Preprint © 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Benefits of FlowQueue-based Active Queue Management for Interactive Online Games

Grenville Armitage, Russell Collom
{garmitage,rcollom}@swin.edu.au
School of Software and Electrical Engineering
Swinburne University of Technology, Australia

***Abstract*—Consumers frequently get Internet access through a single home gateway. Gateways using conventional FIFO queue management can introduce hundreds or even thousands of milliseconds of additional delay when congested by bulk TCP data transfers. This delay impacts negatively on any latency-sensitive interactive traffic (such as Voice over IP, or First Person Shooter games). Such applications prefer network connectivity having low latency and low packet loss. New approaches using Active Queue management (AQM) schemes keep queuing delays low by getting TCP to react sooner through the implementation of early packet drops. We characterise the collateral damage caused to interactive application flows by single-queue (CoDel and PIE) and multi-queue (FQ-CoDel and FQ-PIE) AQM schemes, and provide strong experimental evidence to support widespread deployment of multi-queue (FQ) variants.**

***Index Terms*—Games, Packet loss, Active Queue Management, TCP, FlowQueue, Codel, PIE, ECN**

I. Introduction

Consumer broadband services frequently see a mix of traffic and application types competing for access across the 'last mile' link between homes and local service providers. When such links are congested, and use traditional first-in-first-out (FIFO) buffer management at either end, consumers can experience significantly inflated RTT to the outside world. This becomes a significant problem when interactive applications (which prefer low latencies and packet loss rates) are trying to concurrently share the last mile link with file sharing and other bulk data transfer applications.

Recent years have seen significant industry discussion around new active queue management (AQM) techniques for reducing network layer latency while still providing tolerance for bursty Internet traffic. The Internet Engineering Task Force (IETF) has three new AQM schemes – PIE (Proportional Integral controller Enhanced [1]), CoDel (Controlled Delay [2]) and FQ-CoDel (FlowQueue-CoDel [3]). FQ-CoDel has also inspired a FreeBSD implementation of FQ-PIE [4]

Such schemes primarily focus on controlling the behaviour of flows that are using the Transmission Control Protocol (TCP) for end to end congestion control and reliability [5]. Common TCP algorithms react to packet loss as a sign that the network congested, so AQM bottlenecks typically uses packet dropping to provide early congestion signals to TCP senders well before queues build up.

What has not been explored in great detail is the collateral impact on interactive application flows sharing an AQM-managed bottleneck with TCP flows. In this paper we characterise the impact on interactive application flows by single-queue (CoDel and PIE) and multi-queue (FQ-CoDel and FQ-PIE) AQM schemes, and show that single-queue AQMs inflict significant packet losses on low-rate interactive flows as the cost of providing low latency. Our experiments provide evidence to support widespread deployment of multi-queue (FQ) AQM variants such as FQ-CoDel in preference to single-queue variants such as PIE. We additionally demonstrate that enabling Explicit Congestion Notification (ECN) [6] does not materially change the need to protect interactive traffic with a multi-queue variant of AQM.

The rest of the paper is structured as follows. Section II provides background and overview of the typical consumer use case, congestion and RTT inflation, the needs of interactive applications such as online games, and emerging AQM schemes. Section III describes the experimental methodology we have used to explore the interactions between game traffic and AQM schemes. Section IV presents our results, and in Section V we discuss how these results generalise. The paper concludes in Section VI.

II. Background and Motivation

A. Consumer broadband access and traffic types

Figure 1 captures the modern reality of an increasingly diverse range of applications competing for a share of each home's broadband link to the Internet. Bandwidth over the last-mile link is often asymmetric, being significantly higher towards the home (downstream) than away from the home (upstream). The traditional home-as-content-consumer is being replaced by homes where traffic is just as likely to be outbound (sharing content, or offsite backups) as inbound (traditional content consumption, software updates, and so forth). Challenges occur 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.

Common consumer applications can be mapped into three broad categories: Latency-sensitive / interactive traffic, latency-tolerant (elastic) traffic, and streaming content traffic. Consumer experience of poor interactive service is often due to the way current home gateway designs allow mutual interference between traffic belonging to each of these three categories.

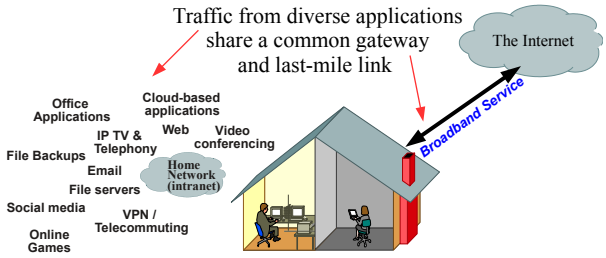


Figure 1: Consumer network access often involves diverse applications sharing a common gateway and link to the Internet

Latency-sensitive covers applications that value timeliness of information transfer. Some are continuously interactive, such as Voice over IP (VoIP), online games (particularly 'twitch' games like First Person Shooters [7]) and other highly immersive (virtual) environments. Traffic tends to be relatively consistent packet flows in each direction, at modest data rates dictated (and limited) by the applications themselves.[1] Some are sporadically interactive, like domain name lookups, where the response time to a query is strongly influenced by network latency. A good network path offers low round trip time[2] (RTT) rather than bandwidth in excess of what the latency-sensitive application needs.

