

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

Jonathan Kua, *Graduate Student 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 Internet Service Provider (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 multiqueue FlowQueue Controlled Delay (FQ-CoDel) or the hybrid FlowQueue Proportional Integral Controller Enhanced (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 Proportional Integral Controller Enhanced (PIE).

Index Terms—Active queue management (AQM), first-in-first-out (FIFO), home broadband networks, Internet of Things (IoT).

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 realized 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].

Manuscript received August 11, 2016; revised March 31, 2017 and June 27, 2017; accepted June 27, 2017. Date of publication July 3, 2017; date of current version October 9, 2017. This work was supported in part by the project “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 Ph.D. stipend support from Netflix, Inc. (*Corresponding author: Grenville Armitage.*)

The authors are with the Internet for Things (I4T) Research Laboratory, Swinburne University of Technology, Hawthorn, VIC 3122, Australia (e-mail: jtkua@swin.edu.au; nhsuong@gmail.com; garmitage@swin.edu.au; pbranch@swin.edu.au).

Digital Object Identifier 10.1109/JIOT.2017.2722683

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 utilize 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. This 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.

Proportional integral controller enhanced (PIE) [4], controlled delay (CoDel) [5], and FlowQueue-CoDel (FQ-CoDel) [6] are three new AQM schemes presently being considered by the Internet engineering task force (IETF) for deployment on Internet service provider (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 characterizing 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 tradeoffs.

Using an in-house testbed [8], we investigate the effect of PIE and FQ-CoDel AQMs¹ 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 network conditions such as path delays and Internet connection bandwidths. We also evaluate the impact of a novel FlowQueue PIE (FQ-PIE) hybrid AQM scheme. We find that FQ-based algorithms (FQ-CoDel and 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 multiqueue 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 this 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 summarizes some IoT applications and their QoS requirements, adaptive video streaming applications, and PIE, CoDel, FQ-CoDel, and 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 teleconsultation and telediagnosis, telemonitoring is an important part of telemedicine. To monitor the health condition of a patient, their vital signs (such as electrocardiogram, 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: 1) high bitrates (≥ 10 kb/s, such as EMG and EEG) and 2) low

bitrates (< 10 kb/s, such as blood pressure and glucose level). In our experiments, we choose 2 kb/s and 100 kb/s 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 (nonemergency scenarios) [12]. Both real-time and non real-time telemonitoring services are intolerant of data loss. Delay constraints for real-time services are generally ≤ 250 ms [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 analyze, 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 1.4 and 250 kb/s [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 kb/s to more than 10 Mb/s 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 multisec-ond 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 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 prefill a local playout buffer with multiple chunks before initiating local playback to minimize 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.

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

C. Active Queue Management

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 (also known as *bufferbloat*) [19], [20]. AQM schemes aim to either drop, or apply explicit congestion notification (ECN) marking to, packets at far lower levels of queuing delay than is typical of classical FIFO (or *tail drop*) queue discipline.² Here, we summarize 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 150 ms 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 $T_{\text{target}} = 15$ ms 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 derandomization (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 100 ms). When the minimum queuing delay is less than $T_{\text{target}} = 5$ ms 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

²Aiming to avoid the large variations in queuing delays that occur when TCP’s capacity probing fills and drains bufferbloat FIFO queues.

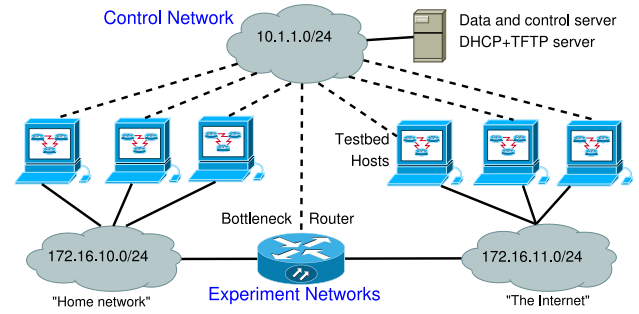


Fig. 1. Testbed emulating a home connected to the Internet [8].

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, prioritizing queues with packets from new flows.

III. EXPERIMENT SETUP

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

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 toward 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

Fig. 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.

