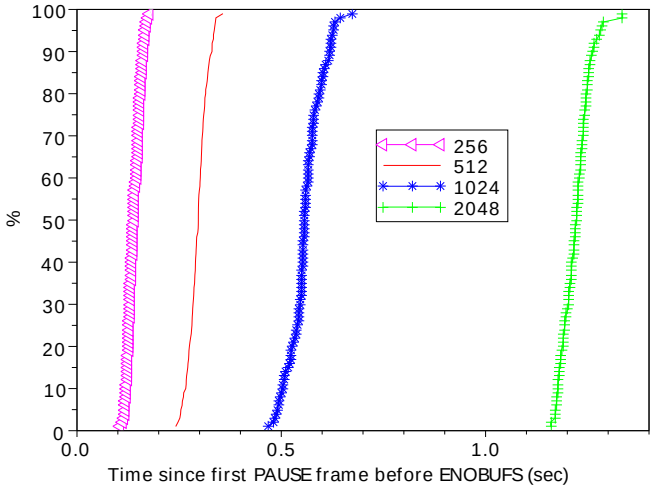

Figure 8. Variation wrt txd of duration of kernel buffering before triggering ENOBUFS

When fed by a a long-lived TCP connection that always has data to send, more `em` driver buffering simply increases everyone's RTT as all packets heading out via this NIC will be queued and delayed.

We also ran trials with `hw.em.txd = 4096`, but did not see any ENOBUF events. Concatenating the XAVB2001 (2720 frames) and `em` driver (4096 frames) creates $1500 * (2720 + 4096)$ bytes (~10 MB), of bottleneck buffering. Under such circumstances our TCP connection (with `rwnd` = 10 MB) could not generate enough packets in flight to overflow the `em` driver's buffer.

VII. Impact of CC algorithm on RTT and throughput over excessively buffered paths

Sections V and VI demonstrate that the XAVB2001 can create an exceedingly bufferbloated path, due to concatenation of the source host's own NIC buffers with the XAVB2001's buffering. In this section we consider the impact of using delay-based rather than loss-based CC algorithms over a significantly bloated HomePlug AV path where our source host is configured for maximum buffering (`hw.em.txd = 4096`).

As described in Section III we used `nttcp` and a range of `rwnd` values to generate long-lived TCP data transfers over the topology of Figure 1, repeated each set of trials for CC={NewReno, CUBIC, Vegas, CDG}, and used FreeBSD's SIFTR kernel module to record `cwnd`

and RTT over time[9]. Most of the following results are derived using XAVB2001 adapters in separate rooms, approximating a common home configuration. At the end of this section we comment on differences we saw when the XAVB2001 adapters were moved onto the same powerboard.

A. *Throughput, induced RTT and* `cwnd` *as a function of* `rwnd` *and CC algorithm*

Figure 9 shows the impact of `rwnd` on median throughput for each of our four TCP algorithms. NewReno and CUBIC were able to extract roughly 39 Mbps from the HomePlug AV link, CDG achieved roughly 34 Mbps and Vegas achieved roughly 12 Mbps. All values of `rwnd` could essentially 'fill the pipe', since even a 50 Kbyte receive window exceeded the link's *intrinsic* (unloaded) bandwidth-delay product[10].

