

# Priority Queuing of Network Game Traffic over a DOCSIS Cable Modem Link

Jason But, Shaun Burriss and Grenville Armitage

Centre for Advanced Internet Architectures

Swinburne University of Technology

Melbourne, Australia

Email: {jbut,4150996,garmitage}@swin.edu.au

**Abstract**—Despite the greater bandwidth available for home Internet users, network problems continue to surface with regard to the mix of network applications being run. The ease in which Internet sharing can be configured, coupled with the increase in the number of computers present in consumer residences, increase the likelihood that concurrent network applications will have to compete for access to a limited bandwidth resource. The performance of applications such as network games and VoIP is heavily dependent on transmission delays and packet loss rates, which can be improved through priority queuing at the Internet Access device to minimise the impact of queuing delay. In this paper we compare the performance of both regular TCP traffic and network game traffic competing for Internet bandwidth on a DOCSIS Cable Modem link to quantify the performance gain and loss of both applications in an environment where priority queuing is enabled for game traffic. Our results show that the decrease in queuing delay and packet loss rate experienced by game flows is significant in the case of prioritised queuing while the un-prioritised TCP flows experience a slight drop in network performance.

## I. INTRODUCTION

With the recent growth in adoption of broadband access technologies such as Asymmetric Digital Subscriber Loop (ADSL), Cable Internet and 802.11x, it has become more practical to run multiple networked applications concurrently. Sharing an Internet connection between a number of users or computers in a single residence has been simplified by the availability of this feature on most Internet access devices (modems) and/or computer Operating Systems.

Internet sharing amongst multiple computers at the same site leads to contention for available network resources, particularly for the bottleneck which is the available upstream and downstream bandwidth. A common example would include a user on one PC playing an online game while a second PC is used to either browse the web or transfer large files. While TCP based applications can use TCPs inherent congestion control to share bandwidth amongst multiple concurrent streams, real-time applications place different requirements on the network. The impact of contention for the bottleneck bandwidth will impact these types of applications differently.

Highly interactive network games, such as the very popular first-person shooter (FPS) genre, place different requirements on the network compared to traditional non-interactive applications (e.g. web, email). For these games the user's ability to play successfully greatly depends on the Quality of Service

(QoS) present in the network. Recent studies found that with degrading network conditions (increasing delay, jitter, packet loss) users play less effectively (e.g. they achieve a lower score) and their perceived quality of the game play significantly decreases (see [1]–[4]).

Voice over IP (VoIP) is another application with similar characteristics and network requirements to games. While the generated traffic is low bandwidth, the application is very sensitive to transmission time and queuing delays. While games players display decreased performance under adverse network conditions, VoIP applications result in broken-up audio streams.

One approach to offer improved network QoS to network games is to use priority queuing to minimise the queuing delay that the game traffic experiences. Others have proposed schemes to automatically detect game and other delay sensitive traffic and automatically reconfigure CPE devices to prioritise these flows [5]. What is unclear is the benefits that can be gained from prioritisation, and the effect on non-prioritised flows over typical home consumer links. In this paper we will quantify the effect of priority queuing of game traffic on both the game and other network flows as applied to a DOCSIS Cable Modem access network.

The paper is structured as follows. Section 2 discusses how Internet connections are shared in a consumer context and how this affects the performance of networked games. Section 3 describes our experimental configuration while Section 4 reports our results. Section 5 concludes and outlines future work.

## II. INTERNET SHARING IN THE CONSUMER CONTEXT

With the greater bandwidth afforded by broadband Internet Access technologies, there is greater scope for network applications to make concurrent use of the available Internet resources. This form of *time-sharing* allows better use of the services provided to the consumer. However it is important to note that Internet sharing does not necessarily mean multiple applications running concurrently on a single Internet connected computer. Most game-players would have experienced the importance of an uncongested network during game play and are unlikely to run network applications concurrently during these times.