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

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 100 B (as in the ZigBee protocol [23]).

D. Emulated Bottleneck and Path Conditions

Using “ X/Y Mb/s” to represent X Mb/s *downstream* and Y Mb/s *upstream*, we explore cases where the ISP's last-mile provides rates of {1.5/0.5, 4/1, 12/1, 25/5} Mb/s. We represent connections to local/intraISP and off-net/inter-ISP with RTT_{base} (a path's intrinsic, unloaded RTT) ranging from 20 to 240 ms for bidirectional video call, upstream and downstream TCP bulk transfers. We fixed IoT flows and DASH streams' RTT_{base} to 40 ms 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.

³We 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 behavior. 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].

The bottleneck buffer is 490 packets for FIFO experiments [to exceed each path's bandwidth delay product (BDP)] 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 s. The synthetic packet pairs tool [31] was used to estimate nonsmoothed 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 s. Per experiment roughly 680 throughput samples and over 5000 RTT measurements are collected between $t = 30$ s (after DASH buffer prefill and other traffic settles) and $t = 200$ s. We confirmed that results from these experiments are reproducible.

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 75th and 25th percentiles of throughput or RTT samples, respectively.

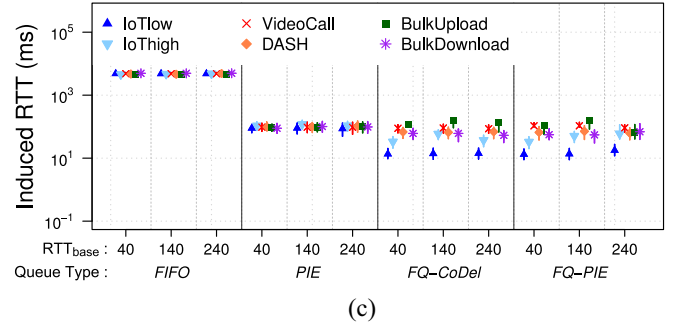
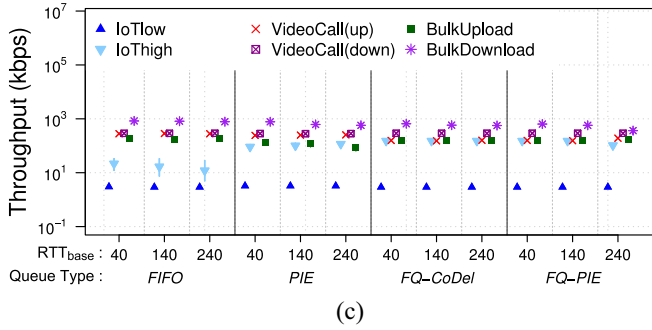
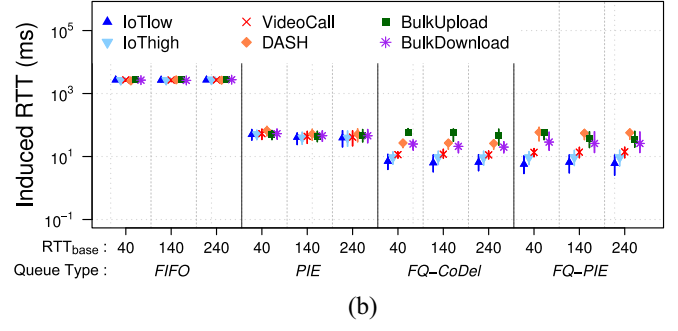
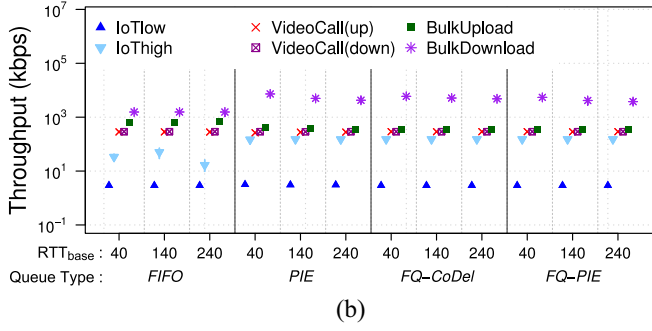
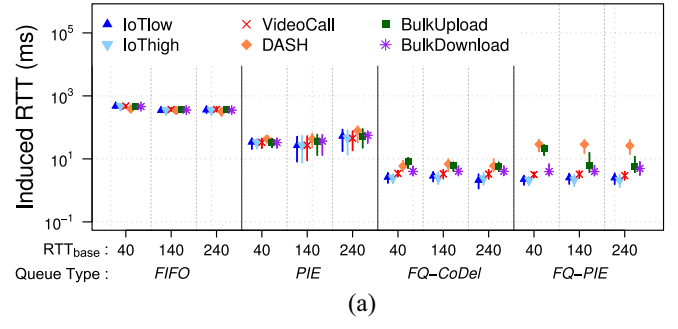
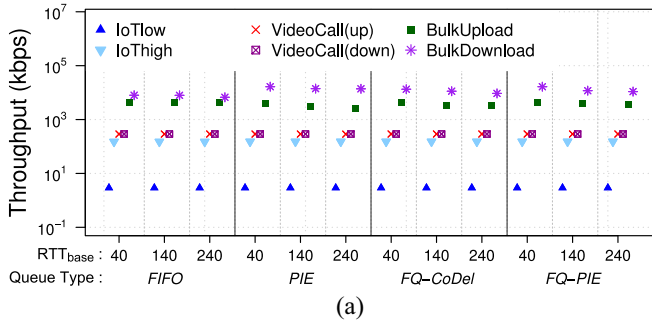