Latency-tolerant (or *elastic*) covers applications that are relatively unconcerned about the absolute RTT at any point in time and willing to tolerate slowing down or speeding up as dictated by available bandwidth. Examples include sync'ing content to/from "the Cloud", sending/receiving emails, application or operating system updates, downloading audio-visual content in their entirety for deferred, offline playback, and so forth. Such applications tend to be judged more on overall time to completion (TTC) – such as how long to upload or download a podcast, photo or app update. However, they commonly use the Transmission Control Protocol (TCP) [5] as underlying transport, creating a key issue: While the elastic application has data to send, TCP will seek to consume all the free bandwidth available at any given instant. Often this will congest the link between home and Internet in the direction of data flow, leading to inflation of RTT for all traffic sharing the gateway at the time. The elastic traffic's own TTC also degrades as RTT increases.[3]

Streaming involves multimedia content delivery of content (such as over-the-top Internet movie, TV and radio services) just fast enough (on average) to be consumed by multimedia clients in real-time. Once a consumer has selected some content and the stream has begun playing, streaming services are usually more relaxed than interactive services about short-term variations in network RTT. Modern audio-visual streaming services are migrating towards technologies such as, or similar to, DASH (Dynamic Adaptive Streaming over HTTP) [8]. DASH creates repeated bursts of short-lived elastic traffic on the network as clients retrieve small 'chunks' of content piece-meal from the server over time. In conjunction with conventional TCP, DASH-like streams cause periodic spikes of congestion on the home broadband link.

[1]For example, video codecs generate data at variable rates, but no faster than required to encode the content given the chosen encoding algorithms.

[2]Time taken by a packet to go back and forth between two points due to both queuing delays at bottlenecks and transmission (propagation) delays dictated largely by network topology, technology and physical distance.

[3]Higher RTT slows TCP's loss detection & congestion control feedback loop, in addition to slowing the setup of new TCP connections.

B. Congested home gateway buffers, TCP and RTT inflation

The Internet requires certain amounts of buffering in routers and gateways to absorb transient bursts of traffic. Conventionally this has been in the form of finite-sized, first-in-first-out (FIFO) queues. With FIFO queues, newly arrived packets are added to the end of the queue (enqueued) if there is space available, or dropped if the queue is full. Successfully enqueued packets move towards the front of the queue as older packets are transmitted (dequeued) at the outbound link speed and in the order they arrived. In other words, when multiple flows of traffic meet in a FIFO queue, everyone's packets get backed-up and suffer wait times (queuing delays) of up to $\frac{MaxQueueSize}{LinkSpeed}$ seconds.

TCP uses windowed flow control, where the sender allows only `cwnd` (congestion window) bytes to be in flight and unacknowledged at any given time. TCP senders probe for path capacity by growing `cwnd` as packets are successfully acknowledged, and shrinking `cwnd` when congestion is detected. As a result, standard *loss-based* TCPs (such as NewReno and CUBIC) will cyclically fill FIFO bottleneck queues to the point of packet loss (while growing `cwnd`), let them drain somewhat (when packet loss triggers reduction of `cwnd`), and then repeat. The result is cyclically varying queuing delays for all traffic sharing a bottleneck.

To ensure TCP does not under-utilise a path, and because traditional TCPs halve their `cwnd` on packet loss[4], bottleneck buffering is usually sized to equal or exceed a path's unloaded *bandwidth delay product* (BDP). To avoid negative market perceptions, consumer gateways are motivated to provide at least a BDP of buffering based on the highest likely last-mile bandwidth and RTT_{base} (unloaded RTT) to offsite locations with whom their customers are likely to exchange data.

The last-mile link between home and Internet is usually a bandwidth bottleneck both upstream and downstream. Upstream traffic causes queue build-up in the home gateway's upstream buffers while downstream traffic causes queue build-up in the downstream buffers on the ISP side. Signficant RTT inflation is experienced by all traffic sharing a last-mile with TCP-based consumer applications, due to cyclical queue filling and draining at one or other end of the link.

Router and gateway vendors have also traditionally including significant amounts of buffering to minimise packet losses by TCP connections. In recent years this strategy (often referred to as *bufferbloat* [9], [10]) has been recognised as the source of hundreds to thousands of milliseconds of RTT inflation for all traffic during periods of congestion [11].

[4]CUBIC flows reduce by 30%, but NewReno-like flows still use 50%.

C. Online games: Latency and packet loss considerations

Multiplayer online games cover a wide range of game play styles and interactivity requirements [7]. For example, first person shooter (FPS) games are based around fast-paced inter-player interactions and rapid response times, which leads to observable player preferences for network RTTs under 100-150ms [12], [13]. Games where 'success' revolves around rapid in-game interactions (sometimes referred to as *twitch games*) frequently exhibit this preference for low network RTT. Massively multiplayer online (MMO) games have been seen to exhibit sensitivity to RTTs over 150-200ms [14]. Real-time strategy (RTS) games can exhibit greater RTT tolerance (such as the 500-800ms tolerance discerned by players of Warcraft3 in [15]) while players of turn-based games are not as concerned by the impact of typical network RTT.

There are no precise upper bounds on RTT for twitch games in general, as individual player experience depends as much on the in-game latency-mitigation techniques deployed by particular game developers [7]. But twitch games drive wide-spread consumer awareness of, and disappointment with, RTT inflation occuring on their home Internet service. In that respect they share common ground with interactive Voice over IP (VoIP) – user satisfaction degrades when RTT is inflated.

Sensitivity to packet loss is less well studied. Most latency-sensitive interactive services utilise UDP for transport, and rely on application-layer techniques to 'hide' the information missed when a packet is lost.[5] Games may interpolate between received updates (in either direction) to estimate what a client (or the virtual world) was doing during a missed update. Similarly, audio-visual codecs can interpolate between received samples to smooth over missing information.

However, this requires a play-out buffer of game update messages (or audio-visual samples) in order to detect, and hide, information losses without interrupting the continuity of in-game activities (or rendering audio-visual content). Such application-layer buffering adds latency to a participant's experience, and lowers the network RTT that can be tolerated if the participant's sense of immersion is to remain intact.

Or put succintly, we trade off tolerance for packet losses against tolerance for network RTT.[6] So it becomes important to consider how emerging network-layer RTT reduction schemes impact on packet loss rates.

