Multimedia Systems Journal (MMSJ) – Network and Systems Support for Games

On the Performance of OnLive Thin Client Games

Mark Claypool · David Finkel · Alexander Grant · Michael Solano

Received: October 4, 2013

Abstract Computer games stand to benefit from "cloud" technology by doing heavy-weight, graphics-intensive computations at the server, sending only the visual game frames down to a thin client, with the client sending only the player actions upstream to the server. However, computer games tend to be graphically intense with fast-paced user actions necessitating bitrates and update frequencies that may stress end-host networks. Understanding the traffic characteristics of thin client games is important for building traffic models and traffic classifiers, as well as adequately planning network infrastructures to meet future demand. While there have been numerous studies detailing online game traffic and streaming video traffic, this paper provides the first detailed study of the network characteristic of OnLive, a commercially available thin client game system. Carefully designed experiments measure OnLive game traffic for several game genres, analyzing the bitrates, packet sizes and inter-packet times for both upstream and downstream game traffic, and analyzing frame rates for the games. Results indicate OnLive rapidly sends large packets downstream, similar but still significantly different than live video. Upstream, OnLive less frequently sends much smaller packets, significantly different than upstream traditional game client traffic. OnLive supports only the top frame rates with high capacity end-host connections, but provides good frame rates with moderate end-host connections. The results should be a useful beginning to building effective traffic models and traffic classifiers, and for preparing end-host networks to support this upcoming generation of computer games.

Computer Science and Interactive Media & Game Development
Worcester Polytechnic Institute
Worcester, MA 01609, USA
email: {claypool,dfinkel}@cs.wpi.edu

1 Introduction

The computer game industry has seen tremendous growth in recent years, and is forecasted to be over a $100 billion industry by 2015 [3], on par with the U.S. movie industry revenue worldwide (DVD and film).[1] Online games have also seen considerable growth, spurred on by the growth in residential broadband Internet connections with high capacities and low latencies that have encouraged game developers to incorporate networked features into their products.

Thin clients, where the local computer is primarily an input and output device and the remote computer does the majority of the processing, have seen a resurgence in use because today's network capacities and latencies can support bandwidth-intensive, client-server interactions. Thin clients are expected to grow from over 12 million units shipped in 2011 to over 25 million in 2013, and by 2014, 15% of traditional professional desktop PCs are projected to be replaced by virtual desktops accessed from thin clients [3]. Using thin clients for games has already become commercial, with companies such as OnLive[2] and GameNow[3] having commercial success and continuing to expand.

In order to classify network traffic and adequately plan network infrastructures, it is important for engineers to have knowledge of the network load caused by emerging end-host applications and devices. For traditional network games, traffic from a number of popular games has been characterized to provide suitable traffic models for testing existing or planned network designs.

[1] "How Much Does Hollywood Earn?", Information is Beautiful, `http://www.informationisbeautiful.net/2012/how-much-does-hollywood-earn/`
[2] `http://www.onlive.com/`
[3] `http://www.ugamenow.com/`

There have been numerous studies of traffic models for popular PC games [4,7,15,22,23] and even game consoles [32]. There have also been studies of traditional video, both pre-recorded [17,33] and live [24,30]. However, to the best of our knowledge, measurements of thin clients across games with comparisons to traditional games and streaming video have not been done.

Using network sniffing to capture OnLive traffic, this paper investigates the network characteristics (the size and frequency of data sent and the overall bitrate), which we call *turbulence.*[4] Additional analysis is provided on the frame rate achieved by OnLive over a variety of network conditions. This study seeks to answer the following questions (with a brief answer as revealed by this study provided in parentheses):

1. *What is the network turbulence for OnLive games?* Characterizing the data rates, packet sizes and inter-packet times for OnLive games is a critical first step for building accurate game traffic models and traffic classifiers. (Answer: OnLive games have high downstream bitrates, about 5 Mb/s with 1000 byte packets, with much more moderate upstream bitrates, about 100 Kb/s with 150 byte packets.)
2. *Does the network turbulence for different game genres (such as first-person vs. omnipresent) differ from each other?* If the answer is no, then research efforts can study traffic on one game genre only, saving hardware costs and time. However, previous work has shown that different game genres have different amounts of visual motion and scene complexity [10]. (Answer: The characteristics of game traffic is similar for all genres, but total bitrates for downstream and upstream traffic can vary by as much as 50% across genres.)
3. *Does the network turbulence for OnLive games differ from traditional games?* If the answer is no, then mature, previously developed models for online games can be used to analyze the impact of thin client games. (Answer: OnLive downstream traffic is more similar to downstream live video than traditional games, while upstream traffic is only somewhat similar to upstream game traffic.)
4. *Does the network turbulence for OnLive change with different network conditions (such as packet loss or capacity limits)?* If the answer is no, then this suggests OnLive is not adaptive to network conditions, making it easier to model but with more potential to disrupt networks, while if yes, then further studies are needed to ascertain how, exactly, OnLive adapts to the network. (Answer: OnLive does adapt bitrates to capacity limits, but does not adapt bitrates to loss or latency. OnLive is clearly not TCP-friendly in many cases, having bitrates much higher than conformant TCP flows under conditions of modest loss and latency.)
5. *Do OnLive frame rates change in response to network conditions?* Frame rates have been shown to have a marked effect on player performance for games [9,11], so understanding frame rates during network perturbations can help predict player satisfaction with thin client games. (Answer: OnLive frame rates adapt to both capacity limits and loss, but not latency. Frame rates become unacceptable at the minimum recommended capacities and only provide maximum player performance when capacities are above the recommended limits.)

[4] The term "footprint" typically refers to the memory size of a software process. In a network, the size and distribution of packets over time is important, hence our word "turbulence."