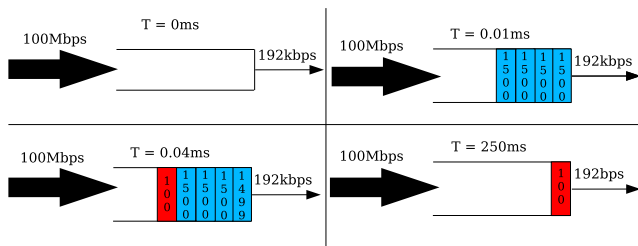


Fig. 1. Queues at a Bottleneck

Of more interest is an Internet connection shared amongst multiple users in a household. An example might include one user browsing the web, another reading and answering emails with potentially large attachments, a third using a peer-to-peer application such as BitTorrent to download a large file while a fourth is involved in an online game - each application running on a different computer within the residence but sharing a single access device. This scenario is becoming increasingly common with both the cost of computers decreasing and the ease in configuring Internet Access (CPE - Consumer Premises Equipment) devices to share their connection.

Under this sort of scenario, we now have multiple concurrent applications competing for the limited resources of the bandwidth constricted link to the ISP. Given that each computer will be connected to the CPE device using higher bandwidth network technologies - such as FastEthernet (100Mbps) - each application has the potential to generate a network traffic flow at a greater rate than the CPE device can serialise onto the upstream link (typically 64kbps - 2Mbps). The CPE becomes a network bottleneck, with the potential for queues to form as packets wait to be serialised upstream. The same problem exists in the downstream direction but with lesser effect since the majority of Internet Access links are asymmetrical.

#### A. Networked Games

The effects of network delay on online games have previously been studied - network game players often choose to play on servers that are located geographically close to the game player to minimise the impact of latency [2], [3]. As such, queuing at the CPE device can have a potentially disastrous effect on the quality of game play. The problem can often be exacerbated by the bursty nature of TCP flows - carrying web browsing and file transfer traffic.

Consider the scenario described in Figure 1 where concurrent applications share an upstream link with available bandwidth of 192kbps. A TCP based application can generate a burst of four TCP packets with a maximum MTU size of 1500 bytes, which are then queued at the CPE device to be serialised onto the upstream link. Immediately following this burst a network game generates a single 100 byte packet for transfer. Due to the burst of TCP packets, the game traffic is queued behind the TCP packets at the CPE device. By the time the TCP packets have been transmitted, the game traffic has suffered a queuing delay of 250ms - too long for the game

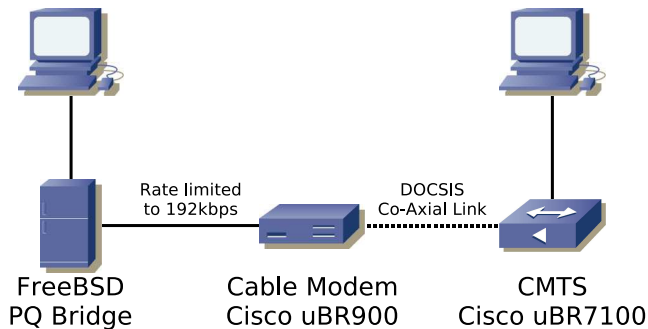


Fig. 2. Experimental Testbed

player to experience no drop in performance.

In this scenario the game player - although generating little actual traffic - experiences network conditions that make the game generally unplayable, while the nature of TCP ensures that the throughput witnessed by the TCP stream remains as high as possible given total requested bandwidth.

One possible solution is to prioritise the network game traffic such that all game traffic is pushed to the head of the queue and can then be serviced as soon as the current packet being serialised has completed. Queuing delay in our above example is maximised to the serialisation time of a 1500 byte packet - in this case about 62ms. In a more generic case the queuing delay is also impacted by the number of other prioritised game packets that have already been queued.

While this approach improves the perceived network performance for the game player, it is also important to ask the question of "What effect will this have on other network users?". In this paper we will attempt to quantify these effects - on both network game traffic and other network flows - over a DOCSIS Cable Modem link.