D. Active Queue Management (AQM) to reduce queuing delay

Actively managing queue occupancy to reduce queuing delays and improve resource sharing has been around since the 1990s [16], with random early detection (RED) [17] being one of the first widely-considered, Internet-oriented approaches. In recent years the Internet Engineering Task Force (IETF) has re-focussed interest on three new AQM schemes – PIE (Proportional Integral controller Enhanced [1]), CoDel (Controlled Delay [2]) and FQ-CoDel (FlowQueue-CoDel [3]). FQ-CoDel has also inspired a FreeBSD implementation of FQ-PIE [4], although there is no related standardisation effort in the IETF yet. All four schemes include burst-tolerance to improve handling of new TCP connections in slow-start. A summary of work studying the performance of different AQMs for a variety of traditional applications can be found in [18].

1) PIE & CoDel: PIE and CoDel are both single-queue management strategies that drop packets, or use explicit congestion notification (ECN) marking of packets [6], when queuing delays persistently exceed a target delay T_{target}.

With a PIE queue, packets arriving within the first 150ms of an empty queue will be enqueued successfully. After this burst-tolerance period, arriving packets are enqueued or randomly dropped (or ECN marked) with a certain probability. This probability is periodically updated and is based on how much the current queuing delay[7] differs from $T_{target} = 15ms$ and whether this delay is going up or down. As a form of overload protection, even ECN-capable packets are dropped rather than marked when the dropping probability is $> 10\%$.

With a CoDel queue, drop/mark decisions are made when a packet is dequeued for transmission. CoDel tracks the (local) minimum queuing delay experienced by packets in a burst tolerance interval (initially 100ms). Packets are neither dropped nor ECN marked while the minimum queuing delay is less than $T_{target} = 5ms$ or the buffer contains less than one full-size packet. If the minimum recent queuing delay exceeds T_{target}, CoDel enters the dropping state – a packet is dropped/marked and a 'next drop time' is set, at which time CoDel will again drop/mark a packet if the queue has sustained a queuing delay above T_{target}. At successive drops, 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 queueing delay is below T_{target} again, CoDel exits the dropping state.

2) FQ-CoDel & FQ-PIE: By default FQ-CoDel classifies flows into one of 1024 independent CoDel-managed queues by hashing the 5-tuple of IP protocol number and source and destination IP and port numbers. The "FlowQueue" aspect of FQ-CoDel involves a modified deficit round-robin (DRR) scheme. Each queue can dequeue up to a quantum of bytes (one MTU by default) per iteration, and priority is given to queues with newly-arrived packets from "sparse" flows.[8]

Latency reductions occur through CoDel on individual queues, while the DRR scheduling provides relatively even capacity sharing between flows hashed into separate queues.

The authors of PIE mention as an aside in section 6 of RFC8033 [1] that a FlowQueue-like variant of PIE is conceivable. FreeBSD's FQ-PIE [4] implements this suggestion, using FQ-CoDel's FlowQueue strategy for distributing flows

[5]TCP is usually avoided as the application has no control over the time (potentially many seconds) that TCP can stall while recovering lost packets.

[6]A game designer *could* instead send N duplicates of every game update message, and hope one of the N always gets through. But this increases the game's required network bandwidth by N, making it an unattractive strategy.

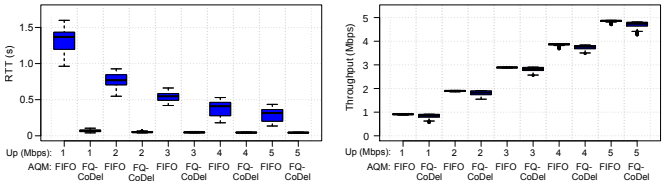
[7]Based on the queue length and recent estimates of dequeue rate.

[8]Flows with packet arrival rate small enough that a new queue is assigned to them upon packet arrival, which includes transactional flows such as DNS lookups and TCP connection establishments.

across queues while using PIE rather than CoDel to manage individual queues.

E. AQM deployment – how and why

Modern AQM is slowly gaining traction in different markets. For example, a variant of PIE is mandatory for cable modem service based on DOCSIS 3.1 [19], [20], FQ-PIE is available in FreeBSD 10.x onwards, and PIE, CoDel and FQ-CoDel are available from the late 3.x series Linux kernel onwards. Embedded Linux has a strong presence in the home gateway market, so vendors can offer AQM once their products move from 2.x to 3.x-based Linux kernels. Technically-adept owners can already upgrade a variety of consumer gateways using dedicated Linux distributions such as OpenWRT [21].



(a) RTT: Very low over FQ-CoDel, very high over FIFO

(b) Throughput: Slightly lower under FQ-CoDel compared to FIFO

Figure 2: A single upstream TCP flow through a FIFO or FQ-CoDel bottleneck, $RTT_{base} = 40ms$, link speeds 1-5Mbps

Ideally both ends of the last-mile will be upgraded, but upgrades need not be simultaneous. Figure 2[9] illustrates the major motivation for replacing FIFO with AQM in the upstream. A hypothetical home user is using a TCP-based application to push data to offsite storage location 40ms away ($RTT_{base} = 40ms$). With a FIFO gateway having 180 packets of buffering, Figure 2a shows upstream congestion resulting in median RTTs for any interactive traffic sharing the gateway at the same time ranging from ∼1400ms down to ∼300ms as the upstream rate is varied from 1Mbps to 5Mbps. Switching to FQ-CoDel results in RTTs well below 100ms even at only 1Mbps upstream. Figure 2b shows minimal loss of TCP throughput relative to using FIFO in order to gain this significant reduction in RTT.

F. Packet loss – collateral damage from deploying AQM

Aside from their initial burst tolerance, PIE and CoDel (or FQ-{CoDel,PIE} for flows hashed into the same queue) effectively instantiate 'short' queues (both in terms of milliseconds and bytes). Consequently, TCP flows will experience more capacity probing cycles (congestion epochs), and trigger more congestion notifications per unit time, than would occur over conventionally sized (larger than BDP) FIFO queues [19].