While the core network turbulence results appeared in an earlier paper [14], this paper extends that work significantly with: 1) an in-depth analysis of OnLive frame rates in the presence of network perturbations, 2) a prediction of player performance based on latency and frame rate, 3) analysis of OnLive bitrates over a broad range of packet loss and latencies, and 4) a comparison of OnLive bitrates to TCP-friendly bitrates over the same ranges.

The rest of this paper is organized as follows: Section 2 provides related work on measuring performance of thin clients; Section 3 describes our measurement setup and methodology; Section 4 analyzes our results in relation to the questions posed above; and Section 5 summarizes our conclusions and presents possible future work.

2 Related Work

This work overlaps and builds upon research from two main areas: work measuring the performance of thin clients (Section 2.1) and work specific to thin client games (Section 2.2).

2.1 Performance of Thin Clients

Lai and Nieh [21] use a novel, slow-motion benchmarking technique to evaluate the performance of several thin client platforms. Their measurements include analysis of network traffic, where bitrates can vary across platforms by ten-fold for the same tasks. Bitrates for displaying video across thin clients varies from about 1 Mb/s to over 40 Mb/s for some platforms. They show general performance is often adequate for high capacity links, even when running across the entire U.S. However, thin clients with more efficient bitrates can still have latency as the bottleneck to performance. They provide a summary of design choices that can aid thin client computing development for traditional applications.

Packard and Gettys [26] passively monitor network traffic between X clients and X servers under network capacity controlled conditions. A variety of test applications assess a shallow understanding and quantification of performance issues. Among other findings, latency was found to dominate capacity in limiting performance for some applications and image transfers were found to dominate the overall network capacity used.

Billinghurst et al. [2] describe communication asymmetries for a wearable (and thin) display and a desktop computer. An accompanying user study attempts to determine the effects on collaboration, showing that the extent to which communication over a thin client is difficult largely depends upon the task being undertaken.

Kim et al. [19] describe pTHINC, a thin-client solution for PDAs that runs applications such as Web browsers on more powerful, but remote, servers. With pTHINC, server-side scaling of the display provides improved performance for a variety of heterogeneous client displays. The results show pTHINC provides superior Web browsing performance for a PDA.

The above studies provide detailed insights into thin client performance, but predominantly pertain to traditional applications with no study specifically focusing on thin clients and games.

2.2 Games on Thin Clients

De Winter et al. [29] propose a thin client system designed specifically for streaming and interactive games. Their system streams screen images after rendering by the graphics card, thus reducing bitrates and increasing visual quality for streaming video games.

Chen et al. [8] study the performance of OnLive by comparing it to another cloud gaming platform called StreamMyGame. Unlike OnLive which has a service provided by OnLive, StreamMyGame is a software solution that is managed and operated by the researchers themselves, allowing for greater control of system parameters but with perhaps less generality.

Chang et al. [6] also propose a methodology for studying thin client game systems based on the game as displayed on the server compared to the game as displayed on the client, quantifying frame rate and frame distortion. Similar to work on traditional client-server games [13], the authors find that frame rate is more critical to gameplay than frame distortion. The authors apply their methodology to Ms. Pac-Man for three thin client systems, LogMeIn, TeamViewer, and UltraVNC, and find frame-based metrics correlate well with game performance, and that the thin client implementations have different amounts of robustness against network impairments.

Our work complements these approaches by providing insight into the characteristics of streaming video games, enabling better scaling decisions for the streamed images after rendering, as well as evaluating a commercially successful thin client technology with several representative games.

3 Methodology

To measure the turbulence and performance of OnLive, the following methodology was used:

– Select games to play on OnLive (Section 3.1).
– Setup testbed to measure network turbulence and frame rates (Section 3.2).
– Run experiments, gathering network statistics and performance data (Section 3.3).
– Analyze data (Section 4).

3.1 Game Selection

In order to ascertain if turbulence for OnLive varies with the type of game, representative games from three different genres were selected. Following the game classification described in [12], games were chosen from each dominant genre: *first person avatar*, *third person avatar*, and *omnipresent*. The selection of games was limited to those available via OnLive (about 300 titles at the time of the experiments). Games were chosen based on perceived popularity and with similar release dates in an attempt to provide for relatively comparable visual graphics quality. Table 1 summarizes the games selected, with indicated classification and year of initial release. The first person avatar, third person avatar and omnipresent games selected were *Unreal Tournament III* (UT), *Batman: Arkham Asylum* (Batman) and *Grand Ages: Rome* (Rome), respectively. Screen shots of each game are depicted in Figure 1, Figure 2 and Figure 3. As a measure of the system impact, the minimum system requirements as specified by the game manufacturers is provided in Table 2. The resources required for each game are similar, with Batman having a higher memory requirement than the other two games and Rome requiring a dual core Intel processor.

Table 1 Games used in experiments

Game	Classification	Release Year
Unreal Tournament III	First person avatar	2007
Batman: Arkham Asylum	Third person avatar	2009
Grand Ages: Rome	Omnipresent	2009

Table 2 Recommended system requirements

Part	UT	Batman	Rome
CPU	PentD 2.66	PentD 3.0 Ath64 X2 Dual 3800+	Core 2 Duo 2.2 Ath64 X2 3600+
Grph	GeForce 8800 Radeon x800	GeForce 7900 Radeon x800	GeForce 7800 Radeon x850+
Mem	1 GB	2 GB	1 GB
Sftwre	DirectX v9	DirectX v9	DirectX v9

Fig. 1 Unreal Tournament III screenshot

Some idiot is running round the asylum dressed like a bat. I know. Crazy!
Fig. 2 Batman: Arkham Asylum screenshot

Fig. 3 Grand Ages: Rome screenshot

3.2 Measurement Testbed

Figure 4 depicts the measurement testbed setup for our experiments.

