This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal

Using Active Queue Management to Assist IoT Application Flows in Home Broadband Networks

Jonathan Kua, *Member, IEEE*, Suong H. Nguyen,
Grenville Armitage, *Member, IEEE,* and Philip Branch, *Member, IEEE*

***Abstract*—Internet of Things (IoT) applications such as telehealth, smart appliances and smart energy are becoming more common within the home. However, they must compete for bandwidth with traditional applications such as video streaming, video conferencing and bulk file transfers. Such competition can be detrimental to the IoT applications when home gateways use traditional first-in-first-out (FIFO) queue management. Simply increasing bandwidth between the home gateway and the ISP, even when possible, provides no guarantee of bandwidth for IoT applications since many traditional applications will consume as much bandwidth as is available. In this paper we explore whether Active Queue Management (AQM), now being implemented in home gateways, can provide protection for IoT flows. We investigate the effect of different AQM algorithms deployed at the home gateway in scenarios with multiple concurrent application flows. We find that deploying multi-queue FQ-CoDel or the hybrid "FQ-PIE" at the home gateway can provide excellent capacity sharing, flow isolation and good protection in terms of throughput and queuing delays for IoT flows and other applications, which cannot be achieved with traditional FIFO or other single-queue AQMs such as PIE.**

***Index Terms*—IoT, FIFO, AQM, home broadband networks.**

I. Introduction

Recently we have witnessed the fast growth of the Internet of Things (IoT), the network of physical objects with the capability to connect to each other and to other Internet-enabled devices and systems. IoT within the home is being realised through the emergence of Internet enabled devices including smart appliances, smart sensors and health and fitness monitors. Such systems can provide services such as home security and safety, energy management and remote health monitoring [1].

IoT applications usually exhibit event-triggered or low-rate traffic patterns. Event-triggered traffic might include a washing machine sending a text to a resident's phone once a wash cycle finishes, while low-rate traffic might include physiological sensors periodically sending measurements of vital signs to a remote patient portal. Many IoT scenarios involve objects sending telemetry offsite, so IoT flows compete for (often scarce) upstream capacity out of the home.

Even when requiring only low bandwidth, IoT flows often have strict Quality of Service (QoS) requirements (such as consistent delay and low packet loss) because of their importance such as real-time telehealth monitoring or real-time surveillance [2]. Competition over a shared broadband connection with aggressive Internet traffic (such as bulk data transfer or video streaming) can violate these QoS requirements. Protecting IoT flows is an important new task for modern home gateways, for which there are several approaches.

The simplest approach of deploying faster home broadband connections has limited utility. Traditional Internet applications are *elastic*, so they utilise any increase in available bandwidth and still cause degraded performance for competing IoT flows. This is particularly true when the home gateway uses first-in-first-out (FIFO) queue management – the elastic flows induce significant queuing delays, increasing the round trip time (RTT) experienced by all flows sharing the gateway. Another approach is traffic differentiation, with IoT flows assigned to the high priority class [3]. Unfortunately this approach adds configuration complexity, limiting its utility to a small, technically-minded subset of the consumer market. Our paper aims to investigate the efficacy of an emerging third solution based on Active Queue Management (AQM) techniques inside the home gateway. Modern AQM techniques promise to keep queuing delays down without requiring custom local configuration in home gateways.

PIE (Proportional Integral controller Enhanced [4]), CoDel (Controlled Delay [5]) and FQ-CoDel (FlowQueue-CoDel [6]) are three new AQM schemes presently being considered by the Internet Engineering Task Force (IETF) for deployment on ISP and consumer sides of home broadband links. Much work has studied the performance of different AQMs for a variety of traditional Internet applications, a summary of which can be found in [7]. However, we are not aware of any experimental work evaluating the potential performance of sparse IoT-like flows under different AQMs in a home broadband network environment where a blend of different traffic types is present. We contribute to this space by experimentally evaluating and characterising the impact of FIFO and AQMs on the performance of IoT-like flows in terms of achieved throughput and induced queuing delays. We also study the implications of AQMs on other traffic flows, their benefits and trade-offs.

Using an in-house testbed [8], we investigate the effect of PIE and FQ-CoDel AQMs[1] algorithms deployed at the home gateway on the performance of IoT flows in scenarios where various traffic mixes are competing with IoT flows at different

The authors are with the Internet For Things (I4T) Research Lab, Swinburne University of Technology, Australia. E-mail: jtkua@swin.edu.au; nhsuong@gmail.com; garmitage@swin.edu.au; pbranch@swin.edu.au

[1]We do not test CoDel, as the CoDel authors themselves recommend deployment efforts focus on FQ-CoDel.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal

network conditions such as path delays and Internet connection bandwidths. We also evaluate the impact of a novel "FQ-PIE" hybrid AQM scheme. We find that FQ-based algorithms (FQ-CoDel, FQ-PIE) provide excellent capacity sharing and good protection in terms of throughput and delay for the IoT flows in all tested cases, which cannot be achieved with single-queue PIE or FIFO queue management. The advantages of FQ-Codel and FQ-PIE are evident under highly constrained circumstances, where applications that adopt adaptive streaming strategies are protected from collateral damage. Hence, we conclude that multi-queue FQ-based schemes provide good incentives for home gateways to support AQMs where IoT applications share broadband connection with traditional Internet traffic.

The rest of the paper is organized as follows. First, a brief summary of IoT applications, adaptive streaming applications and AQM algorithms is provided in Section II. The experiment setup is described in Section III, followed by experimental results and analysis presented in Section IV. Finally, we offer concluding remarks and outline future work in Section V.

II. Background

This section summarises some IoT applications and their QoS requirements, adaptive video streaming applications and PIE, CoDel, FQ-CoDel, FQ-PIE AQMs.

A. IoT applications

We focus on IoT applications that generate regular offsite traffic, categorized into three groups: health telemonitoring, energy efficiency, and home security [9].