This is a problem because much of today's Internet still requires congestion signalling via packet loss (with ECN yet to see significant deployment). All packets who *will* experience (PIE) or *have* experienced (CoDel) too much delay in a queue are candidates to be dropped, regardless of whether they belong to the flow(s) actually causing the congestion. And given that PIE and CoDel make drop decisions on a per-packet basis, a flow sending small packets is just as likely to be hit as a flow sending a similar number of large packets per second, despite the latter flow being the dominant cause of queue build-up.

The price for significantly reduced RTT is that relatively low-bitrate interactive flows are likely to experience collateral damage in the form of increased packet loss rates. In the rest of this paper we (a) explore and characterise the degree to which packet loss rates under PIE and CoDel alone may negatively impact on multiplayer game traffic (as an illustrative example), (b) show why FlowQueue variants of AQM should be promoted for real-world deployment, and (c) explore the impact of enabling ECN more widely.

III. Experimental Methodology

We have chosen to explore the impact of AQM and ECN on game traffic packet loss rates using a representative subset of plausible real-world network conditions. We describe our experimental testbed in detail sufficient to assist others in reproducing (and extending) our results.

A. A simplified use-case scenario

Our primary simplification is to focus on one significant scenario: An online twitch game competing with elastic traffic in the upstream direction.

Using *X/Y Mbps* to indicate X Mbps downstream and Y Mbps upstream, we consider a hypothetical last-mile offering 15/1Mbps or 15/5Mbps service.[10] Queue management at each end is FIFO (common industry default), PIE, CoDel or FQ-{CoDel,PIE}. We repeat each combination with unloaded RTT (RTT_{base}) varying from 20ms to 100ms.

In Section V we discuss how this simple case provides enough insight into the impact of AQM on induced packet losses that readers can extrapolate to cases with more or less competing TCP flows, flows in the opposite direction, ON-OFF elastic flows (such as streaming services) and different upstream or downstream last-mile bandwidths.

B. Experimental testbed and traffic sources

Figure 3 shows our physical testbed topology based on [23] and controlled by a publicly available tool called TEACUP [24], [25].[11] The bottleneck router provides configurable rate shaping, AQM and RTT_{base} between 172.16.10.0/24 (home) and 172.16.11.0/24 (internet) using FreeBSD 10.2 for FQ-PIE experiments and openSUSE 12.3 Linux (kernel 3.17.4) for PIE, CoDel and FQ-CoDel experiments.[12]

The competing elastic traffic consists of two overlapping NewReno TCP flows generated by two iperf sessions started one second apart between hosts running FreeBSD 10.2-RELEASE-p7.

[9]Based on experimental results in Figure 3 of [22].

[10]The former representing plausible ADSL2+ service, while the latter a plausible low-end fibre-to-the-home service offering.

[11]TEACUP enables repeatable network performance experiments where configurable traffic sources congest an emulated network path having a specified range of RTT_{base}, bottleneck bandwidth and bottleneck AQM.

[12]See section IV-B of [25] for details on how TEACUP appropriately configures netem and tc for this purpose.

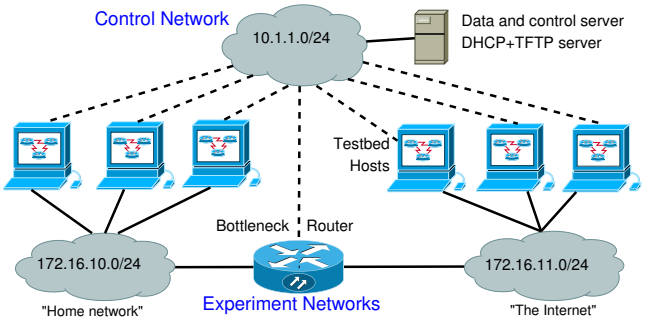


Figure 3: TEACUP testbed to emulate mixed-traffic over last-mile between home and Internet

Game traffic is emulated using pktgen v0.3.1[13], which uses a statistical model [26] to synthesise *server to client* (s2c) and *client to server* (c2s) Quake III Arena traffic for different numbers of players. In our experiments we emulate one client participating in a four-player game. The traffic is two streams of small UDP packets, averaging around 20 packets per second in the s2c direction and 95 packet per second in the c2s direction.

For FIFO experiments we configure the upstream bottleneck with 40 packets of buffering (approximately BDP of a 5Mbps/100ms path) and the downstream with 200 packets of buffering (in excess of 15Mbps/100ms BDP). We allow CoDel, PIE and FQ-{CoDel,PIE} to grow their queues to 1000 packets (per IETF recommendations).

IV. Experimental Results

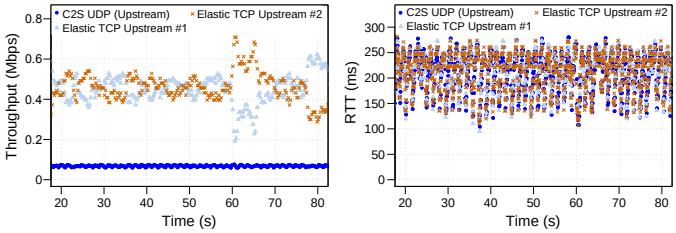
Each experiment involves two-way game traffic running for 20 seconds before two TCP flows begin pushing data from Figure 3's home to Internet (i.e. *upstream*). The elastic flows then last for roughly 80 seconds. During this period we look closely at the impact on c2s packet loss rates, the upstream throughput achieved by the elastic TCP flows, and the overall RTT experienced by all three application flows during the period of congestion.

A. Latency and throughput with FIFO

Figure 4 shows the impact of upstream congestion on our 15/1Mbps last-mile using FIFO queue management. In this case $RTT_{base} = 20ms$, emulating a case where the unloaded RTT to the game server is only 20ms (excellently close for interactive play) and the elastic TCP flows are pushing data to 'cloud' servers also only 20ms away.