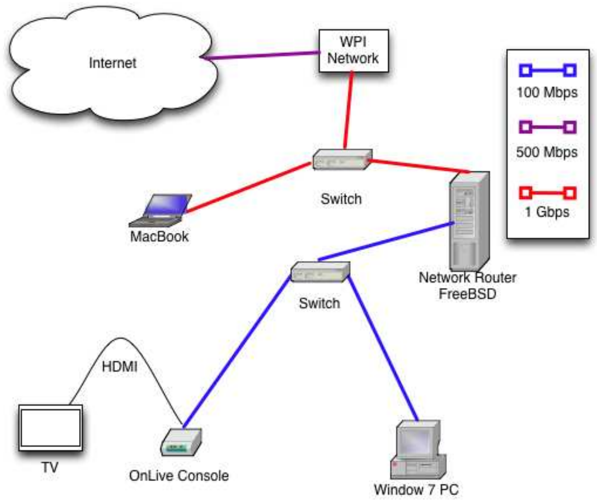

Fig. 4 Measurement testbed

For turbulence experiments, an OnLive MicroConsole runs the games and is connected to an HDTV over HDMI running at 1080p. Another PC is configured as a network router running FreeBSD and Dummynet [5] to allow emulation of a wide variety of network conditions at the IP level, including fine tuning of network capacity, loss, and latency. The PC runs tcpdump to capture traffic for analysis of turbulence after the experiments completed. For frame rate experiments, a desktop PC runs Win7-64 bit on a 3.01 GHz Intel Core 2 Duo, with 4GB RAM and a GeForce 9800 GT video card. The

PC runs Fraps[5] to record frame rates for analysis after the experiments completed. The WPI campus egress to the Internet is 500 Mb/s, while switches on the campus have 100 Mb/s and 1 Gb/s capacities.

3.3 Experiments

Pilot studies captured game traces for varying lengths in order to determine how long individual game sessions should run. Based on these runs, it was determined 2.5 minutes provided steady state behavior.

3.3.1 Games

For each game, a scenario was selected to represent core game play.

– **UT:** A free-for-all match on the Rising Sun map was started with: 10 Bots, no mutators, forced respawns, and no limit on score or time. The player then competed for weapons, armor, and health using them to defeat the Bots.
– **Batman:** A challenge mode was used where the player started in a small square room with 3 enemies, with more appearing continuously. The player used attack combos to incapacitate as many enemies as possible.
– **Rome:** A empty map was used where the player constructed buildings using the technology tree, with no enemies encountered. The player built (in order): 3 insulas, 1 pig farm, 1 wheat farm, 1 aqueduct, 1 large water fountain, 1 logging shed, 2 more insulas, 1 butcher shop, 1 farmers market, 1 grape farm, and 2 logging sheds.

The packet captures proceeded from the start until the end of each game session, with analysis trimming each session to 2.5 minutes of core gameplay (i.e., loading, authentication, menu phases, etc. are not included – just gameplay).

3.4 Parameters

All together, the three games were tested over conditions with varied capacity, loss and latency:

– *Game genre*: UT, Batman, and Rome.
– *Capacity* (downstream:upstream): 1 to 10 Mb/s and no restriction
– *Latency* (round-trip): 0 milliseconds to 1000 milliseconds
– *Loss* (downstream): 0% to 18% loss
– *Iterations*: 3 runs for each experiment condition, except where noted

Performance measures captured include network and application statistics:

– Network: packet sizes (bytes), inter packet times (milliseconds), bitrates (Kb/s and Mb/s)
– Application: frame rates (f/s)

4 Analysis

Analysis proceeds by first exploring OnLive turbulence for different game genres (Section 4.1) and with network perturbations (Section 4.2). Frame rate analysis provides OnLive performance in addition to the network implications (Section 4.3). Turbulence comparisons with other applications is next (Section 4.4), followed by a comparison with traditional game traffic (Section 4.5). Analysis concludes with a summary (Section 4.6).

4.1 Turbulence

Our traces show OnLive primarily uses UDP for both downstream and upstream game traffic. To assess network turbulence, bitrate, packet size and inter-packet times are analyzed.

Figure 5 depicts a comparison of the downstream bitrate (computed every second) for OnLive for the three games under test. In terms of bitrates, all three trials look visually similar. Thus, in this graph, and all subsequent bitrate graphs, only the second trial is shown. The x-axis is the measurement time in seconds, and the y-axis is the bitrate in Kb/s. Each game is depicted by a separate trendline. The top two lines are UT and Batman, while the bottom line is Rome. The similarity in bitrates for UT and Batman (about 6.3 Mb/s) could be because of the relatively similar camera movements they provide to the player, while Rome has around half the bitrate (about 3.8 Mb/s), possibly because of the different camera angles afforded by an omnipresent game. Moreover, our gameplay in Rome had only construction and not combat which could account for the difference. There is visually more variance in the bitrate in Rome than in UT or Batman.

Figure 6 depicts a comparison of the cumulative distribution functions (CDFs) of the downstream packet sizes for the three games. The x-axis is packet size and the y-axis is the cumulative distribution. Note, for readability, only 1 out of every 1000 points are plotted. Visually, the trendlines are all similar, probably due to