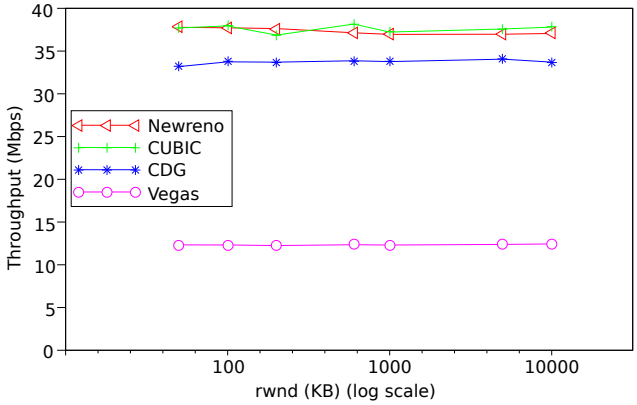

Figure 9. Variation of Median Throughput wrt Receiver Window Size

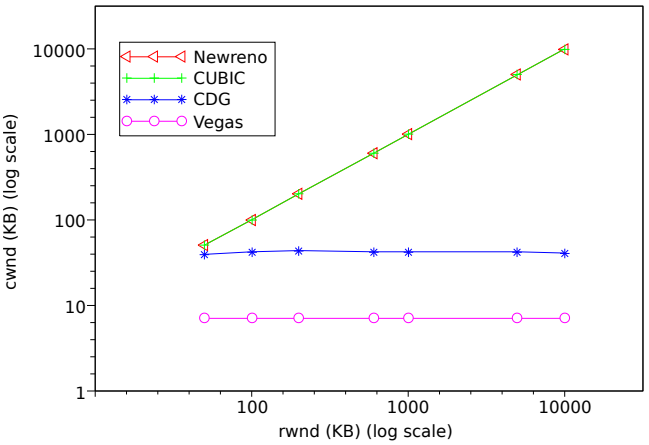

Figure 10. Variation of Median cwnd wrt Receiver Window Size

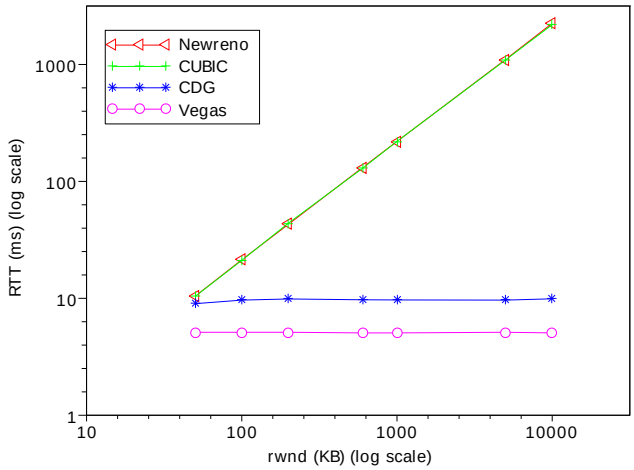

Figure 11. Variation of Median RTT wrt Receiver Window Size

Figures 10 and 11 reveal how median `cwnd` and induced RTT respectively are impacted by `rwnd` and CC algorithm. The loss-based TCPs (NewReno and CUBIC) take advantage of larger `rwnd` by growing `cwnd` and having more packets in flight. But these additional packets simply accumulated in the path's available buffer space – significantly increasing the overall RTT from < 10 milliseconds to multiple seconds without actually improving overall throughput.

In contrast, the delay-based TCPs (Vegas and CDG) tune `cwnd` (and hence number of packets in flight) in response to RTT fluctuations, aiming to send traffic just fast enough to utilise a path's capacity without unecessarily congesting any bottlenecks.

Since they do not 'need' to trigger packet loss as a feedback mechanism, they generally keep buffer utilisation and queuing delays low despite the increased `rwnd`. Compared to the 10 MB `cwnd` experienced by Newreno and CUBIC, CDG's median `cwnd` stays between 40 KB and 45 KB while Vegas' median `cwnd` remains at around 7.2 KB. The median RTT remains roughly 10 ms with CDG and 5 ms with Vegas.

Unfortunately, the low RTT achieved by Vegas is likely related to its significant underutilisation of the path. The underlying reasons why Vegas consistently shows quite poor throughput are a topic for future work.