In Figure 4a the low-rate c2s game flow weaves itself between the 1500byte TCP data packets as the two iperf flows somewhat noisely share the available 1Mbps. Unfortunately, Figure 4b, reveals that despite $RTT_{base} = 20ms$ all three flows are experiencing RTTs ranging from just under 150ms to just over 250ms (swinging rapidly in time with the cyclical capacity probing of each TCP flow).

[13]http://caia.swin.edu.au/bitss/tools/pktgen-0.3.1.tgz



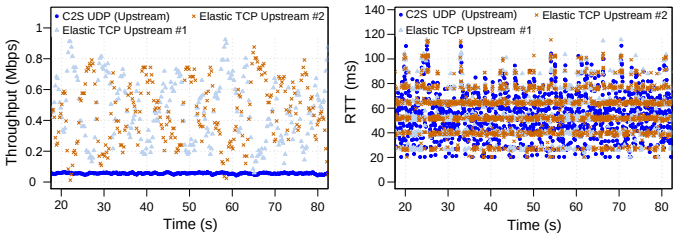
(a) Throughput (b) RTT

Figure 4: Upstream congestion of a 15/1Mbps link using FIFO bottleneck, $RTT_{base} = 20ms$

B. Latency and throughput with AQM (no ECN)

Now we look at the experience of each flow when running through a CoDel, PIE or FQ-CoDel bottleneck.

1) Using single-queue CoDel: Although the IETF does not recommend CoDel be deployed on its own, we use Figure 5 to illustrate the potential impact of a 15/1Mbps last-mile using CoDel when the three flows have $RTT_{base} = 20ms$. Figure 5a reveals the two iperf flows sharing the available 1Mbps far more chaotically than for the FIFO case in Figure 4a. On the upside, Figure 5b demonstrates a significant improvement in RTTs experienced by all three flows, ranging from 20ms to 80ms with intermittent peaks to around 100ms.



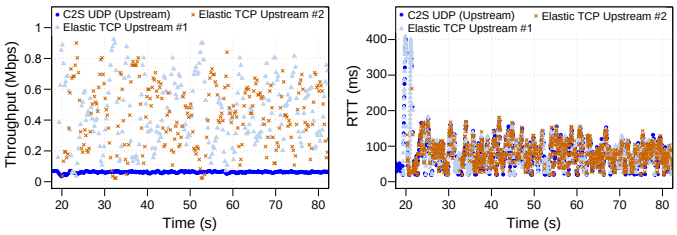
(a) Throughput (b) RTT

Figure 5: CoDel during upstream congestion, 15/1Mbps, $RTT_{base} = 20ms$

The noticeable banding of elastic flow RTTs in Figure 5a is because at 1Mbps enqueuing a 1500 byte TCP packet causes queuing delay to jump in increments of 12ms. This also causes trouble for CoDel, as a single full-size packet immediately exceeds CoDel's default $T_{target} = 5ms$, yet CoDel wont drop packets unless there's at least one full-size packet in the queue.

2) Using single-queue PIE: The IETF does propose PIE for use in single-queue scenarios. So in Figure 6 we illustrate the potential impact of a 15/1Mbps last-mile using PIE when all three flows have $RTT_{base} = 20ms$. Figure 6a reveals chaotic capacity sharing between the two iperf flows reminiscent of CoDel in Figure 5a. Figure 6b shows a reduction in RTT relative to FIFO (Figure 4b), but not as good as the reduction provided by CoDel (Figure 5b). Aside from the initial RTT spike (due to PIE's burst tolerance as the iperf flows begin) all three flows experience RTTs ranging from 20ms to around 150ms.

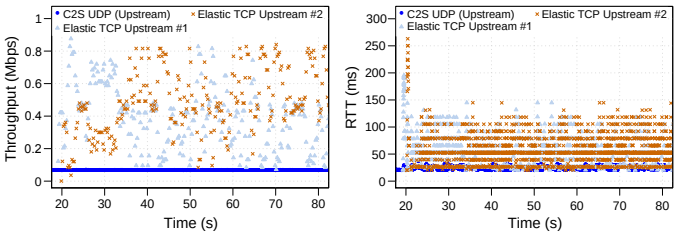
3) Using multi-queue FQ-CoDel: The IETF recommends FQ-CoDel as the way to introduce CoDel's benefits into a net-



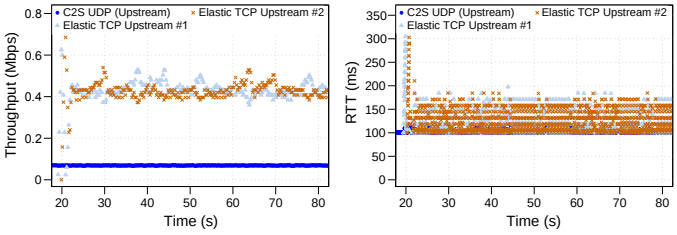
(a) Throughput (b) RTT

Figure 6: PIE during upstream congestion, 15/1Mbps, $RTT_{base} = 20ms$

work. Figure 7 shows the potential impact of a 15/1Mbps last-mile using FQ-CoDel when all three flows have $RTT_{base} = 20ms$ or $RTT_{base} = 100ms$. With $RTT_{base} = 20ms$ the chaotic bandwidth sharing (Figure 7a) and RTT spread (Figure 7b) experienced by the iperf flows is similar to single-queue PIE and CoDel. However, a significant difference is that the c2s game flow now experiences consistent throughput and RTT that is significantly lower than that of the iperf flows.



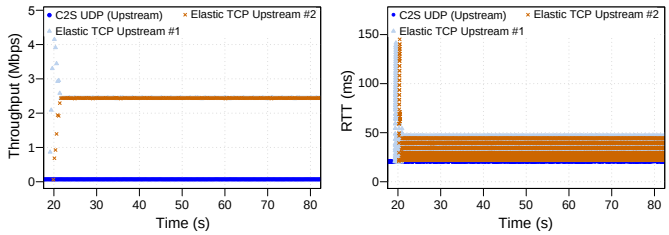
(a) Throughput, $RTT_{base} = 20ms$ (b) RTT, $RTT_{base} = 20ms$



(c) Throughput, $RTT_{base} = 100ms$ (d) RTT, $RTT_{base} = 100ms$

Figure 7: FQ-CoDel during upstream congestion, 15/1Mbps

With 15/1Mbps and $RTT_{base} = 100ms$ the iperf flows see far better bandwidth sharing (Figure 7c) and slightly tighter RTT spread (Figure 7d). The c2s game flow again experiences consistent throughput and RTT close to RTT_{base}.