[5] `http://www.fraps.com`

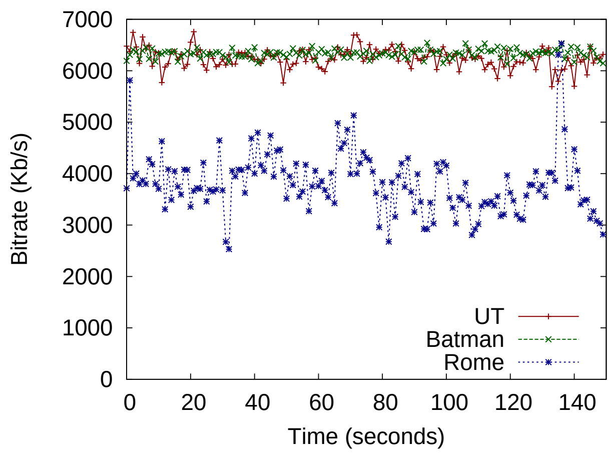

Fig. 5 Downstream bitrate

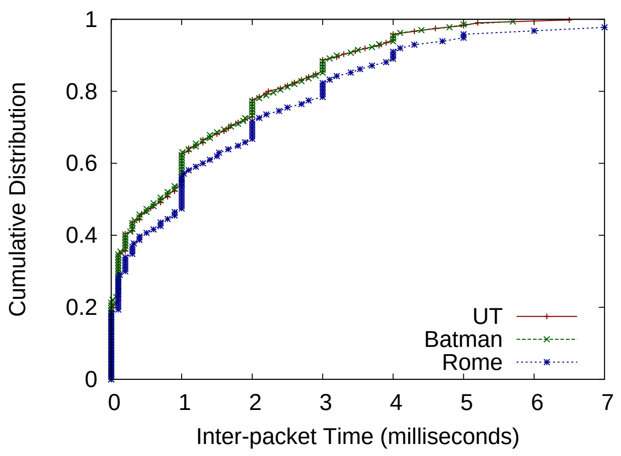

Fig. 7 Downstream inter-packet times

the nature of the encoding of the display images sent from the server to the client. The mode, about 30% of all packets, is 1414 bytes, which is smaller than the MTU of 1500 bytes that could be used. About 10% of the packets are 622 bytes. The other 60% of the packets are distributed fairly uniformly between about 100 and 1400 bytes.

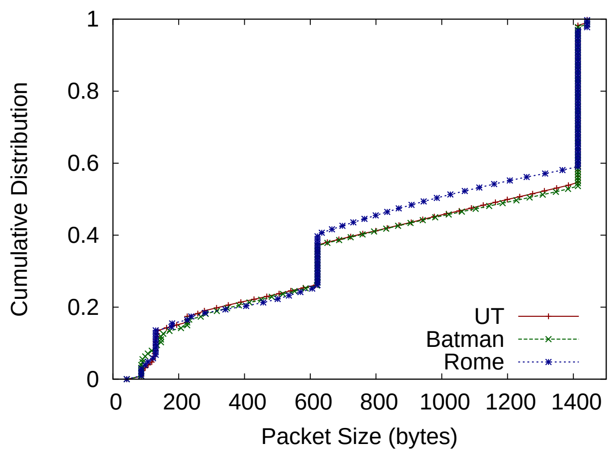

Fig. 6 Downstream packet size

Figure 7 depicts a comparison of the inter-packet time CDFs for the three games, also plotting 1 out of every 1000 points for readability. The x-axis here is the time between packets, in milliseconds. In general, inter-packet times are quite small, with a median less than half a millisecond. The maximum inter-packet times are only tens of milliseconds. Visually, the trendlines are all similar, particularly so for UT and Batman.

Figure 8 depicts a comparison of the upstream bitrate for OnLive for the three games, showing one trial (as for the downstream traffic, the other trials were similar). The x-axis is the measurement time in seconds, and the y-axis is the bitrate in Kb/s with trendlines as in earlier graphs. Visually, all three games have considerable variation in their upstream bitrates, ranging from about 50 to 150 Kb/s. UT has a slightly higher upstream bitrate than Batman or Rome, possibly due to the fast-action nature of first person shooter games. Note that the y-axis scale in Figure 5 is 40 times greater than the y-axis scale in Figure 8 since the downstream to upstream bitrate ratios are about 40 to 1.

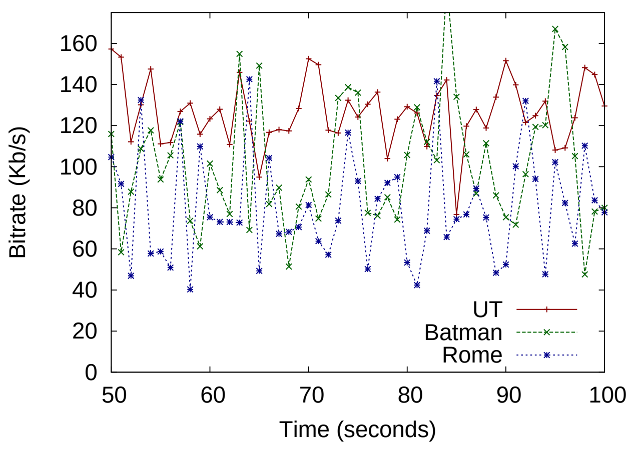

Fig. 8 Upstream bitrate

Figure 9 depicts a CDF of the upstream packet sizes for the three games. Most upstream packets are small, about 100 bytes of application payload after subtracting the IP headers. Nearly all the packets are under 250 bytes. Note, the x-axis scale is only to 350 bytes compared to a typical 1500 byte MTU.

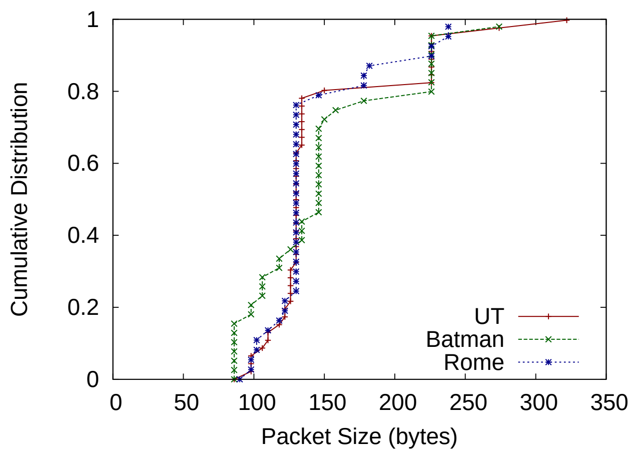

Fig. 9 Upstream packetsize

Figure 10 depicts a comparison of the inter-packet time CDFs for the three games, plotting 1 out of every 200 points for readability. The inter-packet times are still small, most under 20 milliseconds, but are considerably larger than the downstream inter-packet times (the x-axis scale for Figure 10 is about 10x larger than for Figure 7). The maximum inter-packet times upstream are slightly over 50 milliseconds. Visually, the trendlines are all similar, despite the fact that the player interactions with each game genre may be considerably different.