[9]We found that `cwnd` and number of bytes in flight were identical throughout our trials, so only report `cwnd` in our graphs

[10]This differs from the single-flow `rwnd`=50 Kbyte case in Section IV-C because here we have a lower PHY rate

B. Detailed RTT distributions – NewReno and CUBIC

Figures 12, 13 14 and 15 provide a more detailed insight into the distribution of RTTs induced by NewReno and CUBIC as `rwnd` increases. (The distributions for `rwnd` = {50 KB, 100 KB, 200 KB, 600 KB} and `rwnd` = {1 MB, 5 MB, 10 MB} are separated for clarity.)

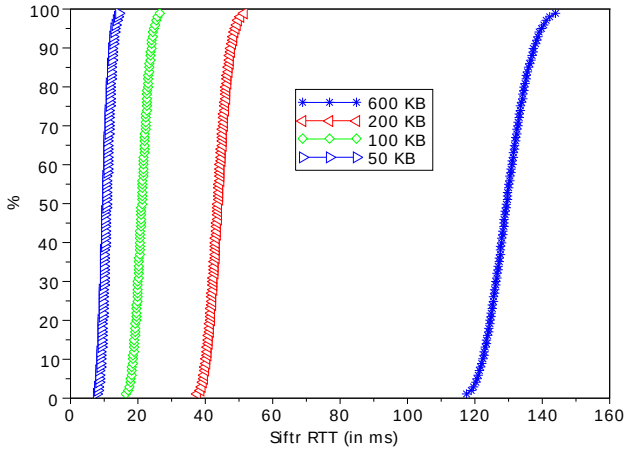

Figure 12. RTT Distribution of CUBIC with lower rwnd

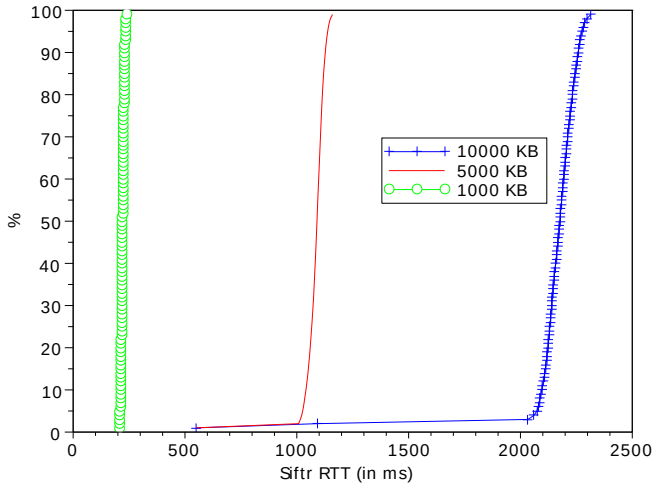

Figure 13. RTT Distribution of CUBIC with larger rwnd

Aside from the `rwnd` = 10 MB cases, the RTT distributions are tightly distributed around their medians. This is not entirely surprising. As noted earlier, the path has a low intrinsic BDP, so even `rwnd` as low as 50 KB can ensure high path utilisation. But with just over 6800 packets of buffering available, neither NewReno nor CUBIC connections are able to trigger any packet loss. Each TCP connection begins in Slow Start with a `cwnd` of three packets, which rapidly grows until the