Fig. 2. (a)–(c) Throughput per application (log scale) for {25/5, 12/1, 1.5/0.5} Mb/s links; FIFO, PIE, FQ-CoDel, and FQ-PIE queue management; RTT_{base} (ms) applied to competing flows.

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

A. Throughput and Induced Queuing Delays

Figs. 2 and 3 are boxplots of the throughput and induced RTT experienced by application flows across all four queue management techniques in high (25/5 Mb/s), mid-range (12/1 Mb/s), and low (1.5/0.5 Mb/s) bandwidth scenarios. We ran every experiment with RTT_{base} from 20 to 240 ms in steps of 20 ms, 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 4/1 Mb/s results, as they are dominated by the same 1 Mb/s uplink bottleneck as the 12/1 Mb/s scenario.

1) *High Bandwidth—25/5 Mb/s Case:* Figs. 2(a) and 3(a) show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 25/5 Mb/s 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 multiqueue FQ-CoDel and FQ-PIE.

2) *Mid-Range Bandwidth—12/1 Mb/s Case:* Figs. 2(b) and 3(b) show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 12/1 Mb/s 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 2700 ms of induced RTT (over ten times the highest RTT_{base}). The IoT_{high} flow is significantly impacted, unable to achieve its 100 kb/s application-layer (~ 150 kb/s 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—1.5/0.5 Mb/s Case:* Figs. 2(c) and 3(c) show how throughput and induced RTT for each flow vary with queue management and RTT_{base} of the competing flows for a 1.5/0.5 Mb/s bottleneck. Intuitively, this low bandwidth scenario can result in undesirable capacity sharing and low achieved throughput. FIFO again exhibits the usual bufferbloat behavior with median induced queuing delays up to 5000 ms. 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 0.5 Mb/s 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 [Fig. 3(a)–(c)]. 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 4/1 Mb/s results as both 12/1 Mb/s and 4/1 Mb/s cases have the same upstream speed and thus the same dominant component of the overall delays. The 25/5 Mb/s case has a “less congested” upstream, keeping overall delays lower.

Fig. 4 illustrates how bufferbloat-induced RTTs can also be detrimental to throughput in the FIFO cases. Despite running over a 12/1 Mb/s link, the BulkDownload flow’s throughput is less than 2 Mb/s under FIFO while jumping to between 4–8 Mb/s with the AQMs. In the FIFO case the path’s RTTs around 2.7 s 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 utilize 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.)

Fig. 4’s BulkDownload results also illustrate the tradeoffs 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% utilization during each congestion epoch.