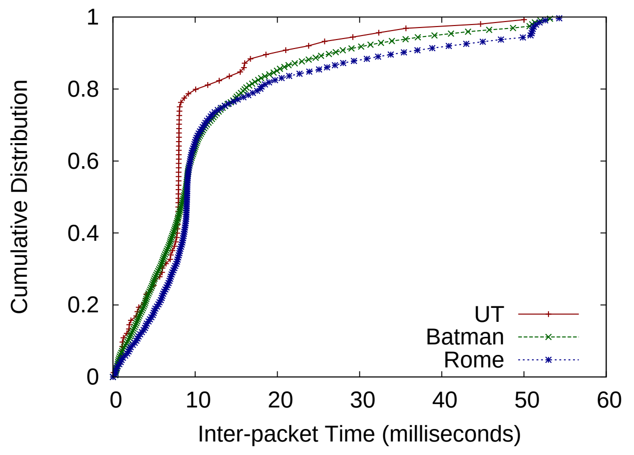

Fig. 10 Upstream inter-packet times

Table 3 provides summary statistics (mean values) for OnLive games, showing both upstream and downstream turbulence. Overall, OnLive traffic is very asymmetric, with downstream having about 50x higher bitrates, 10x more packets per second and 6x larger packets.

4.2 Network Perturbations

Given the relative similarity in turbulence for the three games studied, despite their different genres, for subsequent experiments UT is used as the representative OnLive game. Also, graphs of packet sizes and inter-packet times are not shown – instead the concentration is on bitrate to depict network turbulence. Also, as a final focus, since OnLive downstream traffic dominates that of upstream, primarily downstream traffic is analyzed unless noted otherwise.

As mentioned in Section 3, Dummynet was used to restrict capacity on the link to the game console. Figure 11 depicts UT bitrates for 10 Mb/s and 5 Mb/s, with the unrestricted trendline (top) from Figure 5 for comparison. Restricting the bitrate to 10 Mb/s (roughly, that of a residential broadband link in the U.S.) on the downlink yields a bitrate of about 4200 Kb/s, with a further restriction to 5 Mb/s dropping the bitrate to just over 2000 Kb/s. There is a slight decrease in variance with increased capacity restrictions. In both restricted cases, the downstream bitrate is about half the capacity restriction.

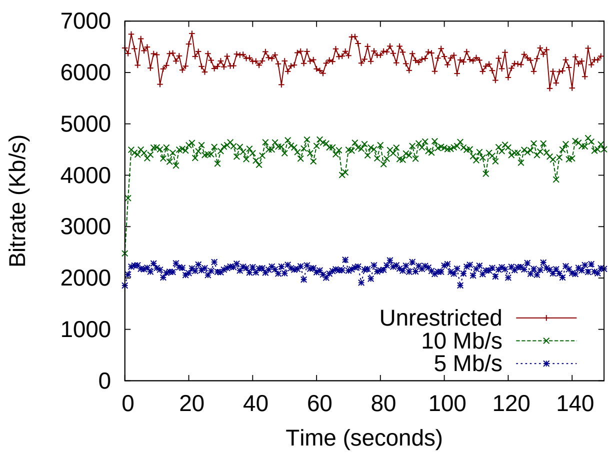

Fig. 11 Downstream bitrate with capacity restrictions

Dummynet was used to induce 1% and 1.5% packet loss on the link to the game console and then also to

Table 3 Turbulence for OnLive games (upstream and downstream, mean values)

Game	Bitrate (Kb/s) up	Bitrate (Kb/s) down	Packet Size (bytes) up	Packet Size (bytes) down	Inter-packet Time (msec) up	Inter-packet Time (msec) down
UT	125	6247	146	947	9.5	1.2
Batman	100	6333	152	953	11.7	1.2
Rome	86	3817	143	914	13.4	1.6

add 40 and 70 msec of round-trip latency. Figure 12 depicts the corresponding UT bitrate comparisons. Visually, there is little effect on bitrate from added latency and loss, except perhaps slightly more bitrate variance with increased loss.

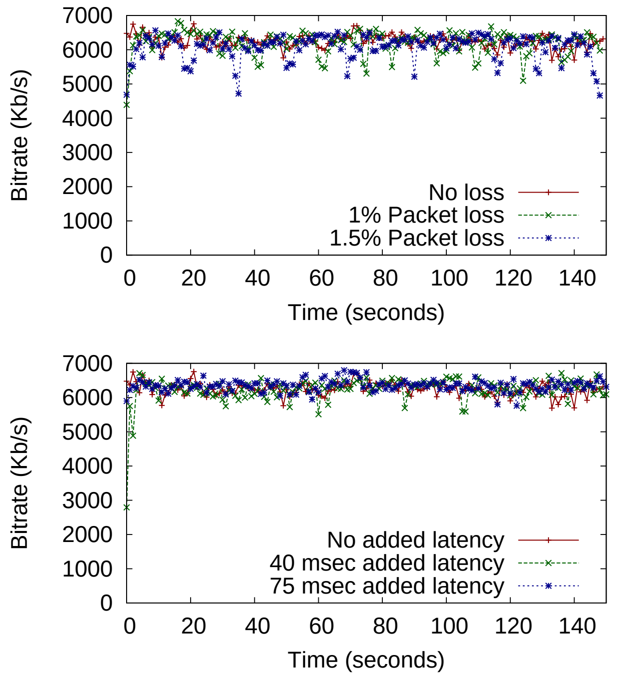

Fig. 12 Downstream bitrate with induced packet loss (top) or added latency (bottom)