1) Health telemonitoring: Together with teleconsulation and telediagnosis, telemonitoring is an important part of telemedicine. To monitor the health condition of a patient, their vital signs (such as Electrocardiogram - ECG, blood pressure, pulse oximetry and glucose level) are periodically collected by implanted or wearable sensors, then transmitted to a remote patient portal for storage and/or processing. The portal will be accessed by health professionals either immediately to respond to any alarms or later to check/diagnose the patient's health. In some cases actuators may be instructed to perform immediate action upon detecting abnormalities. Other signs, such as Electroencephalogram (EEG) and Electromyogram (EMG) signals, may also be monitored remotely.

These signs and their application-layer bitrates are summarized in [10] and [11], which can be divided into two groups: high bitrates ($\geq$10Kbps, such as EMG and EEG) and low bitrates ($<$10Kbps, such as blood pressure and glucose level). In our experiments, we choose 2Kbps and 100Kbps to represent low and high rate IoT flows. These two bit rates are also representative for IoT applications of other groups.

Delay tolerance for telemonitoring traffic depends on whether the service is real-time (emergency scenarios) or not (non-emergency scenarios) [12]. Both real-time and non real-time telemonitoring services are intolerant of data loss. Delay constraints for real-time services are generally $\leq$ 250ms [13].

2) Energy efficiency: The emergence of smart appliances such as ovens, fridges, dryers, washing machine, light bulbs, meters and thermostats can lead to the development of home energy management to reduce the cost of energy provision in households. These smart appliances can send their real-time energy consumption to a remote server which uses it to analyse, optimize and then control the energy usage of these appliances in an effective manner. The bitrates required by smart appliances for energy management purposes can vary between 1.4Kbps and 250Kbps [9].

3) Home security: These applications range from simple event-based alarm/detection sensors to real-time video surveillance. While event-based traffic is very low bitrate, real-time surveillance generates consistent outbound traffic from tens of Kbps to more than 10Mbps depending on camera quality [14].

B. Adaptive streaming applications

IoT applications are likely to be deployed in home networks where video streaming applications are already in regular use. Dynamic Adaptive Streaming over HTTP (DASH) is an emerging technology that has recently become an international standard for content delivery [15]. DASH-like streaming strategies aim to provide uninterrupted content streaming while adapting video quality in response to dynamically varying network conditions. Rather than being a continuous stream of data, DASH flows involve periodic ON-OFF traffic cycles. Consequently we are interested in the implications of DASH traffic and IoT flows sharing the same bottleneck.

In DASH systems, a video content is encoded into multiple versions at different discrete bitrates, or *representation rates* (RR) [16]. Each version is then fragmented into multi-second video chunks. Chunks are served to clients using standard HTTP servers, which rely on the underlying TCP layer to regulate the actual transmission of packets making up each chunk. DASH clients use a media presentation description (MPD) file associated with the video content to identify available chunks and RRs. DASH clients implement an adaptive bitrate algorithm (ABR) [17] in order to retrieve chunks encoded at the highest RR sustainable by recently observed network conditions.

We first define *achieved rate* (AR) as chunk size divided by the time taken to receive it. When AR is high, DASH will seek to retrieve future chunks encoded at a higher RR to provide better quality of experience. When AR is low, DASH will seek to retrieve future chunks encoded at a lower RR to avoid playout buffer under-run. The client will pre-fill a local playout buffer with multiple chunks before initiating local playback to minimise video stalls in content during playback. The DASH client's ABR algorithm helps to maintain an adequate backlog of content in the face of fluctuating network conditions.

C. Active Queue Management (AQM)

Recent IETF interest in new AQM schemes (revisiting ideas from at least the late 1990s, such as RFC 2309 [18]) has been motivated by the proliferation of oversized buffers in network devices (aka *bufferbloat*) [19], [20]. AQM schemes aim to either drop, or apply Explicit Congestion Notification (ECN)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal

marking to, packets at far lower levels of queuing delay than is typical of classical FIFO (or *tail drop*) queue discipline.[2] Here we summarise three AQM schemes presently being considered by the IETF – PIE [4], CoDel [5] and FQ-CoDel [6]; and a novel "FQ-PIE" [21] hybrid scheme.

1) PIE and CoDel : PIE and CoDel operate on single queues and keep queuing delays low by dropping packets when queuing delays persistently exceed a target delay, T_{target}.

a) PIE : PIE [4] introduces a burst tolerance parameter which (by default) allows packets arriving within the first 150ms of an empty queue to pass successfully. After this, when a packet arrives, it is randomly dropped with a certain probability. This probability is periodically updated, based on how much the current queuing delay (estimated from the queue length and the dequeue rate) differs from $T_{target} = 15ms$ and whether the queuing delay is currently increasing or decreasing. Packets of ECN-enabled flows will be marked instead of being dropped when the dropping probability is $<10\%$. RFC8033 also adds drop probability auto-tuning (maintaining queue stability by avoiding large swings in drop probability) and drop derandomisation (preventing packet losses from occurring too close or too far apart).

b) CoDel: CoDel [5] tracks the (local) minimum queuing delay experienced by packets in a certain interval (initially 100ms). When the minimum queuing delay is less than $T_{target} = 5ms$ or the buffer size is less than one full-size packet, packets are neither dropped nor ECN marked. When the minimum queuing delay exceeds T_{target}, CoDel enters the drop state where a packet is dropped and the next drop time is set. The next drop time decreases in inverse proportion to the square root of the number of drops since the dropping state was entered. When the minimum queuing delay is below T_{target} again, CoDel exits the drop state.

2) FQ-CoDel: In FlowQueue-CoDel [6], flows are classified into one of 1024 (by default) different queues by hashing the 5-tuple of IP protocol number and source and destination IP and port numbers. Each queue is separately managed by the CoDel algorithm. FQ-CoDel uses a Deficit Round Robin (DRR) scheme to service these queues in which each queue can dequeue up to a quantum of bytes (one MTU by default) per iteration. This scheme gives priority to queues with packets from new flows or from "sparse" flows with packet arrival rate small enough so that a new queue is assigned to them upon packet arrival. A bottleneck managed by FQ-CoDel can achieve low latency (due to CoDel), relatively even capacity sharing (due to the fixed hashing function) and priority for low-rate or transactional traffic (e.g. DNS and VoIP).