All AQMs allow maximal throughput in the 25/5 Mb/s case. PIE (with its single queue) shows significantly lower queuing delays than FIFO, but higher and more variable

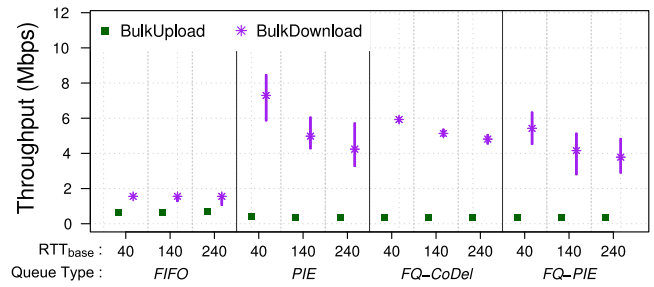


Fig. 4. Each AQM improves BulkDownload performance relative to bufferbloat FIFO (12/1 Mb/s link, linear y-axis).

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 Fig. 3(a) is relatively high because every 2 s the DASH flow is able to enqueue many more packets during PIE’s burst tolerance period than it can enqueue over FQ-CoDel.)

FQ-based schemes are superior in terms of adaptive capacity sharing. This is attributable to the FlowQueue scheduler prioritizing “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 behavior 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 utilize otherwise-unused bottleneck capacity, and if the sub-queues belonging to IoT flows drain, the next IoT “burst” is prioritized by the FlowQueue scheduler as a new flow.

The impact of FQ-based schemes can be summarized 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 regrow 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 section.

B. DASH Video Streaming Performance

Fig. 5 captures the impact of various traffic mixes over different queue management techniques on DASH median

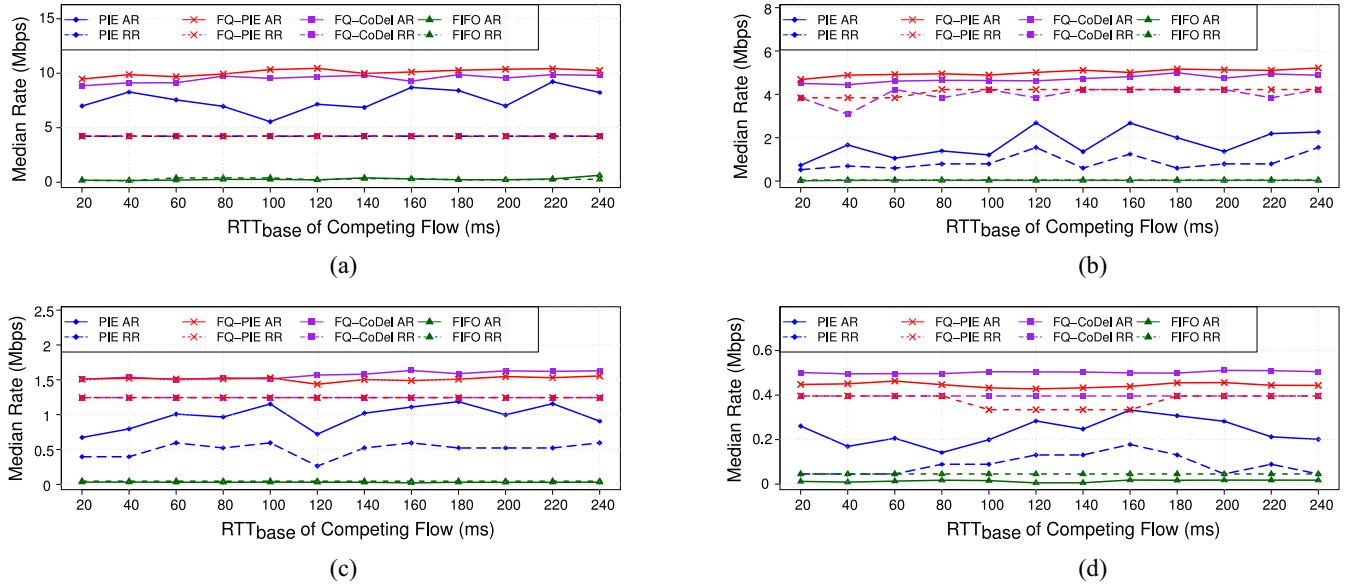


Fig. 5. (a)–(d) Impact on DASH streams: median AR and RR for {25/5, 12/1, 4/1, 1.5/0.5} Mb/s links; FIFO, PIE FQ-CoDel and FQ-PIE queue management; RTT_{base} applied to competing flows.