(a) Throughput, $RTT_{base} = 20ms$ (b) RTT, $RTT_{base} = 20ms$

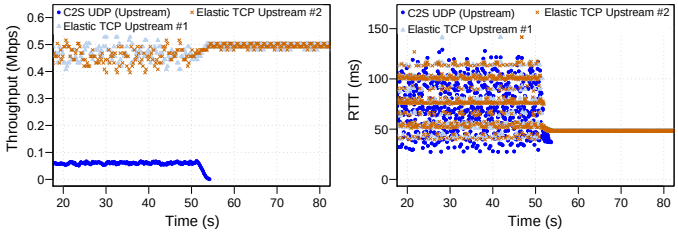
Figure 8: FQ-CoDel during upstream congestion, 15/5Mbps

FQ-CoDel is far better at flow isolation and bandwidth sharing as the bottleneck bandwidth increases. Figure 8 shows the impact of FQ-CoDel on a 15/5Mbps last-mile and $RTT_{base} = 20ms$. There is almost perfect bandwidth sharing between the iperf flows (Figure 8a) and a much tighter RTT spread (Figure 8b). The c2s game flow again experiences consistent throughput and RTT close to RTT_{base}.

C. Latency and throughput with AQM and ECN enabled

ECN is usually promoted as a less blunt instrument for congestion notification. Enabling ECN in our testbed allows the two elastic TCP flows to have their packets marked rather than dropped. As ECN does not apply to UDP flows, when a game packet is selected by the AQM it will be dropped.

Broadly speaking ECN made little difference to the RTTs and throughputs experienced by the iperf and game flows under most of the scenarios in Section IV-B except one: CoDel with $RTT_{base} = 20ms$ at 15/1Mbps. Figure 9 shows a dramatic failure mode – about halfway through the experiment CoDel would begin dropping 100% of game traffic.



(a) Throughput, $RTT_{base} = 20ms$ (b) RTT, $RTT_{base} = 20ms$

Figure 9: Dramatic failure mode with CoDel, ECN, 15/1Mbps and $RTT_{base} = 20ms$

This failure mode has its roots in CoDel struggling to effect control over the two iperf flows. As noted earlier, at 1Mbps a 1500 byte TCP packet is 12ms long, immediately exceeding CoDel's default $T_{target} = 5ms$. With ECN disabled, CoDel applies drops to the TCP flows and the traffic abates sufficiently from time to time that we end up with Figure 5. With ECN enabled, the TCP flows experience marks rather than drops, and the load on the CoDel bottleneck never abates. Direct inspection of packets captured during Figure 9's experiment reveal that at a certain point in time CoDel finds itself needing to flag congestion on all packets passing through – every TCP packet was ECN marked, and as collateral damage every game packet was dropped.

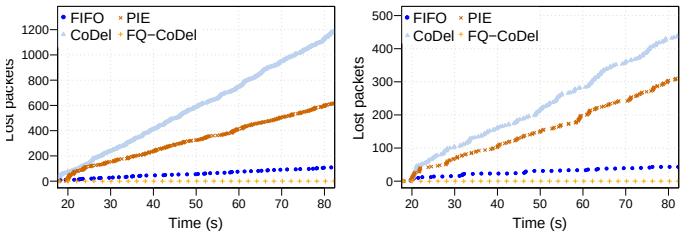
D. Loss vs time, with and without ECN

The preceding results illustrate what is already well known – deploying single-queue PIE or multi-queue FQ-CoDel is a good way to reduce (or close to eliminate) the RTT inflation experienced when elastic TCP flows compete with game traffic over FIFO bottlenecks. However, as noted in Section II-F we must also investigate the impact of AQMs on packet loss rates for the game traffic.

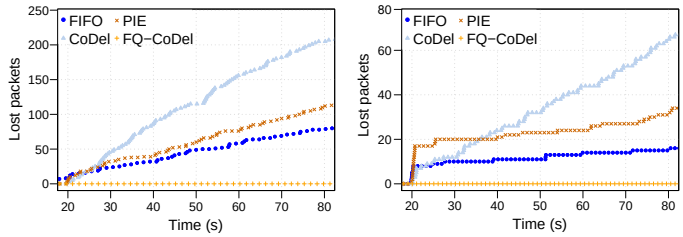
Figure 10 directly compare the cumulative packet loss vs time experienced by c2s traffic during the upstream congestion

periods using FIFO, CoDel, PIE and FQ-CoDel bottlenecks of $15/\{1,5\}Mbps$ last-miles, $RTT_{base} = \{20, 100\}ms$ paths and no ECN. Using FQ-PIE resulted in similar results to FQ-CoDel, so for clarity we only plot the FQ-CoDel results.

Recall that the c2s traffic is a client sending roughly 95 packets per second, and these graphs span a period of 60 seconds. So losing 600 - 1000 packets (PIE and CoDel in Figure 10a) means losing 10 - 17 packets per second, or roughly 11%-17% of the game updates being sent from game client to game server.