3) FQ-PIE : One of our novel contributions is evaluating "FQ-PIE" [21]. Whilst combining FlowQueue with PIE was alluded to by PIE's original developers, [21] is the first well documented combination of the FQ aspect of FQ-CoDel with PIE's individual queue management. Like FQ-CoDel, FQ-PIE hashes flows into one of N sub-queues, then applies PIE on these queues and services them with a form of DRR scheduling, prioritising queues with packets from new flows.

[2]Aiming to avoid the large variations in queuing delays that occur when TCP's capacity probing fills and drains bufferbloated FIFO queues.

III. Experiment Setup

Here we describe our experimental testbed and the scenarios for exploring interactions between IoT and non-IoT traffic.

A. A home network blending IoT and traditional traffic

We envisage a home where IoT devices and other consumer services access remote servers across their ISP's broadband last-mile link. The last-mile is commonly asymmetric and a bottleneck for traffic both towards the home (*downstream*) and away from the home (*upstream*). The challenge for IoT services occurs when a mix of traffic overlaps in time, whether due to explicit action of the end-user(s) or background activities launched by in-home devices without end-user intervention. Whether brief or long-lived, the competition for last-mile resources degrades the quality of the IoT service.

Our test conditions explore the potential impact of FIFO and AQMs on various concurrent application flows, when one each of low and high-rate IoT traffic, DASH multimedia stream, bidirectional video calls, upstream and downstream bulk transfers concurrently compete through a single bottleneck.

B. Testbed topology and devices

Figure 1 shows our experimental testbed (based on TEACUP [8]). The bottleneck router uses 64-bit FreeBSD 10.1-RELEASE to provide a two-way path of configurable bandwidth and latency between client(s) on 172.16.10.0/24 and server(s) on 172.16.11.0/24. Hosts run 64-bit FreeBSD 10.2-RELEASE-p7 with NewReno as the default TCP algorithm or 64-bit openSUSE 12.3 (kernel 3.17.4) with CUBIC as default TCP algorithm. For optimal interactive performance, hosts emulating telemonitoring traffic have Nagles algorithm [22] disabled on sources and delayed ACKs disabled on destinations.

C. Traffic generation

Our traffic flows consist of low-rate and high-rate IoT flows (IoT_{low} and IoT_{high}), multimedia streaming (DASH), bidirectional video calls (VideoCall), and file uploads/downloads (BulkUpload/BulkDownload) as described in Table I. We emulate IoT applications that use TCP transport for reliability and that limit their maximum payload size (excluding TCP/IP header) to 100B (as in the ZigBee protocol [23]).

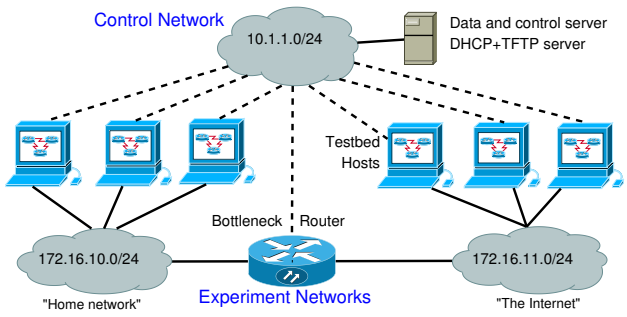


Fig. 1: Testbed emulating a home connected to the Internet [8].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal

TABLE I: Traffic generation of application flows

Name	Description	Application-layer bitrate	RTT_{base}	Host OS and Transport protocol
IoT_{low}	Low-rate telemonitoring traffic, using nttcp [24] to send 100B packets at 2.5 packets/sec upstream over a TCP connection to a remote server.	2Kbps	40ms	FreeBSD & TCP NewReno
IoT_{high}	High-rate telemonitoring traffic, using nttcp to send 100B packets at 125 packets/sec upstream over a TCP connection to a remote server.	100Kbps	40ms	FreeBSD & TCP NewReno
VideoCall	Video call, using using nttcp to generate concurrent upstream and downstream UDP flows of 700B packets every 20ms.	280Kbps	20-240ms	Linux & UDP
BulkUpload / BulkDownload	Bulk file transfers, using iperf [25] to send full-size packets over a TCP connection as fast as possible upstream or downstream (*greedy* flows).	Elastic	20-240ms	Linux & TCP CUBIC (default for Android devices and Linux-based content servers)
DASH	A dash.js v2.0.0 client [26] retrieving 2-sec BigBuckBunny video chunks (with 20 RRs ranging between 46Kbps and 4.2Mbps [27]) from DASH Server based on lighttpd v1.4.35 web server [28].	Variable (depending on retrieved RR)	40ms	DASH server: FreeBSD (inspired by Netflix' use of FreeBSD for Open Connect servers [29]) & TCP NewReno

D. Emulated bottleneck and path conditions

Using "X/YMbps" to represent X Mbps *downstream* and Y Mbps *upstream*, we explore cases where the ISP's last-mile provides rates of {1.5/0.5, 4/1, 12/1, 25/5}Mbps. We represent connections to local/intra-ISP and off-net/inter-ISP with RTT_{base} (a path's intrinsic, unloaded RTT) ranging from 20ms to 240ms for bidirectional video call, upstream and downstream TCP bulk transfers. We fixed IoT flows and DASH streams' RTT_{base} to 40ms to emulate telehealth/telemonitoring services based nearby (e.g. local hospital or medical provider) and content streaming from a local Content Delivery Network.