### III. EXPERIMENTAL CONFIGURATION

Our experimental testbed (see Figure 2) is BART [6], which consists of a real DOCSIS Cable Modem Network comprised of a Cable Modem Head End (Cisco uBR7100) and a series of Cable Modems (Cisco uBR900). A computer was placed at each end of the Cable Modem link to both generate and sink traffic flows across the network. To control the experimental procedure and the repeatability of the experiment, traffic was generated using **iperf** [7].

#### A. Traffic Generation

Two types of flows were generated. TCP traffic was generated to emulate regular network usage. Most non real-time network applications use TCP flows to transfer data, including web browsing, file transfer and peer-to-peer file sharing applications. We generate traffic flows in both directions, while the predominant transfer of data is downstream, broadband availability directly contributes to the amount of upstream bandwidth utilised:

- Greater upstream capacity means that sending large attachments in an email takes less time.

- File sharing applications like BitTorrent have become popular recently. These peer-to-peer applications are large generators of upstream traffic flows.

TCP traffic is bursty in nature but uses flow control mechanisms to share bandwidth usage and to regulate the number of packets generated. We use **iperf** to report the average throughput experienced by the generated TCP streams and this estimation of TCP RTT [8] to report the effect of queuing delay on the estimated RTT witnessed by the TCP flows.

Game traffic is emulated as a stream of UDP packets of fixed size and packet inter-arrival times using statistical values for Half-Life Counterstrike as reported in [9]. Of primary concern when evaluating the performance of the network for game flows is the experienced queuing delay - which directly impacts network latency and game performance - and the packet loss rate - too many dropped packets can result in an unplayable game [2], [3].

### B. Implementing Priority Queuing

It was not possible to implement priority queuing within the Cable Modem CPE device itself. As such we deployed a FreeBSD bridge between the consumer side traffic generator and the Cable Modem to implement prioritisation of game flows. A basic prioritisation service will not work as the bridge will simply prioritise traffic at the Ethernet line rate (100Mbps), resulting in no difference to the packet transmission order and queuing still occurring at the CPE device. It is necessary to forward traffic to the Cable Modem at the line rate available on the Cable Modem media, ensuring that all queuing - and prioritisation - takes place within the bridge, and that all packets are immediately serialised upon arrival at the Cable Modem.

Because the DOCSIS Cable Modem standard implements a shared upstream link, the available bandwidth at any given time is dependent on the demand of other Cable subscribers connected to the same CMTS. While our experimental configuration ensured that no other users were generating network traffic, DOCSIS works on a bandwidth request and allocation scheme where the Cable Modem requests a number of upstream slots based on the amount of traffic to send, the CMTS allocates a number of slots to the Cable Modem based on the bandwidth requested by all Modems on the network.

This is problematic since even in a network where only one modem is transmitting data, the allocated slots - and therefore available upstream bandwidth - is non constant. For our purposes we run a number of trials using flows of different rates to determine the maximum rate at which flows experience no queuing delay, and then rate limit our priority queuing bridge to this rate. This approach has two potential problems:

- Traffic is queued at the bridge when it could be possible to actually transmit it over the broadband link
- The bridge could forward packets to the Cable Modem faster than the currently allocated transmission slots, resulting in queuing at the Cable Modem - the very problem we are trying to resolve.

These problems can only be fully addressed by implementing priority queuing *within* the Cable Modem, thereby ensuring that all queuing takes place at the true bottleneck and that prioritised flows are subjected to minimal queuing delays.

Two queues were configured on the bridge. The prioritised queue queued all UDP flows (game packets) while all other traffic was queued in the non-prioritised queue.

### C. Procedure

The constant upstream available bitrate on our Cable Modem link has been determined as 192kbps, higher rates resulting in variable queuing delays within the network. Upstream bandwidth on many current consumer Internet plans is 256kbps for ADSL and up to 1Mbps for ADSL2 plans. However, cheaper plans still offer upstream rates of 64 or 128kbps.

As noted above, fixing the priority queuing bridge to this rate will occasionally result in packets being queued when they can be transmitted as well as the occasional packet being being queued at the modem. Bandwidth limiting was also employed in the base-line scenario to ensure a fair comparison with the priority queued case.