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 1.5/0.5 Mb/s case [Fig. 5(d)] 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 behavior of 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 25/5 Mb/s case [Fig. 5(a)] 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 derandomization [4]. Although single-queue PIE has a lower AR, it is ample to allow DASH client to attain the highest RR.

In the 12/1 Mb/s scenario [Fig. 5(b)], both FQ-CoDel and FQ-PIE enable DASH streams to achieve an AR greater than 4 Mb/s most of the time, hence allowing DASH client to retrieve the highest RR. Similar to the 25/5 Mb/s 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 1 Mb/s RR most of the time.

In the 4/1 Mb/s scenario [Fig. 5(c)], 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 ~ 1.5 Mb/s AR, allowing a consistent retrieval of 1.24 Mb/s RR across all RTT_{base}. PIE shows improvements when compared to FIFO, delivering 0.5 Mb/s RR in most cases.

The highly constrained 1.5/0.5 Mb/s case in Fig. 5(d) shows the DASH stream struggling to achieve a fair share of capacity when using PIE, but achieves ~ 400 kb/s 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. CONCLUSION

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. This paper 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.

REFERENCES

- [1] S. Kim *et al.*, "Restful design and implementation of smart appliances for smart home," in *Proc. 11th IEEE Int. Conf. Ubiquitous Intell. Comput.*, 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, pp. 19582–19608, 2014. [Online]. Available: <http://dx.doi.org/10.3390/s141019582>
- [3] P. Nanda and R. C. Fernandes, "Quality of service in telemedicine," in *Proc. 1st Int. Conf. Digit. Soc. (ICDS)*, Jan. 2007, p. 2. [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," Internet Eng. Task Force, Fremont, CA, USA, RFC 8033, 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 Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-aqm-codel-07, Mar. 2017. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-aqm-codel-07>
- [6] D. Taht, T. Hoeiland-Jørgensen, P. McKenney, J. Gettys, and E. Dumazet, "The FlowQueue-CoDel packet scheduler and active queue management algorithm," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-aqm-fq-codel-06, Mar. 2016. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06>
- [7] T. Hoeiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The good, the bad and the WiFi: Modern AQMs in a residential setting," *Comput. Netw.*, vol. 89, pp. 90–106, Oct. 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 Adv. Internet Archit., Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep. 150529A, 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. Catalão, "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 Netw.*, vol. 17, no. 1, pp. 1–18, Jan. 2010. [Online]. Available: <https://doi.org/10.1007/s11276-010-0252-4>
- [11] E. Kartsakli *et al.*, "A survey on M2M systems for mHealth: A wireless communications perspective," *Sensors*, vol. 14, no. 10, pp. 18009–18052, 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. Telemed. Appl.*, vol. 2010, pp. 1–18, 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 Commun.*, 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, Zhejiang, China, Aug. 2012. [Online]. Available: <http://oversea-download.hikvision.com/uploadfile/doc/NVR%20Camera%20Capacity%20based%20on%20Bit%20Rate.pdf>
- [15] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*, ISO/IEC Standard 23009-1:2014, 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 *Proc. 2nd Annu. ACM Conf. Multimedia Syst. (MMSys)*, San Jose, CA, USA, 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 Commun. Surveys Tuts.*, to be published. [Online]. Available: <https://doi.org/10.1109/COMST.2017.2685630>
- [18] B. Braden *et al.*, "Recommendations on queue management and congestion avoidance in the Internet," Internet Eng. Task Force, Fremont, CA, USA, RFC 2309, Apr. 1998. [Online]. Available: <https://tools.ietf.org/html/rfc2309>
- [19] F. Baker and G. Fairhurst, "IETF recommendations regarding active queue management," Internet Eng. Task Force, Fremont, CA, USA, RFC, 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. 40–54, 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 Adv. Internet Archit., Swinburne Univ. Technol., Melbourne, VIC, Australia, Tech. Rep. 160418A, Apr. 2016. [Online]. Available: <http://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf>
- [22] J. Nagle, "Congestion control in IP/TCP internetworks," Internet Eng. Task Force, Fremont, CA, USA, RFC 896, Jan. 1984. [Online]. Available: <https://tools.ietf.org/html/rfc896>
- [23] J. Misić and V. B. Misić, "Bridging between IEEE 802.15.4 and IEEE 802.11b networks for multiparameter healthcare sensing," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 4, pp. 435–449, May 2009. [Online]. Available: <https://doi.org/10.1109/JSAC.2009.090508>
- [24] *FreshPorts—Benchmarks/nttcp*. Accessed on Mar. 13, 2017. [Online]. Available: <http://www.freshports.org/benchmarks/nttcp>
- [25] *Iperf*. Accessed on Mar. 13, 2017. [Online]. Available: <https://sourceforge.net/projects/iperf2/>
- [26] *dash.js: Dash Industry Forum Reference Client*. Accessed on Mar. 13, 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 *Proc. 3rd Multimedia Syst. Conf. (MMSys)*, Chapel Hill, NC, USA, 2012, pp. 89–94. [Online]. Available: <https://doi.org/10.1145/2155555.2155570>
- [28] *Lighttpd—Fly Light*. Accessed on Mar. 13, 2017. [Online]. Available: <http://www.lighttpd.net/>
- [29] *Netflix Openconnect Appliance Software*. Accessed on Mar. 13, 2017. [Online]. Available: <https://openconnect.netflix.com/software/>
- [30] R. Pan *et al.*, *PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem*, Internet Eng. Task Force, Fremont, CA, USA, 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 *Proc. 38th IEEE Conf. Local Comput. Netw. (LCN)*, Oct. 2013, pp. 264–267. [Online]. Available: <https://doi.org/10.1109/LCN.2013.6761245>
- [32] *TCPDUMP & LIBPCAP Public Repository*. Accessed on Mar. 13, 2017. [Online]. Available: <http://www.tcpdump.org/>