Packet loss typically indicates congestion for TCP flows, causing them to reduce their sending rates. OnLive, running over UDP, does not benefit from TCP's automatic response to congestion and has the potential to consume more capacity than competing TCP flows, or worse, to contribute to Internet congestion collapse [16]. Ideally, high bitrate flows, such as OnLive (with 6 Mb/s or more), are *TCP-friendly* with data rates that do not exceed the maximum rates of conformant TCP flows under the same network circumstances (i.e., same round-trip times, loss rates and packet sizes). In general, the bitrate of a TCP flow decreases with round-trip time and loss rate. The TCP-friendly bitrate, T, in bytes per second, for a flow is [16]:

$$T \leq \frac{1.5\sqrt{2/3} \times s}{R \times \sqrt{p}} \tag{1}$$

with packet size s in bytes, round-trip time R in milliseconds and packet loss fraction p.

Figure 13 depicts the bitrate for OnLive versus loss rate (top) and added round-trip latency (bottom), with the TCP-friendly rate trendline shown for each graph. The loss rate is 0.01% in the latency graph and the latency is 50 milliseconds (the round-trip time to the OnLive server) in the loss graph. From the graphs, OnLive is clearly *not* TCP-friendly for loss rates of 1% or higher or for latencies of 40 milliseconds or higher. Worse, OnLive's bitrates are completely unresponsive to packet loss, with OnLive flows with loss rates as high as 20% having the same offered bitrate as loss rates of 0% – the bitrate shown at 20% loss in Figure 13-bottom is 20% less than the bitrate at 0% loss simply because 20% of the packets are dropped.

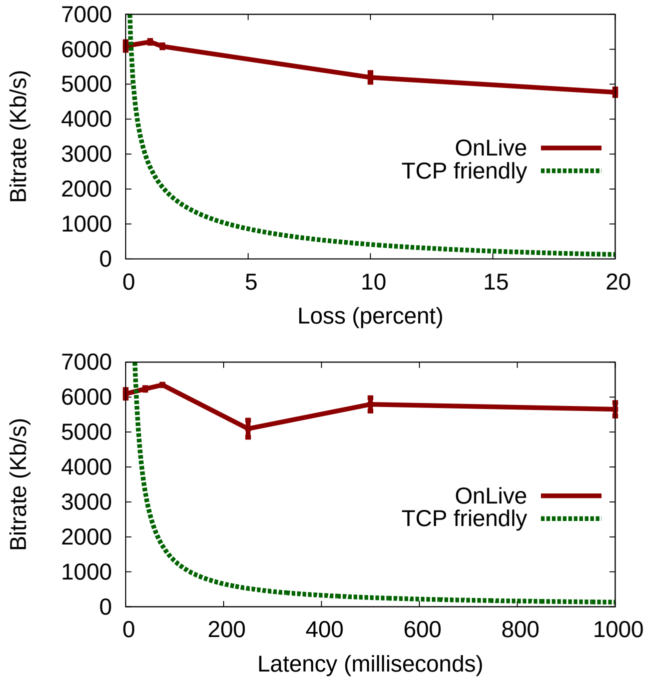

Fig. 13 Bitrate versus loss rate (top) and latency (bottom)

Since OnLive and other thin client game systems provide a new paradigm for network gaming, it is worth considering the TCP-friendliness of games played on traditional "fat" clients. Our earlier work [1] measures the network traffic of traditional UT over a range of loss and latency conditions. Bitrates are found to be unresponsive to both latency and loss, suggesting, as for OnLive UT, that traditional UT is not TCP-friendly at high levels of loss and latency. Given the relatively low fixed bitrate of 67 Kb/s for a single, traditional UT client, traditional UT only becomes non-TCP-friendly for loss rates higher than about 25% or latencies higher than about 2000 milliseconds.

4.3 Frame Rate

Analysis of game frame rates provides insights into the effects of network perturbations on player performance. To broadly assess the impact of capacity on OnLive frame rates, a 3 minute, 20 second game of UT was played. Capacity was initially restricted to 10 Mb/s and was decreased by 1 Mb/s every 10 seconds. Once at 1 Mb/s, the capacity was increased by 1 Mb/s every 10 seconds until it reached 10 Mb/s. As described in Section 3.2, Fraps was used to record frame rates.

Figure 14 depicts the results. The y-axis is the frame rate, the bottom x-axis is time in seconds since the experiment started, and the top x-axis indicates the corresponding capacity in Mb/s at that time. In the figure, the first observable change in the frame rate is around 40 seconds (6 Mb/s), with a rapid decrease in frame rate thereafter. The flat "steps" that are about 5-10 seconds long, as observed in the trend line, suggest 10 seconds is enough time for the OnLive system to respond to decreased network capacities. There is an rise in frame rate at about 80 seconds (2 Mb/s), even though the capacity has not increased but rather is still decreasing. Visually, the frame rate pattern to the left of 100 seconds, when capacities are decreasing, is different than the frame rate pattern to the right of 100 seconds, when capacities are increasing. This may arise from the ability of systems to more easily observe decreased capacity (observable by data loss), than to observe increased capacity which does not have a corresponding signal.