Under consideration is the scenario of concurrent data transfer using bi-directional TCP flows, this case is consistent with the use of peer-to-peer file sharing programs such as BitTorrent where data is transferred in proportionally equal amounts in both directions.

The initial base-line case involves one upstream and one downstream TCP flow running over the Cable Modem link. This will allow us to compare both the witnessed TCP RTT and the TCP throughput rate with the scenario where an additional network game flow is introduced.

The second scenario runs all three flows concurrently, one upstream TCP flow, one downstream TCP flow and one bi-directional emulated game flow. This scenario is run with the default configuration where packets are queued and transmitted in the order they are received at the bridge.

Finally we repeat the previous scenario except in this case the network game traffic flow is prioritised for transmission over the Cable Modem uplink.

## IV. RESULTS

In this section we outline our experimental results, and compare the baseline - no priority queuing - with the priority queued scenario. In each case we examine the impact the priority queuing had on both the game flow and the non-game TCP flows.

### A. TCP Traffic Only

In this case we merely rate limited the upstream link to 192kbps on the FreeBSD bridge, all received packets are queued here and transmitted at the lower rate upstream. The downstream link is unchanged.

In the first instance we generated a single downstream TCP flow to find a reported throughput by **iperf** of 2.7Mbps. TCP

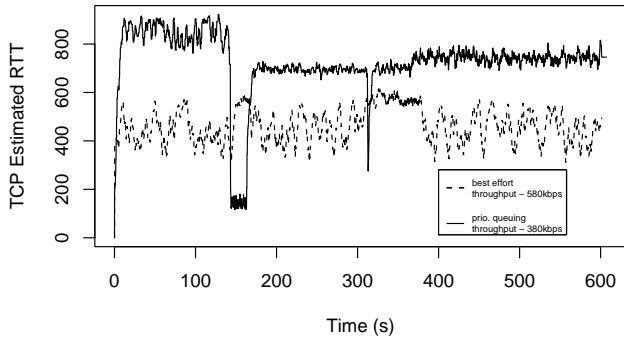


Fig. 3. Downstream TCP Throughput and Estimated RTT

throughput is primarily determined by the rate of acknowledgements being returned upstream and the growth in the TCP window size at the sender.

Running a TCP upstream flow concurrently with the downstream flow caused downstream throughput to drop to 590kbps. The primary cause in this large drop in data throughput is the bandwidth asymmetry of the Cable Modem link coupled with the bursty nature of upstream traffic. Packets being generated for the upstream flow tend to be generated in bursts which are then queued for transmission at the lower line rate of 192kbps. Acknowledgements for the downstream flow are queued behind these packets and must wait for access to the upstream link. This effect can be seen through inspection of the timestamps and ACK sequence numbers of upstream packets and comparing them with corresponding downstream data packets and their timestamps. The delay in transmission of TCP ACKs result in a larger estimated RTT for the downstream flow, slower adjustment of window sizes, and a more frequent retransmission of packets - resulting in reduced throughput. The presence of a single upstream flow has effectively cut the maximum downstream throughput to 25% of its nominal amount.

In comparison, the upstream flow Acknowledgements are not as severely affected as since there is higher capacity on the downstream link to support this flow. As such queuing is minimised, estimated RTT is low and the window size can grow to support the available bandwidth on the upstream link.

Internet access technologies offer asymmetric bandwidth plans since downstream data flow is typically larger than upstream flows. Our data indicates that this asymmetric approach would tend to decrease the potential utilisation of these lower bandwidth links. Even if the bandwidth in each direction were increased, the asymmetry would allow a flow on the lower bandwidth channel to effectively steal bandwidth from the higher bandwidth direction.

### B. Adding a Game Flow

We then added a simulated game flow to the Cable Modem link, and obtained results where the game flow was both prioritised in terms of access to the Cable Modem upstream link and where it was treated as best-effort.