Our bottleneck router uses FreeBSD's ipfw/dummynet framework to emulate separately configurable bottlenecks and artificial latency in each direction [21]. In a given direction packets first hit a rate-shaping stage (representing the bottleneck link bandwidth) which incorporates the queue management (FIFO, PIE, FQ-CoDel and FQ-PIE) and bottleneck buffer size selected for a given experiment.[3] Each packet is then subjected to additional configurable one-way delay as required (with sufficient buffering to ensure no packets are dropped after passing through the AQM/rate-shaping stage). We emulate RTT_{base} by adding $RTT_{base}/2$ in each direction.

The bottleneck buffer is 490 packets for FIFO experiments (to exceed each path's bandwidth delay product) and 1000 packets for PIE, FQ-CoDel and FQ-PIE experiments. As they are largely intended to require minimal operator control or tuning, we used PIE, FQ-CoDel and FQ-PIE at their default settings. ECN is disabled on all hosts.

E. Measurements

Throughput plots in Section IV are derived from the IP-layer bytes transferred during a five-second window of time sliding forward in steps of 0.25 seconds. The Synthetic Packet Pairs (SPP) tool [31] was used to estimate non-smoothed host-to-host RTT values from packets captured by tcpdump [32] at each host. For DASH flows, per-chunk AR were calculated using payload lengths extracted from HTTP response headers and the time taken to transfer each chunk. Client's HTTP GET requests were parsed to obtain the requested RR.

All experiments run for 200 seconds. Per experiment roughly 680 throughput samples and over 5000 RTT measurements are collected between t=30sec (after DASH buffer pre-fill and other traffic settles) and t=200sec. We confirmed that results from these experiments are reproducible.

[3]We used FreeBSD because its AQM implements PIE based on the algorithm described in RFC 8033 [4] whereas at the time Linux PIE was based on an earlier draft [30] with different burst tolerance behaviour. Linux PIE has not changed between kernel 3.14 and 4.9, http://lxr.free-electrons.com/diff/net/sched/sch_pie.c?diffvar=4.9;diffval=3.14. Both Linux and FreeBSD implement CoDel and FQ-CoDel using the current drafts [5], [6].

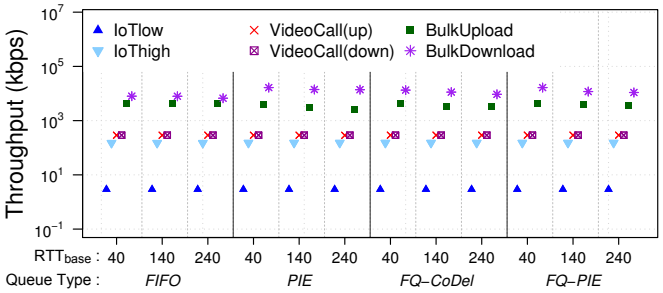
IV. Results and Analysis

Here we present and discuss results of the experiments described in Section III. In the remaining discussion we refer to the VideoCall, BulkUpload and BulkDownload traffic as *competing flows* whose RTT_{base} (unloaded path RTT) is being varied. First we present the impact of these AQMs on the non-DASH flows' IP-layer throughput and all flows' induced queuing delays in Section IV-A, then we explore the impact on DASH stream performance (AR and RR) in Section IV-B. For graphs encompassing a wide range of application throughputs or induced RTTs we use logarithmic Y-axis scales for readability. Where visible, "whiskers" above/below the median depict the 75^{th} and 25^{th} percentiles of throughput or RTT samples respectively.

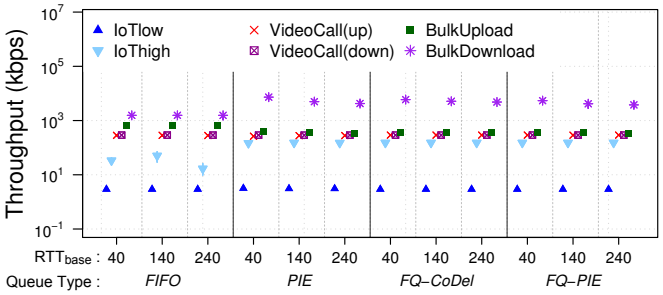
A. Throughput and induced queuing delays

Figures 2 and 3 are boxplots of the throughput and induced RTT experienced by application flows across all four queue management techniques in high (25/5Mbps), mid-range (12/1Mbps) and low (1.5/0.5Mbps) bandwidth scenarios. We ran every experiment with RTT_{base} from 20ms to 240ms in steps of 20ms, and confirm that showing results for $RTT_{base} = \{40, 140, 240\}ms$ adequately captures the influence of RTT_{base}. To further save space we do not present 4/1Mbps results, as they are dominated by the same 1Mbps uplink bottleneck as the 12/1Mbps scenario.

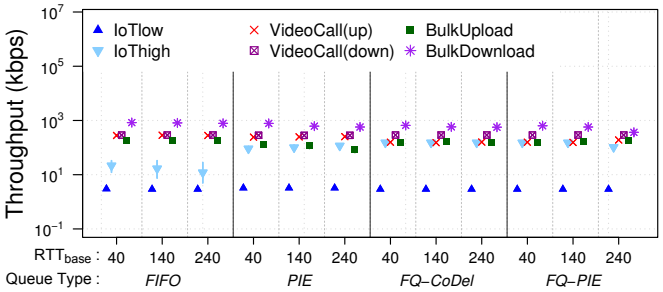
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal