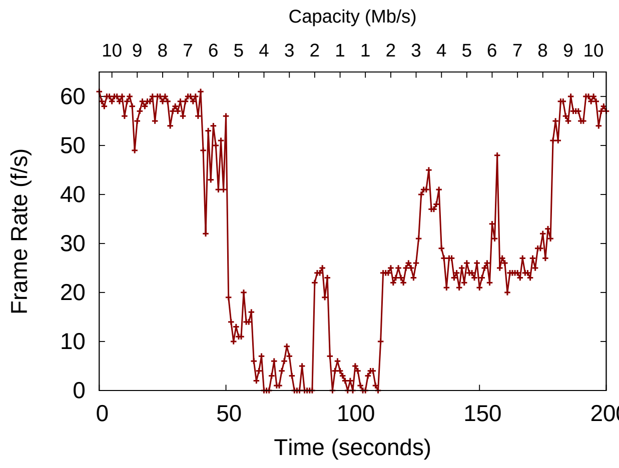

Fig. 14 Frame rate over time, capacity decreasing 1 Mb/s every 10 seconds then increasing

Instead of changing capacity mid-game, 1 minute games were run with fixed capacity constraints ranging between 1 Mb/s and 10 Mb/s. Figure 15 shows the results, with each point representing the average frame rate at a specific capacity, shown with 95% confidence intervals. Generally, capacities between 2 Mb/s and 6 Mb/s provide 25-30 f/s. There is a linear increase in frame rate from 1 Mb/s to 2 Mb/s, and a roughly linear increase in frame rate from 6 Mb/s to 9 Mb/s. However, the larger confidence intervals in this range suggest OnLive is adjusting the target frame rate frequently. At 9 Mb/s or greater, the average frame rate is near the 60 f/s maximum.

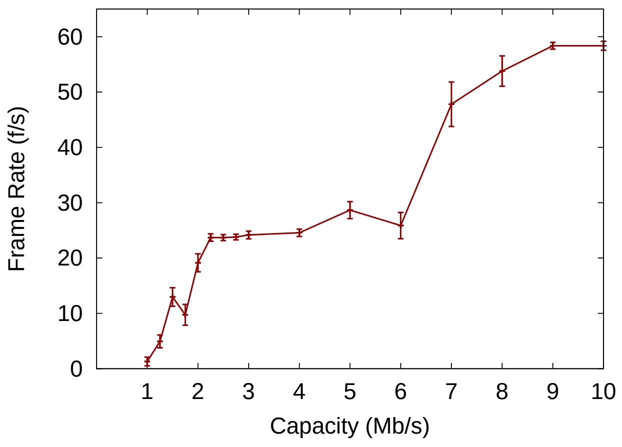

Fig. 15 Average frame rates with capacity constraints

The mean frame rate may be indicative of performance, but the distribution of frame rates provides a more nuanced view. OnLive lists 2 Mb/s as the minimum downstream Internet capacity and 5 Mb/s as the recommended downstream Internet capacity.[6] Figure 16 depicts the distributions of bitrates for different capacity constraints. The distributions in Figure 16-top are above the recommended capacity, distributions in Figure 16-middle are between the minimum and recommended capacities, and distributions in Figure 16-bottom are below the minimum capacity. For all graphs, the x-axes are the frame rates achieved and the vertical axes are the cumulative distributions.

When capacities are 7 Mb/s or higher, most of the time OnLive achieves frame rates near 60 f/s. Capacities of 9 Mb/s or 10 Mb/s always have at least 50 f/s, while capacities 7 Mb/s and 8 Mb/s have frame rates between 20 f/s and 50 f/s about 20% of the time. For capacities in the range 3 Mb/s to 6 Mb/s, OnLive consistently achieves frame rates between 20 f/s and 30 f/s, with a median around 25 f/s. At 2 Mb/s, the minimum according to OnLive, the median frame rate drops to about 15 f/s and frame rates are never above

[6] `https://support.onlive.com/entries/22264983-computer-and-internet-requirements-for-pc-mac`

25 f/s. Below 2 Mb/s, capacities of 1.75 Mb/s and 1.50 Mb/s achieve frame rates mostly between 10 f/s and 15 f/s, with only about 10% of the frame rates above this range and about 10% of the frame rates below 5 f/s. For capacities of 1.00 Mb/s and 1.25 Mb/s, median frame rates are below 3 f/s, and only about 10% of the frame rates are above 7 f/s.

Frame rates affect the smoothness of the visual display and also the responsiveness of the game for players. Thus, understanding the effects that capacity constraints and other network perturbations have on game players is critical to ascertaining the efficacy of thin client gaming. Although a user study measuring the effects of OnLive frame rate on player performance is beyond the scope of this paper, predictions can be made based on previous work. Claypool and Claypool [9] conducted several detailed user studies that measured the effects of frame rate on performance for UT 2003 played on traditional "fat" clients. Player performance peaked at about 60 f/s and degraded with a decrease in frame rate until it bottomed out at 3 f/s. Based on this data, assuming the "best" player performance occurs at 60+ f/s (normalized to 1), then a model of performance p based on frame rate x in frames/second is derived in Equation 2. Figure 17 depicts Equation 2, with the x-axis the frame rate and the y-axis the normalized performance (1 is "best" and 0 is "worst").

$$p(x) = 0.30 \times ln(x) - 0.25 \tag{2}$$

While capacity constraints have also been observed to degrade the visual quality of OnLive, our previous work [13,11] has shown that scaling the visual quality of games, while noticeable and important to players, has little effect on player performance. Since OnLive has been observed to adapt bitrates to the network capacity (see Figure 11), the loss rates remain similar (negligible) given a fixed capacity.