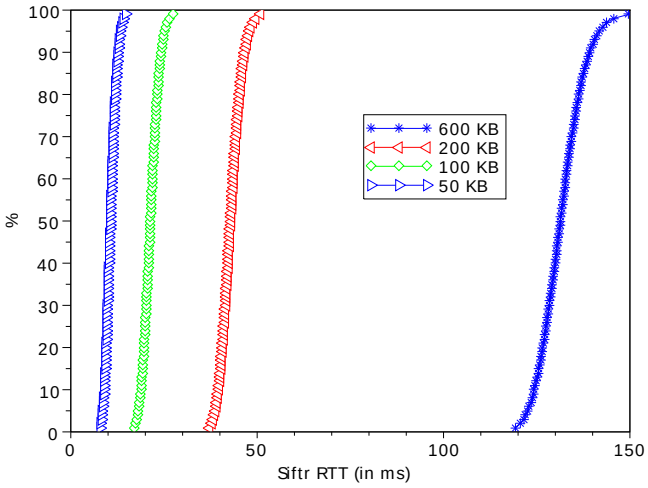

Figure 14. RTT Distribution of Newreno with lower rwnd

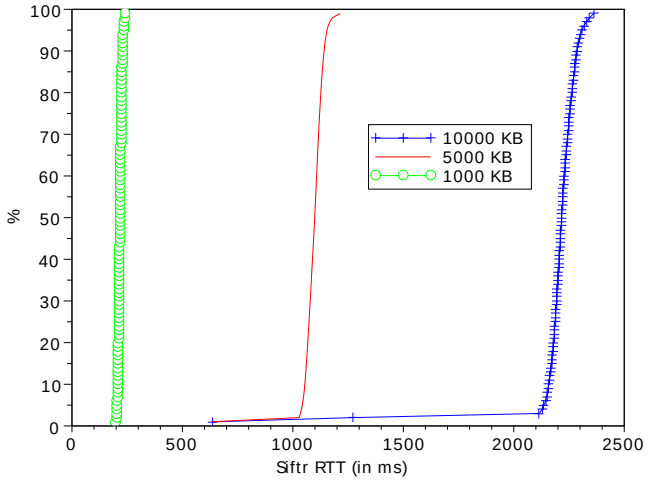

Figure 15. RTT Distribution of Newreno with larger rwnd

number of packets in flight is capped by `rwnd`. At this point the connection is sending data with a roughly steady number of packets in flight, and hence roughly steady number of packets queued along the path.

This steady-state behaviour can be seen in Figure 16's presentation of RTT over time during a NewReno connection with `rwnd` = 10 MB. After rising sharply at the beginning, RTT samples fluctuate in a small band around 2.5 sec for most of the 4 minute data transfer.

Figure 17 magnifies the first few seconds of this connection, revealing clearly how RTT appears to rise linearly from a few ms to about 2.5 s within the first 6 seconds. From this point onwards, RTT stays at around ~2.5 sec as seen in Figure 16.

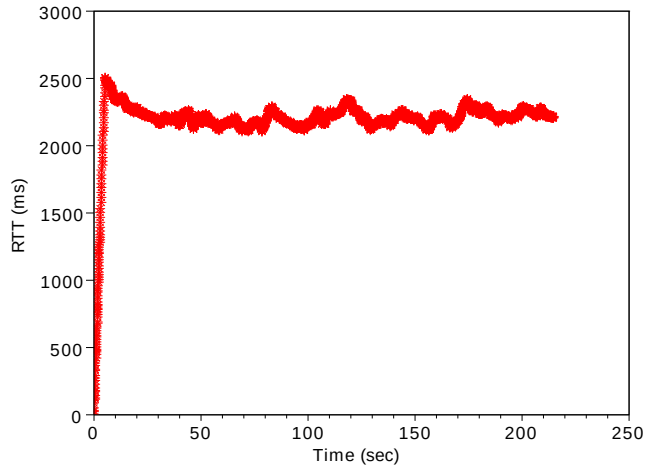

Figure 16. Time varying RTT of newreno with 10 MB rwnd

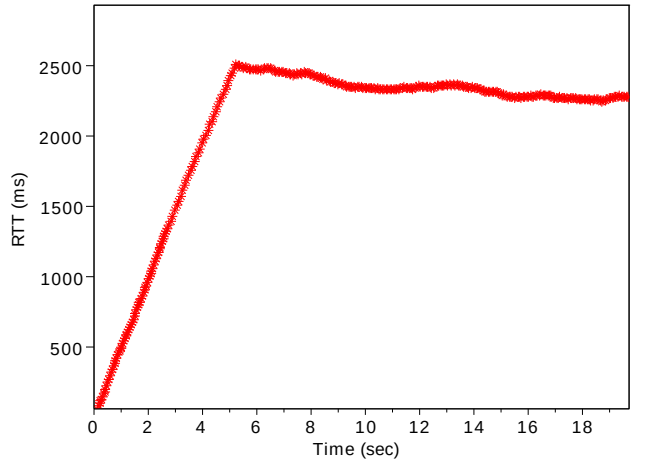

Figure 17. RTT of newreno with 10 MB rwnd (Figure 16 magnified)