(a) 25/5Mbps

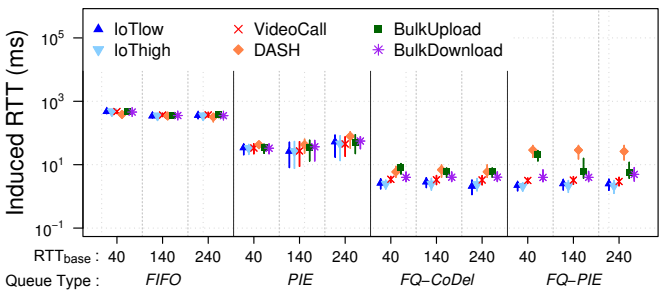


(b) 12/1Mbps

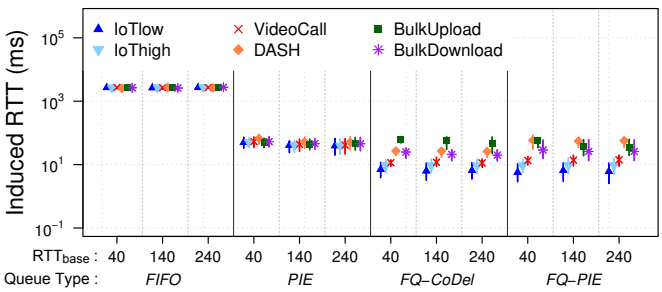


(c) 1.5/0.5Mbps

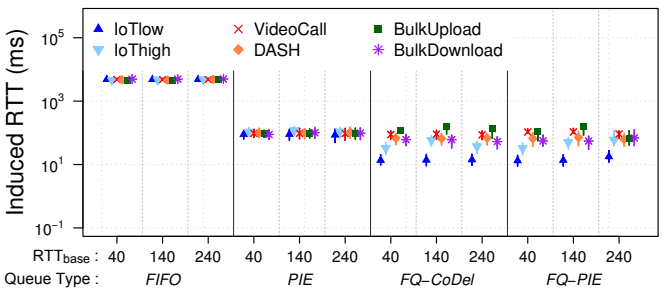
Fig. 2: Throughput per application (log scale) for {25/5, 12/1, 1.5/0.5}Mbps links; FIFO, PIE, FQ-CoDel and FQ-PIE queue management; RTT_{base} applied to competing flows.



(a) 25/5Mbps



(b) 12/1Mbps



(c) 1.5/0.5Mbps

Fig. 3: Induced RTT per application (log scale) for {25/5, 12/1, 1.5/0.5}Mbps links; FIFO, PIE, FQ-CoDel and FQ-PIE queue management; RTT_{base} applied to competing flows.

1) High bandwidth – 25/5Mbps case: Figures 2a and 3a show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 25/5Mbps bottleneck. The tested bottleneck bandwidth provides sufficient capacity for all application flows over all AQMs to achieve maximal throughput, hence capacity sharing of these queuing schemes are not quite evident. Bulk download performs slightly worse in FIFO than in AQMs in terms of throughput, and FIFO trials exhibited significant induced RTT which is consistent with previous bufferbloat observations [20]. All three AQM cases show significantly lower induced RTT than for FIFO. Due to being a single shared queue, PIE shows higher median and larger variations in induced RTT than multi-queue FQ-CoDel and FQ-PIE.

2) Mid-range bandwidth – 12/1Mbps case: Figures 2b and 3b show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 12/1Mbps bottleneck. The difference in capacity sharing and queuing delay management is more evident in this scenario. FIFO has the worst capacity sharing ability, interleaving packets from different flows until the buffer is filled. Under FIFO all traffic flows experience around 2700ms of induced RTT (over ten times the highest RTT_{base}). The IoT_{high} flow is significantly impacted, unable to achieve its 100Kbps application-layer (∼150Kbps IP-layer) target upstream bitrate while competing with the greedy upstream bulk transfer and upstream VideoCall under such high RTT conditions. PIE shows significant reduction in queuing delays and modest capacity sharing, allowing IoT_{high} to achieve its target bitrate and bulk download to achieve a higher throughput than in FIFO case. FQ-CoDel and FQ-PIE provide the best latency reduction, with small variations in queuing delays.

3) Low bandwidth – 1.5/0.5Mbps case: Figures 2c and 3c show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 1.5/0.5Mbps bottleneck. Intuitively, this low bandwidth scenario can result in undesirable capacity sharing and low achieved throughput. FIFO again exhibits the usual bufferbloat

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal

behaviour with median induced queuing delays up to 5000ms. While IoT_{low} achieves its desired throughput, IoT_{high} suffers lower throughput when trying to compete with the greedy upstream bulk transfer and upstream VideoCall across 0.5Mbps capacity. PIE shows significant reduction in induced queuing delays, but IoT_{high} still just misses achieving its target bitrate. FQ-CoDel and FQ-PIE both show significant improvements, enabling IoT_{high} flow to achieve its target bitrate, with much lower and consistent (minimal spreading) induced queuing delays.

4) Discussion: Using AQM yields benefits compared to FIFO across all bottleneck bandwidths. Induced RTTs are significantly reduced, and in some cases throughput is noticeably improved.

When using FIFO we see significant induced RTTs (the known side-effect of bufferbloat), increasing as link speeds decrease (Figures 3a, 3b and 3c). The induced RTTs recorded for the downstream flows are significantly dominated by the congestion-induced queuing delays in the upstream. We do not show the 4/1Mbps results as both 12/1Mbps and 4/1Mbps cases have the same upstream speed and thus the same dominant component of the overall delays. The 25/5Mbps case has a 'less congested' upstream, keeping overall delays lower.

Figure 4 illustrates how bufferbloat-induced RTTs can also be detrimental to throughput in the FIFO cases. Despite running over a 12/1Mbps link, the BulkDownload flow's throughput is less than 2Mbps under FIFO while jumping to between 4-8Mbps with the AQMs. In the FIFO case the path's RTTs around 2.7 seconds lead to a bandwidth delay product (BDP) higher than the maximum TCP window allowed by common end-hosts, thus limiting the BulkDownload flow's achievable throughput. In contrast, the lower RTTs when using PIE, FQ-CoDel or FQ-PIE allow the BulkDownload flow's TCP connection to better utilise the path's shared capacity. (This is also a contributing factor to the poor throughput achieved by the IoT_{high} flow when using FIFO compared to the AQM schemes.)