Jonathan Kua (GS'16) received the Bachelor of Engineering (First Class Hons.) degree in telecommunication and network engineering from the Swinburne University of Technology, Melbourne, VIC, Australia, in 2014, where he is currently pursuing the Ph.D. degree at the Internet for Things Research Laboratory, School of Software and Electrical Engineering.

His current research interests include IP-based content delivery, adaptive multimedia streaming, data transport protocols, and active queue management.



Grenville Armitage (M'03) received the B.Eng. (Hons.) degree in electrical engineering and Ph.D. degree in electronic engineering from the University of Melbourne, Melbourne, VIC, Australia, in 1988 and 1994, respectively.

He is a Full Professor of telecommunications engineering, was the Founding Director of the Centre for Advanced Internet Architectures, and the Founding Head of the Internet for Things Research Laboratory, Swinburne University of Technology, Melbourne. He authored *Quality of Service In IP*

Networks: Foundations for a Multi-Service Internet (Macmillan, 2000) and co-authored *Networking and Online Games—Understanding and Engineering Multiplayer Internet Games* (Wiley, 2006). He is a member of the ACM and ACM SIGCOMM.



Suong H. Nguyen received the B.Sc. degree from the Post and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam, in 2000, the M.Sc. degree from La Trobe University, Melbourne, VIC, Australia, in 2007, and the Ph.D. degree in telecommunications engineering from the Swinburne University of Technology, Melbourne, VIC, Australia, in 2012.

She was a Research Fellow with the Swinburne University of Technology, Melbourne, VIC, Australia, from 2012 to 2013 and from 2015 to

2016. Her current research interests include wireless communication, Internet of Things, network modeling, and data analysis.



Philip Branch (M'94) received the B.Sc. and M.Tech. degrees from the University of Tasmania, Hobart, TAS, Australia, and Ph.D. degree in computer systems engineering from Monash University, Melbourne, VIC, Australia.

He is currently an Associate Professor with the School of Software and Electrical Engineering, Swinburne University, Melbourne, VIC, Australia. His current research interests include network design and analysis and the applications and design of sensor networks. He co-authored *Networking and*

Online Games—Understanding and Engineering Multiplayer Internet Games (Wiley, 2006). He is a member of the ACM.