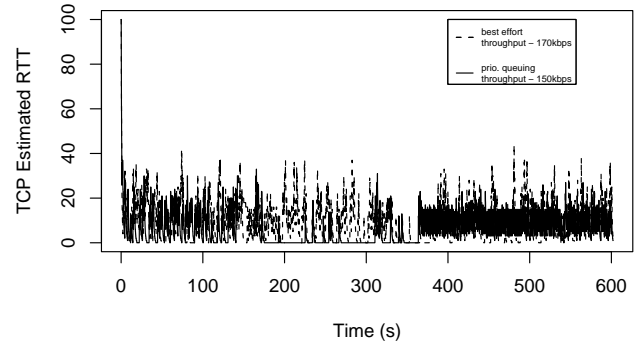


Fig. 4. Upstream TCP Throughput and Estimated RTT

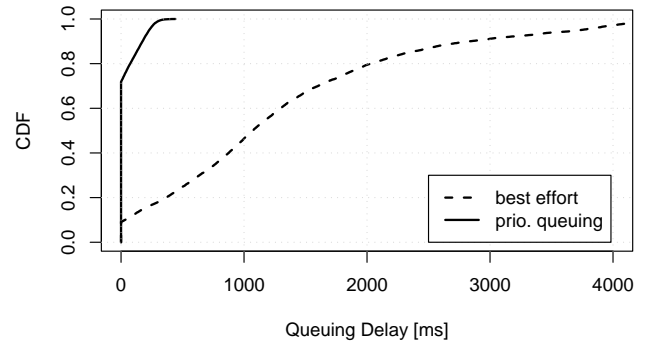


Fig. 5. Game Traffic Queuing Delay - CDF

Comparison between the prioritisation and best-effort scenarios are shown in Figures 3, 4 and 5. Figure 3 compares downstream TCP RTT and throughput. Figure 4 compares upstream TCP RTT and throughput. Figure 5 compares the cumulative distribution function (cdf) of queuing delays suffered by the game flow packets in the upstream direction. The key performance indicators include:

- Downstream data transfer reported throughput dropped from 580kbps (best-effort) to 380kbps (prioritised game traffic)
- Downstream TCP estimated RTT increased from a mean of 464ms to a mean of 724ms
- Upstream data transfer reported throughput dropped from 170kbps to 150kbps
- Upstream TCP estimated RTT stayed relatively constant with a mean of about 10ms
- Game traffic packet loss rate improved significantly from 28% dropped packets to 0.06%
- The queuing delay experienced by network game packets improved significantly. For the best-effort scenario, of the 72% of packets that were not dropped - half of all packets experienced a delay of at least 1072ms (mean 1294ms). Following prioritisation, this improves to a mean delay of 39ms with 75% of all packets experiencing a queuing delay of less than 29ms

The benefits of game traffic prioritisation are clear, a best-effort service would render the game unplayable when the Internet was being used by others in the same residence. The improvement in network performance following prioritisation would result in improvements in not only game quality, but also the performance of the player in game statistics.

Of slight concern is the decrease in performance of TCP traffic with prioritisation of the network game traffic, particularly the case of downstream data transfer. The primary cause of variation is TCP throughput and witnessed RTT is caused by the presence of upstream data transfer, in the case for no upstream transfer there is no contention for access, ACK packets experience little or no queuing delay. Correspondingly, estimated RTT is low and the TCP flow control protocol can push the throughput up to higher values.

We previously saw throughput drop to approximately 25% upon the introduction of an upstream TCP flow. The presence of upstream traffic of a bursty nature caused queues to be created in the Cable Modem, effectively increasing the queuing delay for downstream ACKs and subsequently the RTT. The net effect is an overall decrease in TCP throughput. Addition of a network game flow further dropped TCP throughput by 10kbps. This drop is minimal, since the traffic generated by the network game flow amounted to about 50 small packets every second - the effect of this on downstream ACK queuing delay is minimal.

With prioritisation enabled, the game flow has a much greater effect, game packets are now prioritised and moved to the front of the queue, pushing downstream TCP ACKs further back and generating increased delay and RTT. This has the dramatic effect of decreasing TCP downstream throughput by a further 35%.