Figure 4's BulkDownload results also illustrate the trade-offs on the performance of long-lived TCP flows when using AQMs over long RTT paths. By providing congestion signalling at relatively low queuing delays, AQMs effect a 'short queue'. When the AQM bottleneck's peak queue length is less than the path's BDP, the TCP sender's congestion window (cwnd) drops below BDP each time the AQM triggers a TCP back-off, which prevents 100% utilisation during each congestion epoch.

All AQMs allow maximal throughput in the 25/5Mbps case. PIE (with its single queue) shows significantly lower queuing delays than FIFO, but higher and more variable queuing delays than FQ-CoDel and FQ-PIE. The characteristics of FlowQueue's modified DRR scheduler described in Sections II-C2 and II-C3 enable FQ-CoDel and FQ-PIE to provide each traffic flow with a dedicated sub-queue, allowing each flow to achieve lower and more consistent queuing delays. FQ-PIE can exhibit higher induced delays (and wider variations in induced delay) than FQ-CoDel because CoDel has a lower burst-tolerance and target delay threshold than PIE, effectively emulating a shorter queue than PIE. (For example, the DASH flow's RTT over FQ-PIE in Figure 3a is relatively high because every two seconds the DASH flow is able to enqueue many more packets during PIE's burst tolerance period than it can enqueue over FQ-CoDel.)

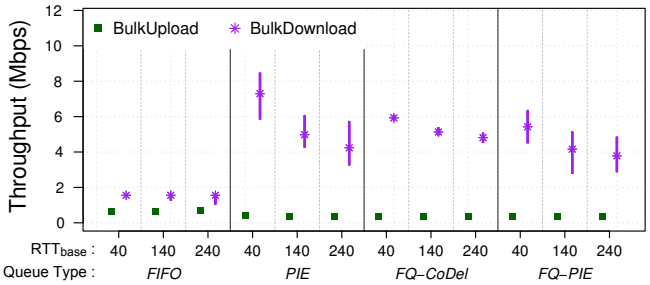


Fig. 4: Each AQM improves BulkDownload performance relative to bufferbloated FIFO (12/1Mbps link, linear y-axis).

FQ-based schemes are superior in terms of adaptive capacity sharing. This is attributable to the FlowQueue scheduler prioritising 'new' or 'recently-seen' flows and sharing the capacity equally between internal sub-queues that are backlogged with packets. The TCP/IP traffic of our IoT flows exhibits periodic ON-OFF behaviour with relatively low bitrate. When a TCP packet from the IoT flow arrives, the FQ scheduler enforces capacity sharing so the IoT flows are not starved of capacity, regardless of how many bulk TCP flows are competing at the bottleneck. During the relatively long gaps between IoT flow packets, the FQ scheduler allows any bulk TCP flow(s) to utilise otherwise-unused bottleneck capacity, and if the sub-queues belonging to IoT flows drain, the next IoT 'burst' is prioritised by the FlowQueue scheduler as a new flow.

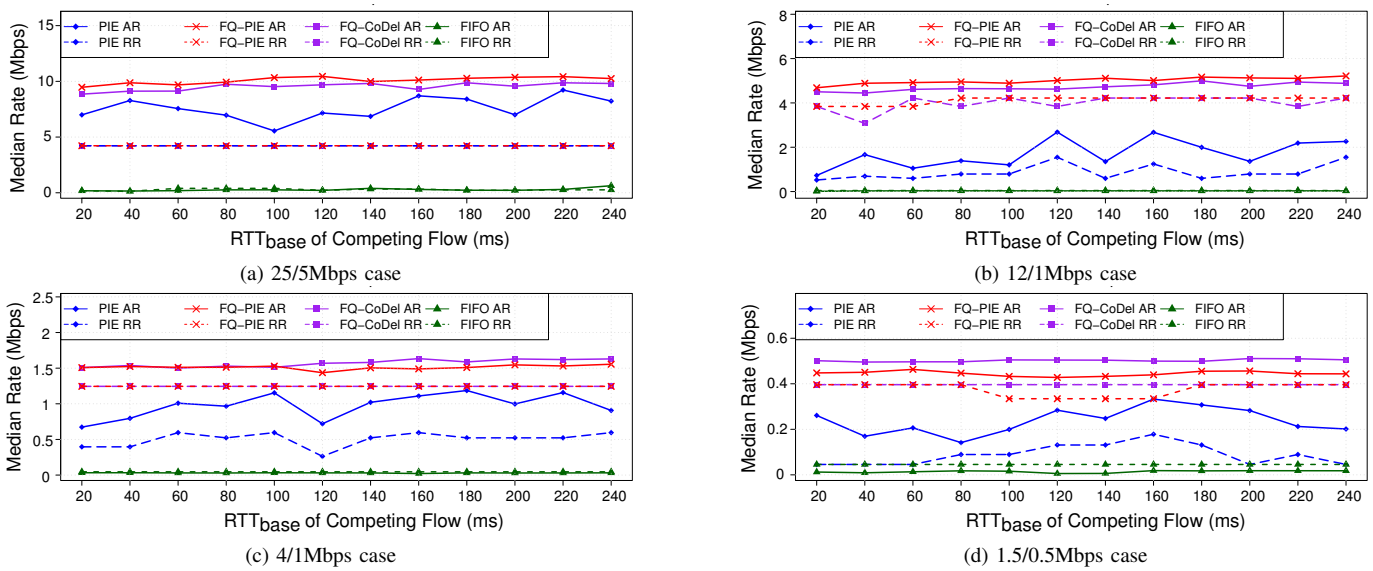
The impact is of FQ-based schemes can be summarised as follows whenever bulk (continuous) TCP flows compete with application-limited (low-rate) ON-OFF TCP flows: The bulk flows will consume whatever capacity is available while the application-limited flow is typically forced to re-grow its cwnd from during successive ON periods. Consequently the application-limited flow may fail to achieve fair capacity sharing unless sharing is enforced (in the case of FlowQueue's DRR scheduler) or the overall RTT is low enough that application-limited flow's ON periods last for multiple RTTs. Using FQ-CoDel or FQ-PIE can satisfy both conditions, using PIE can often satisfy the latter condition, and using FIFO can easily fail on both counts. We discuss the impact on DASH flows in the next sub-section.