(a) 15/1Mbps, $RTT_{base} = 20ms$ (b) 15/1Mbps, $RTT_{base} = 100ms$



(c) 15/5Mbps, $RTT_{base} = 20ms$ (d) 15/5Mbps, $RTT_{base} = 100ms$

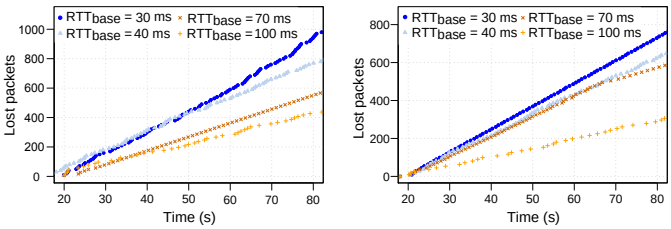
Figure 10: Cumulative c2s packet loss vs time during upstream congestion, no ECN

Three things are evident:

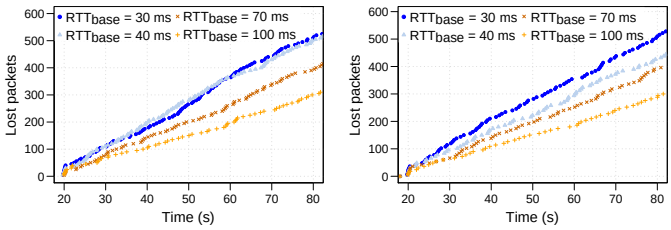
1) Both PIE and CoDel introduce substantially higher packet loss rates than FIFO,
2) FQ-CoDel (and FQ-PIE, not shown) *never* loses a packet during the upstream congestion period, and
3) Support for the intuition in Section II-F that as RTT_{base} rises there are fewer TCP capacity probing cycles per unit time triggering fewer periods of packet loss.

Figure 11 shows in more detail how cumulative c2s flow packet loss vs time drops off with increasing RTT_{base} for CoDel and PIE cases, both without and with ECN enabled. Packet loss is generally constant across time, consistent with it driven by cylical (and recurring) queue filling by the competing TCP flows. The impact can be seen as small imperfections in the c2s throughput plotted in Figures 5a and 6a (which is based on the bytes/second *received* at the game server).

A PIE or CoDel bottleneck introduces the highest packet loss rates when RTT_{base} is low for competing TCP flows – the sort of network conditions that would otherwise be considered ideal for twitch gaming and other interactive applications. Broadly speaking, when using CoDel or PIE, enabling ECN for TCP flows does not make a significant difference to the packet loss rates experienced by our game traffic during periods of congestion. With FQ-{CoDel,PIE} the choice of



(a) CoDel (no ECN) (b) CoDel + ECN



(c) PIE (no ECN) (d) PIE + ECN

Figure 11: Cumulative c2s packet loss vs time during upstream congestion, 15/1Mbps, CoDel & PIE, varying ECN and RTT_{base}

ECN is irrelevant, as FQ-{CoDel,PIE} protects the game traffic flow in either case.

V. Discussion

Here we discuss some implications of the presented work, and how much of it can be applied more generally.

A. Use a FlowQueue-based AQM

As noted in Section II-F, PIE and CoDel make drop decisions on a per-packet basis. So a flow sending X small packets per second is just as likely to be hit as a flow sending X large packets per second at the same time, despite the latter flow likely being the dominant cause of queue build-up.

The CoDel authors note that it is really intended to be used in conjunction with something like the FlowQueue scheduler in FQ-CoDel. On the other hand, discussions around PIE typically present it as being suitable for deployment in single-queue scenarios. This position may be reasonable if we assume all flows sharing a bottleneck are intrinsically loss-tolerant, or can tolerate the additional latency introduced by loss-mitigation techniques at the transport or application layers (as noted in Section II-C).

Our results saw 11%-17% packet loss rates using CoDel and PIE and 0% loss rates using FQ-{CoDel,PIE}. The results for FQ-{CoDel,PIE} are entirely due to the FlowQueue scheduling rather than the CoDel or PIE queue management (as CoDel and PIE were also shown to have great difficulty managing TCP flows at 1Mbps and low RTT_{base}).

So our counter-view is that any addition of AQM to a commercial gateway or router ought to provide a FlowQueue-based option, which means FQ-CoDel for today's Linux-based embedded systems. And service providers ought to enable FQ-{CoDel,PIE} rather than any single-queue AQM (provided their implementation has enough queues to minimise flow-hashing collisions).

B. ECN is neither a help or a hindrance

Turning on ECN is arguably a positive thing for managing TCP flows. However, it makes little difference to the loss rates experienced by low data rate applications using UDP. We still recommend FlowQueue scheduling, with or without ECN.

C. AQM on the downstream and different bandwidths

Our results are equally applicable to the downstream, where bottleneck AQM needs to be deployed on the ISP side of the last-mile link. Although downstream speeds are usually higher, congestion still occurs when the aggregate inbound traffic exceeds the downstream speed offered by the ISP (either due to technical limits or contractual agreement with the customer).

The same basic issues are at play – during periods of downstream congestion (by small or large numbers of concurrent flows) there will be signficant packet loss events induced by single-queue AQM such as PIE, and virtually no packet loss events by multi-queue AQM solutions such as FQ-CoDel.

D. Competition with other application types

Our recommendations are not limited solely to cases where game traffic competes with long-lived TCP flows.

As noted in Section II-A, modern streaming protocols like DASH effectively create a continuous ON-OFF sequence of elastic traffic bursts. During an ON period (which can last fractions of seconds to multiple seconds) the game traffic will again find itself competing with a loss-sensitive TCP flow consisting of full-sized packets quickly filling up the bottleneck queue. If forced into a single queue, the game traffic will experience packet drops as a side effect of the AQM attempting to manage the congestion caused by the TCP flow active at the time.

Game traffic will benefit from the use of a FlowQueue-based AQM anyhere that competition can exist with short-lived, long-lived or repetitious elastic traffic (loss-based congestion control and latency-tolerant). Such competing traffic will push the AQM into dropping/marking, and it is preferable for the game flows to be in their own queues during those times.

(Congestion due to too many constant bitrate flows, such as from multiple VoIP streams simultaneously traversing an under-provisioned upstream, is also certainly possible. But there is no silver bullet in such circumstances – while FQ-CoDel could isolate these flows from each other for a time, if none of them backoff then losses will occur.)