While increased bandwidth would both increase throughput and lower the queuing delay experienced by delay sensitive applications, whenever a bottle-neck is created by a lower-bandwidth Internet access link, contention for this resource by concurrent network applications will result in queues being formed. For applications that are particularly sensitive to queuing delays, their performance can be markedly improved through implementation of priority queuing.

With regard to the performance of TCP flows, it can be useful to consider the types of network applications that generate these flows. While applications like www and email typically are interactive, most users will not notice a drop in throughput of the order of 10-20%. This will result in situations where web pages take an extra few seconds to load. Other applications such as BitTorrent typically run in the background and an extra delay in throughput will probably not be noticed by users. It should also be noted that the primary cause of decreased downstream TCP performance is the queuing delay experienced by the ACKs.

One possible approach that could be used to improve TCP downstream performance in any scenario where significant upstream bandwidth is being utilised is to also prioritise transmission of upstream ACKs. This would decrease the queuing delay of the ACKs and the TCP RTT, allowing TCP

downstream throughput to increase at the expense of slightly decreasing upstream throughput.

## V. CONCLUSION

Delay sensitive applications such as network gaming and VoIP often experience poor performance in an environment where Internet access is shared by a number of concurrent applications. In our trials over a DOCSIS Cable Modem link we found that the packet loss rate for these applications approached 30% and the mean queuing delay for those packets that were not dropped was 1.3s. These figures would render the application unusable.

One approach to improve the performance of delay sensitive flows is to implement priority queuing at the bottleneck, giving traffic from delay sensitive applications priority access to the bottleneck link. Our trials show that this does improve the performance, dropped packets were almost eliminated and the mean queuing delay dropped to less than 40ms.

Of concern is how this would impact other applications which are contending for access to the Internet link. While our trials show that downstream TCP throughput is heavily compromised, this is primarily caused by the bursty upstream TCP flow in conjunction with the asymmetric bandwidth allocation of the Internet link rather than the prioritised game packets. Our work indicates that prioritisation of delay sensitive applications such as network games and VoIP would have a significant impact on the performance of these applications with minimal impact on other TCP-based applications.

We believe that scope exists for further work in quantifying the effect of prioritising upstream TCP ACKs in an attempt to both minimise the downstream TCP RTT and subsequently improve the performance of downstream TCP flows.

## REFERENCES

- [1] G. J. Armitage, "An Experimental Estimation of Latency Sensitivity in Multiplayer Quake3," in *Proceedings 11th IEEE International Conference on Networks (ICON)*, September 2003.
- [2] S. Zander, G. Armitage, "Empirically Measuring the QoS Sensitivity of Interactive Online Game Players," in *Proceedings of Australian Telecommunications and Network Applications Conference (ATNAC)*, December 2004.
- [3] M. Dick, O. Wellnitz, L. Wolf, "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games," in *Proceedings of ACM Network and System Support for Games (NetGames) Workshop*, October 2005.
- [4] 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 *NetGames 2004: Proceedings of the ACM SIGCOMM 2004 Workshop NetGames 2004*, 2004.
- [5] L. Stewart, G. Armitage, P. Branch, S. Zander, "An Architecture for Automated Network Control of QoS over Consumer Broadband Links," in *Poster Presentation at IEEE TENCON*, November 2005.
- [6] "BART - Broadband Access Research Testbed, Centre for Advanced Internet Architectures - Swinburne University," July 2006, <http://caia.swin.edu.au/bart>.
- [7] "IPerf - The TCP/UDP Bandwidth Measurement Tool," July 2006, <http://dast.nlanr.net/Projects/Iperf>.
- [8] J. But, U. Keller, D. Kennedy, and G. Armitage, "Passive TCP Stream Estimation of RTT and Jitter Parameters," in *LCN '05: Proceedings of the IEEE 30th Conference on Local Computer Networks*, 2005, pp. 433-440.
- [9] T. Lang, G. Armitage, P. Branch, and H.-Y. Choo, "A Synthetic Traffic Model for Half-Life," in *ATNAC 2003: Australian Telecommunications Networks and Applications Conference 2003*, 2003.