B. DASH video streaming performance

Figure 5 captures the impact of various traffic mixes over different queue management techniques on DASH median per-chunk throughput (AR) and target video quality (RR). Regardless of bandwidth availability, FIFO delivers the worst DASH performance, retrieving very low RRs in most cases, and causing frequent video stalls and rebuffering events in the 1.5/0.5Mbps case (Figure 5d) where even the lowest RR is not sustainable.

As discussed in Section IV-A4, FQ-based schemes provide excellent flow isolation and capacity sharing. Since our experiments apply AQMs in both directions, the behaviour of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2722683, IEEE Internet of Things Journal



(a) 25/5Mbps case

(b) 12/1Mbps case

(c) 4/1Mbps case

(d) 1.5/0.5Mbps case

Fig. 5: Impact on DASH streams: Median AR and RR for {25/5, 12/1, 4/1, 1.5/0.5}Mbps links; FIFO, PIE FQ-CoDel and FQ-PIE queue management; RTT_{base} applied to competing flows.

FQ-based schemes is noteworthy for DASH streams when competing with other flows, as the DASH client perceives a higher and more consistent per-chunk throughput (AR) , hence resulting in higher RR being chosen. The flow isolation in the upstream direction assists the regulation of DASH by protecting its ACK streams from being incapacitated by the upstream bulk TCP flow. In all cases, FQ-CoDel and FQ-PIE delivers better performance than PIE in terms of AR and RR.

The 25/5Mbps case (Figure 5a) sees all AQMs allowing the selection of the highest RR because of its high bandwidth capacity. The FlowQueue scheduler allows both FQ-CoDel and FQ-PIE to ensure DASH having a fair share of bandwidth. FQ-PIE allows an overall higher AR than FQ-CoDel due to PIE's higher burst tolerance, drop probability auto-tuning and drop derandomisation [4]. Although single-queue PIE has a lower AR, it is ample to allow DASH client to attain the highest RR.

In the 12/1Mbps scenario (Figure 5b), both FQ-CoDel and FQ-PIE enable DASH streams to achieve an AR greater than 4Mbps most of the time, hence allowing DASH client to retrieve the highest RR. Similar to the 25/5Mbps case, FQ-PIE has a slight edge over FQ-CoDel in terms of AR due to the same reasons. Although PIE performs better than FIFO, its AR and RR is much lower than the FQ-based schemes, only allowing DASH client to retrieve less than 1Mbps RR most of the time.

In the 4/1Mbps scenario (Figure 5c), the peak downstream speed is approximately equal to the maximum available RR in our source material. FQ-CoDel and FQ-PIE again deliver the best performance, achieving ∼1.5Mbps AR, allowing a consistent retrieval of 1.24Mbps RR across all RTT_{base}. PIE shows improvements when compared to FIFO, delivering 0.5Mbps RR in most cases.

The highly-constrained 1.5/0.5Mbps case in Figure 5d shows the DASH stream struggling to achieve a fair share of capacity when using PIE, but achieves ∼400Kbps RR through an FQ-CoDel or FQ-PIE bottleneck. This is particularly significant for low bandwidth connections where video streaming is likely to be unsustainable under traditional FIFO, where the AR drops below the lowest available RR. However, we see improvements when using PIE and even better performance with FQ-CoDel and FQ-PIE.

V. Conclusions and Future Work

In this paper, we have studied the performance of low-rate and high-rate IoT flows in the presence of various concurrent application flows when traditional FIFO, PIE, FQ-CoDel and FQ-PIE AQM algorithms are deployed at the home gateway. Our work evaluates the potential performance of IoT flows in a home network where common competing traffic flows are present – UDP-based video call, bursty DASH-based video streaming and upstream/downstream elastic TCP flows. To the best of our knowledge, we are the first to experimentally investigate the impact of FIFO and AQMs on sparse IoT flows with these traffic combinations in a home broadband environment.

Our experiment results show that traditional FIFO queuing discipline, prevalent in current home gateways provides no throughput and delay protection for the IoT flows. This makes it inappropriate for real-time and latency-sensitive IoT traffic. In contrast, IoT flows managed to achieve their target bitrates with consistently low queuing delays in most cases with AQMs. PIE provides the benefits of keeping queue size moderately small, while dynamic flow isolation of FQ-CoDel and FQ-PIE provides enhanced capacity sharing that allows IoT application flows to functionally coexist with competing flows. The benefits of flow isolation and FlowQueue scheduling capabilities of FQ-based schemes are most clearly evident in mid-range and low bandwidth scenarios. In particular, adaptive applications that use DASH-like strategies suffer from collateral damage when trying to compete with other cross-traffic across FIFO but benefit largely from AQMs. We conclude that an Internet-enabled home blending IoT and non-IoT flows

can yield benefits from deploying modern AQM algorithms, especially FQ-based schemes.

Future work will involve investigating the impact of AQMs on other QoS requirements (e.g. application layer latency) of real-time IoT applications, and various embedded devices which are becoming an integral part of smart homes. Enhancements to transport protocols for more efficient AQM interactions to assist IoT flows remain an open research topic.

ACKNOWLEDGEMENTS

This work was enabled in part by financial support for a project titled "An evaluation of household broadband service requirements for educational innovation and Internet of Things" from the Swinburne University of Technology/Cisco Australia Innovation Fund Committee (2014 – 2016), and in part by PhD stipend support from Netflix, Inc.