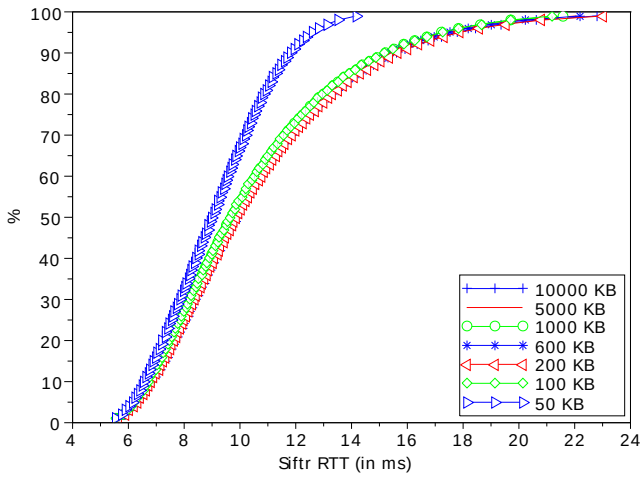

Figure 18. RTT Distribution of CDG

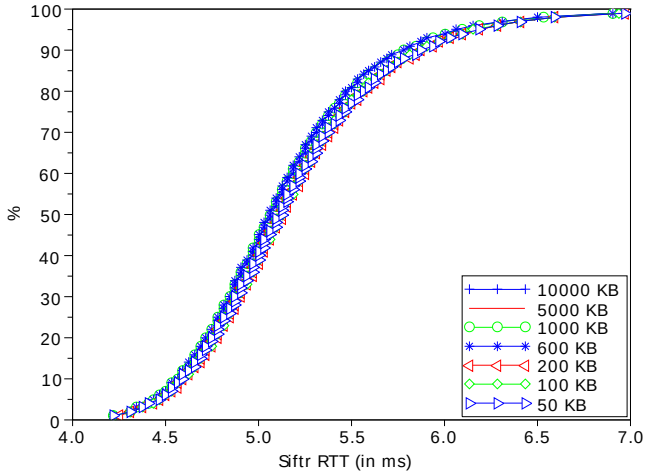

Figure 19. RTT Distribution of Vegas

C. *Detailed RTT distributions – CDG and Vegas*

Figures 18 and 19 provide a more detailed insight into the distribution of RTTs induced by CDG and Vegas. The actual distributions are quite consistent, particularly in the Vegas case, and the upper 80th - 90th percentile RTTs are still quite low in absolute terms.

D. *'Best case' behaviour*

We also re-ran the preceding trials with the XAVB2001 adapters connected to the same powerboard. Although unlikely to be a common household scenario, it is interesting to explore 'best case' PHY rates.

NewReno and CUBIC achieved throughputs of ~70 Mbps, with the induced RTTs dropping from ~2.2 to ~1.2 seconds (due to the increased bandwidth available between the XAVB2001 adapters). However, even this reduced RTT would still be significantly detrimental to any multimedia traffic trying to share such a HomePlug AV link with long-lived TCP flows.

Of the delay-based TCPs, CDG improved from 34 Mbps to 55 Mbps while still inducing low RTT around 10ms. Curiously, Vegas only improved from 12 Mbps to 15 Mbps (with induced RTT around 5ms).