VI. Conclusions

Consumer internet services are expected to concurrently support a mix of interactive and non-interactive/bulk data transfer services. Industry interest in reducing network latency is leading to the promotion and deployment of AQM schemes such as CoDel, PIE and FQ-CoDel. However, although these AQM schemes are well known to reduce bottleneck queuing delays, little detailed analysis exists on how such schemes impact on packet loss rates of interactive traffic flows.

We have experimentally emulated a congested home broadband service using FIFO, CoDel, PIE and FQ-{CoDel,PIE} queue management schemes to explore interactions between elastic, TCP-based bulk data transfers and traffic typical of low-rate, UDP-based interactive applications. Our results demonstrate that single-queue AQM schemes, such as CoDel and PIE, create significantly increased levels of packet loss for low-rate UDP traffic during periods of competition with elastic, TCP-based bulk data transfers. Additionally, enabling ECN for the TCP flows does not materially improve the packet loss rates experienced by any competing game flows or other interactive traffic.

However, we also show that FlowQueue variants, such as FQ-CoDel and FQ-PIE, provide significant protection for interactive traffic flows, allowing for almost zero packet loss during periods of competition with otherwise bandwidth-hungry TCP flows. We observe that applications such as game traffic will benefit from the use of a FlowQueue-based AQM anyhere that short-lived, long-lived or repetitious elastic traffic may compete and push the AQM into dropping/marking. During such periods of competition it is preferable for the game flows to be in their own queues.

We conclude that any addition of AQM to a commercial gateway or router ought to provide a FlowQueue-based option, which today means FQ-CoDel. Service providers ought to enable FQ-CoDel rather than any single-queue AQM if they wish to better serve online game players and users of other interactive applications. Development and evaluation of FQ-PIE implementations would also likely be advantageous for the Internet community.

Acknowledgements

The work was also supported in part by a 2014-2016 Swinburne University of Technology / Cisco Australia Innovation Fund project titled “An evaluation of household broadband service requirements for educational innovation and Internet of Things”.

References

[1] 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

[2] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, “Controlled Delay Active Queue Management,” IETF Draft, Mar 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-codel-07

[3] T. Høeiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, “The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm,” IETF Draft, Mar 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06

[4] 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

[5] M. Allman, V. Paxson, and E. Blanton, “TCP Congestion Control,” RFC 5681, Internet Engineering Task Force, Sep. 2009. [Online]. Available: https://tools.ietf.org/html/rfc5681

[6] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” RFC 3168, Sep. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3168

[7] G. Armitage, M. Claypool, and P. Branch, *Networking and Online Games - Understanding and Engineering Multiplayer Internet Games*. UK: John Wiley & Sons, 2006.

[8] "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

[9] 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: http://dx.doi.org/10.1145/2063166.2071893

[10] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, Internet Engineering Task Force, Jul 2015. [Online]. Available: https://tools.ietf.org/html/rfc7567

[11] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298313

[12] G. Armitage, "An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3," in *11th IEEE International Conference on Networks (ICON 2003)*, Sydney, Australia, 28-1 September 2003, pp. 137–141. [Online]. Available: http://dx.doi.org/10.1109/ICON.2003.1266180

[13] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," in *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '04. New York, NY, USA: ACM, 2004, pp. 144–151, available at http://doi.acm.org/10.1145/1016540.1016556. [Online]. Available: http://doi.acm.org/10.1145/1016540.1016556

[14] K. Chen, P. Huang, and C. Lei, ""How Sensitive Are Online Gamers to Network Quality?"," *Commun. ACM*, vol. 49, no. 11, pp. 34–38, Nov. 2006, available at http://doi.acm.org/10.1145/1167838.1167859. [Online]. Available: http://doi.acm.org/10.1145/1167838.1167859

[15] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The Effect of Latency on User Performance in Warcraft 3," in *"Proceedings of the 2nd Workshop on Network and System Support for Games"*, ser. NetGames '03. New York, NY, USA: ACM, 2003, pp. 3–14, available at http://doi.acm.org/10.1145/963900.963901. [Online]. Available: http://doi.acm.org/10.1145/963900.963901

[16] R. Adams, "Active queue management: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1425–1476, Third 2013. [Online]. Available: http://dx.doi.org/10.1109/SURV.2012.082212.00018

[17] 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, Internet Engineering Task Force, Apr 1998. [Online]. Available: https://tools.ietf.org/html/rfc2309

[18] 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: http://dx.doi.org/10.1016/j.comnet.2015.07.014

[19] G. White, "Active queue management in DOCSIS 3.1 networks," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 126–132, March 2015.

[20] G. White and R. Pan, "Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems," RFC 8034, Internet Engineering Task Force, Feb. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8034

[21] (2017) OpenWrt. [Online]. Available: https://openwrt.org

[22] G. Armitage, J. Kennedy, S. Nguyen, J. Thomas, and S. Ewing, "Household internet and the 'need for speed': evaluating the impact of increasingly online lifestyles and the internet of things," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 170113A, 13 January 2017. [Online]. Available: http://caia.swin.edu.au/reports/170113A/CAIA-TR-170113A.pdf

[23] S. Zander and G. Armitage, "CAIA Testbed for TEACUP Experiments Version 2," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 150210C, 10 Feb 2015. [Online]. Available: http://caia.swin.edu.au/reports/150210C/CAIA-TR-150210C.pdf

[24] ——, "TCP Experiment Automation Controlled Using Python (TEACUP)," May 2015. [Online]. Available: https://sourceforge.net/projects/teacup

[25] ——, "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

[26] A. Cricenti and P. Branch, "A Generalised Prediction Model of First Person Shooter Game Traffic," in *34th IEEE Conference on Local Computer Networks (LCN 2009)*, Zurich, Switzerland, 20-23 October 2009, pp. 213–216. [Online]. Available: http://dx.doi.org/10.1109/LCN.2009.5355165