REFERENCES

[1] S. Kim, J.-Y. Hong, S. Kim, S.-H. Kim, J.-H. Kim, and J. Chun, "Restful Design and Implementation of Smart Appliances for Smart Home," in *The 11th IEEE Intl Conf on Ubiquitous Intelligence and Computing*, Dec 2014, pp. 717–722. [Online]. Available: https://doi.org/10.1109/UIC-ATC-ScalCom.2014.64

[2] C. Pereira and A. Aguiar, "Towards Efficient Mobile M2M Communications: Survey and Open Challenges," *Sensors*, vol. 14, no. 10, p. 19582, 2014. [Online]. Available: http://dx.doi.org/10.3390/s141019582

[3] P. Nanda and R. C. Fernandes, "Quality of Service in Telemedicine," in *The 1st International Conference on the Digital Society (ICDS 2007)*, Jan 2007. [Online]. Available: https://doi.org/10.1109/ICDS.2007.35

[4] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," RFC 8033, Internet Engineering Task Force, Feb. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8033

[5] D. K. M. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," Internet Engineering Task Force, Internet-Draft draft-ietf-aqm-codel-07, Mar. 2017, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-codel-07

[6] D. Taht, T. Hoeiland-Joergensen, P. McKenney, J. Gettys, and E. Dumazet, "The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm," Internet Engineering Task Force, Internet-Draft draft-ietf-aqm-fq-codel-06, Mar. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06

[7] T. Hoeiland-Joergensen, P. Hurtig, and A. Brunstrom, "The Good, the Bad and the WiFi: Modern AQMs in a residential setting," *Computer Networks*, vol. 89, pp. 90 – 106, 2015. [Online]. Available: https://doi.org/10.1016/j.comnet.2015.07.014

[8] S. Zander and G. Armitage, "TEACUP v1.0 - A System for Automated TCP Testbed Experiments," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 150529A, 29 May 2015. [Online]. Available: http://caia.swin.edu.au/reports/150210A/CAIA-TR-150210A.pdf

[9] T. D. P. Mendes, R. Godina, E. M. G. Rodrigues, J. C. O. Matias, and J. P. S. Catalao, "Smart Home Communication Technologies and Applications: Wireless Protocol Assessment for Home Area Network Resources," *Energies*, vol. 8, no. 7, pp. 7279–7311, 2015. [Online]. Available: http://dx.doi.org/10.3390/en8077279

[10] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," *Wireless Networks*, vol. 17, no. 1, pp. 1–18, Nov 2010. [Online]. Available: https://doi.org/10.1007/s11276-010-0252-4

[11] E. Kartsakli, A. S. Lalos, A. Antonopoulos, S. Tennina, M. D. Renzo, L. Alonso, and C. Verikoukis, "A survey on M2M systems for mHealth: a wireless communications perspective." *Sensors (Basel, Switzerland)*, vol. 14, no. 10, pp. 18 009–18 052, Jan 2014. [Online]. Available: http://dx.doi.org/10.3390/s141018009

[12] L. Skorin-Kapov and M. Matijasevic, "Analysis of QoS Requirements for e-Health Services and Mapping to Evolved Packet System QoS Classes," *Int. J. Telemedicine Appl.*, vol. 2010, pp. 9:1–9:18, Jan 2010. [Online]. Available: https://doi.org/10.1155/2010/628086

[13] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 80–88, Feb 2010. [Online]. Available: https://doi.org/10.1109/MWC.2010.5416354

[14] "NVR Camera Capacity based on Bit Rate," Hikvision, Aug 2012. [Online]. Available: http://oversea-download.hikvision.com/uploadfile/doc/NVR%20Camera%20Capacity%20based%20on%20Bit%20Rate.pdf

[15] "ISO/IEC 23009-1:2014 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," ISO/IEC, May 2014. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=65274

[16] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144. [Online]. Available: https://doi.org/10.1145/1943552.1943572

[17] J. Kua, G. Armitage, and P. Branch, "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP," *IEEE Communications Surveys Tutorials*, 2017, in press. [Online]. Available: https://doi.org/10.1109/COMST.2017.2685630

[18] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309 (Informational), IETF, Apr 1998. [Online]. Available: https://tools.ietf.org/html/rfc2309

[19] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567 (Best Current Practice), IETF, Jul 2015. [Online]. Available: https://tools.ietf.org/html/rfc7567

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

[21] R. Al-Saadi and G. Armitage, "Dummynet AQM v0.2 – CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 160418A, 18 April 2016. [Online]. Available: http://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf

[22] J. Nagle, "Congestion Control in IP/TCP Internetworks," Legacy RFC 896, Jan 1984. [Online]. Available: https://tools.ietf.org/html/rfc896

[23] J. Misic and V. Misic, "Bridging between IEEE 802.15.4 and IEEE 802.11b networks for multiparameter healthcare sensing," *IEEE Journal on Selected Areas in Communications*, no. 4, pp. 435–449, May. [Online]. Available: https://doi.org/10.1109/JSAC.2009.090508

[24] "FreshPorts – benchmarks/nttcp," accessed 13th March, 2017. [Online]. Available: http://www.freshports.org/benchmarks/nttcp

[25] "Iperf," accessed 13th March, 2017. [Online]. Available: https://sourceforge.net/projects/iperf2/

[26] "dash.js: Dash industry forum reference client," accessed 13th March, 2017. [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js/wiki

[27] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: https://doi.org/10.1145/2155555.2155570

[28] "Lighttpd – fly light," accessed 13th March, 2017. [Online]. Available: http://www.lighttpd.net/

[29] "Netflix openconnect appliance software," accessed 13th March, 2017. [Online]. Available: https://openconnect.netflix.com/software/

[30] R. Pan, P. Natarajan, F. Baker, B. VerSteeg, M. Prabhu, C. Piglione, V. Subramanian, and G. White, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem," IETF Draft, Oct 2014. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-pie-00

[31] S. Zander and G. Armitage, "Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet Pairs," in *The 38th IEEE Conference on Local Computer Networks (LCN 2013)*. IEEE, Oct 2013, pp. 264 – 267. [Online]. Available: https://doi.org/10.1109/LCN.2013.6761245

[32] "TCPDUMP & LIBPCAP public repository," accessed 13th March, 2017. [Online]. Available: http://www.tcpdump.org/