E. *Delay-based TCPs and exessively buffered paths*

The use of PAUSE frames by the XAVB2001 is ultimately detrimental to networks carrying a mixture of loss-based TCP traffic and latency-sensitive traffic. The root cause is that loss-based TCP algorithms only slow down when the network starts dropping packets. If the effective bottleneck buffer space is large enough to avoid packet losses, a loss-based TCP flow creates a standing queue at the bottleneck whose size is limited

only by the destination host's advertised receive window (`rwnd`). Modern operating systems can set `rwnd` to 100s or 1000s of Kbytes, so the result is a bottleneck queue adding significant RTT to all traffic traversing the bottleneck.

Delay-based TCPs are generally immune to paths containing unexpectedly high (and/or hidden) levels of buffering[11]. Our initial experiments suggest that CDG is a plausible choice for TCP-based applications running across HomePlug AV networks. We observed throughputs of ~34 Mbps in a single-flow home use case, which is not far off the ~39 Mbps achieved by NewReno or CUBIC. And yet CDG induced significantly lower queuing delays regardless of the destination's advertised receive window.

VIII. Future work

We have limited our experiments to HomePlug AV network using Netgear XAVB2001 adapters. Future experiments along the same line shuold be performed with different HomePlug AV adapters, both from Netgear and other vendors. It is quite possible that Ethernet Flow control is not uniformly (or even commonly) used by other HomePlug AV devices. It is also possible other devices do not have the same levels of internal buffering as the XAVB2001.

The poor throughput of Vegas over Netgear XAVB2001 adapter is also a topic for further investigation.

IX. Conclusions

This report explores and describes the additional bufferbloat that can be introduced into a HomePlug AV network by the Netgear XAVB2001 adapter and its use of Ethernet Flow control. Our testbed consisted of two Netgear XAVB2001 devices with 100Mbps LAN ports, carrying long-lived TCP data flows between FreeBSD hosts using two loss-based TCP algorithms (NewReno and CUBIC) and two delay-based TCP algorithms (CDG [8] version 0.1 and Vegas [9]).

Although HomePlug AV devices are often advertised as supporting "up to 200 Mbps", this is a peak unidirectional raw PHY rate. The effective rate at the Ethernet layer is usually much lower than 100 Mbps, making the HomePlug AV adapter a bottleneck for traffic flowing from the LAN side to the HomePlug AV side.

We experimentally determined that the Netgear XAVB2001 can store roughly 2720 Ethernet frames internally. Furthermore, the XAVB2001's use of Ethernet Flow control effectively concatenates the XAVB2001's buffers with the NIC buffers of any directly attached source host. Consequently, a HomePlug AV network built around XAVB2001 (or similar) adapters can buffer thousands of frames and potentially exhibit multi-second queuing delays when applications use common loss-based TCP algorithms such as NewReno and CUBIC.

With two Netgear XAVB2001s placed in separate rooms the path exhibited RTTs in excess of two seconds when NewReno and CUBIC were allowed to 'fill the pipe'. Under similar conditions CDG and Vegas induced RTTs between five and ten milliseconds. While keeping RTT low, CDG achieved a substantial fraction of NewReno and CUBIC throughput and Vegas achieved roughly one third the throughput of the other three TCPs.

In environments where real-time/interactive multimedia applications share a HomePlug AV link, CDG would appear a good choice of TCP algorithm, avoiding excessive RTTs that may be caused by NewReno and CUBIC.

Acknowledgements

Radika would like to thank her supervisors Prof. Grenville Armitage, Dr. Jason But and Dr. Thuy Nguyen for their guidance and support throughout this project. She would also like to thank Warren Harrop, Christopher Holman, Lawrence Stewart and Sebastian Zander for providing technical support with the tools used in the experiments.

References

[1] "HomePlug AV White Paper," White Paper, HomePlug Powerline Alliance, August 2005. [Online]. Available: http://www.homeplug.org/tech/whitepapers/HPAV-White-Paper_050818.pdf

[2] T. Higgins, "HomePlug AV Adapter Roundup," 23 August 2010. [Online]. Available: http://www.smallnetbuilder.com/lanwan/lanwan-reviews/31241-homeplug-av-ad=apter-roundup?showall=3D&start=3D4

[3] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681 (Draft Standard), September 2009. [Online]. Available: http://tools.ietf.org/html/rfc5681

[4] I. Rhee, L. Xu, and S. Ha, "CUBIC for Fast Long-Distance Networks draft-rhee-tcpm-cubic-02.txt," IETF Draft, August 2008. [Online]. Available: http://tools.ietf.org/html/draft-rhee-tcpm-cubic-02

[5] 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 *IFIP/TC6 NETWORKING 2009*, Aachen, Germany, 11-15 May 2009. [Online]. Available: http://link.springer.com/content/pdf/10.1007%2F978-3-642-01399-7_31

[11] Particularly if device manufacturers do not advertise their buffer sizes, or co-opt the buffers of neighbouring devices via Ethernet Flow control

[6] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *ACM Queue*, vol. 9, no. 11, pp. 40–54, November 2011. [Online]. Available: http://doi.acm.org/10.1145/2063166.2071893

[7] G. Armitage, "A rough comparison of NewReno, CUBIC, Vegas and 'CAIA Delay Gradient' TCP (v0.1)," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 110729A, 29 July 2011. [Online]. Available: http://caia.swin.edu.au/reports/110729A/CAIA-TR-110729A.pdf

[8] D. A. Hayes and G. Armitage, "Revisiting TCP Congestion Control using Delay Gradients," in *IFIP Networking*, Valencia, Spain, 9-13 May 2011. [Online]. Available: http://caia.swin.edu.au/cv/dahayes/content/networking2011-cdg-preprint.pdf

[9] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, October 1995. [Online]. Available: http://dx.doi.org/10.1109/49.464716

[10] "Powerline AV500 F5D4085," accessed 18th Jan, 2013. [Online]. Available: http://www.belkin.com/us/p/P-F5D4085

[11] "POWERLINE AV 500 ADAPTER KIT XAVB5001," accessed 18th Jan, 2013. [Online]. Available: http://www.netgear.com.au/service-provider/products/powerline-and-coax/powerline/XAVB5001.aspx#

[12] C. K. Lin, H. W. Chu, S. C. Yen, M. T. Lu, J. Yao, and H. Chen, "Robust video streaming over power lines," in *IEEE International Symposium on Power Line Communications and Its Applications*, Orlando, FL, October 2006, pp. 196 –201. [Online]. Available: http://dx.doi.org/10.1109/ISPLC.2006.247460

[13] M. E. Hazen, "The Technology Behind HomePlug AV Powerline Communications," *IEEE Computer*, vol. 41, no. 6, pp. 90 –92, June 2008. [Online]. Available: http://dx.doi.org/10.1109/MC.2008.2053

[14] J. F. Ren and R. Landry, "Flow Control and Congestion Avoidance in Switched Ethernet LANs," in *IEEE International Conference on Communications. Towards the Knowledge Millennium.*, Montreal, June 1997, pp. 508 –512 vol.1. [Online]. Available: http://dx.doi.org/10.1109/ICC.1997.605359

[15] "FreeBSD Man Pages-em," accessed 18th Jan, 2013. [Online]. Available: http://www.freebsd.org/cgi/man.cgi?query=em&manpath=FreeBSD+9.0-RELEASE

[16] "FreshPorts – benchmarks/nttcp," accessed 18th Jan, 2013. [Online]. Available: http://www.freshports.org/benchmarks/nttcp

[17] "Iperf," accessed 18th Jan, 2013. [Online]. Available: http://iperf.sourceforge.net/

[18] "TCPDUMP/LIBPCAP public repository," accessed 18th Jan, 2013. [Online]. Available: http://www.tcpdump.org/

[19] L. Stewart, "Newtcp-tools," accessed 18th Jan, 2013. [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools.html

[20] "Powerline Utility Version 2.0.0.6," 15 October 2011. [Online]. Available: http://kb.netgear.com/app/answers/detail/a_id/17177/~/powerline-utility-version-2.0.0.6