Whereas Equation 2 provides a prediction of performance for a given frame rate, predicting performance for a given capacity constraint may help inform network choices, such as the amount of capacity to purchase from an ISP provider. Using the median frame rate for each capacity constraint as the input for Equation 2 and fitting a curve to the result gives performance p (between 0 and 1) with bitrate constraint c in Mb/s in Equation 3.

$$p(c) = 0.37 \times ln(c) + 0.2 \tag{3}$$

Figure 18 depicts Equation 3, with the x-axis the capacity constraint and the y-axis the normalized predicted performance (1 is "best" and 0 is "worst"). Players that tolerate no more than a 10% degradation in

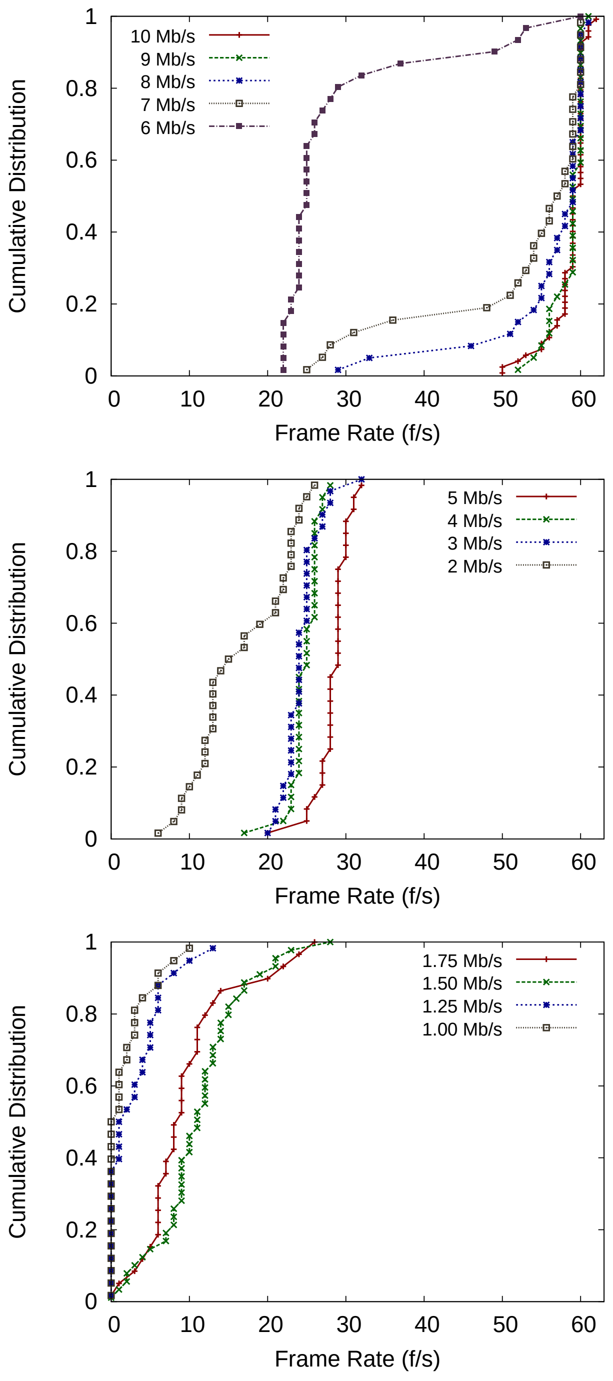

Fig. 16 Downstream bitrates with capacity constraints – higher than recommended capacity (top), minimum capacity to recommended capacity (middle), lower than minimum capacity (bottom)

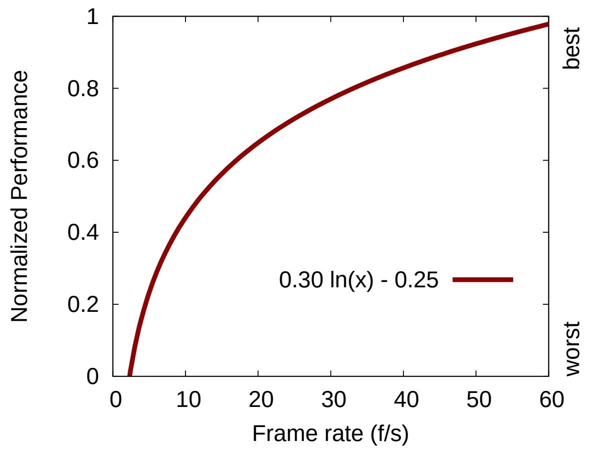

Fig. 17 Model of player performance versus frame rate based on data from [9]

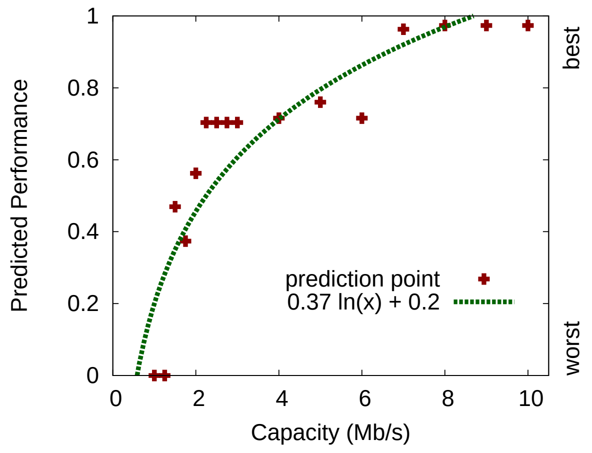

Fig. 18 Predicted player performance versus capacity

performance need a capacity of 7 Mb/s or higher. Playing OnLive at the recommended 5 Mb/s results in player performance degradation of about 20%, while playing OnLive at the minimum of 2 Mb/s results in about 50% degradation.

To broadly assess the impact of packet loss on OnLive frame rates, a 3 minute, 20 second game of UT was played. The packet loss rate was initially set to 0% and was increased by 2% every 10 seconds. Once at 18%, the loss rate was then decreased every 10 seconds until it reached 0%. The capacity constraint remained at 10 Mb/s throughout the test.

Figure 19 depicts the results. The y-axis is the frame rate, the bottom x-axis is time in seconds since the experiment started, and the top x-axis indicates the corresponding loss rate in percent at that time. Up until about 4% loss, OnLive tries to maintain a frame rate near 60 f/s, but there is a noticeable drop at about 6% loss where the frame rate drops to about 30 f/s. There is a slow degradation in frame rate until about 18% having, where the frame rate is below 20 f/s. The decreasing loss rate at time 120 s results in a slightly increasing frame rate. Note, however, even when the loss rate is above 4% at the end of the run, the frame rate does not recover to the 50+ f/s seen in the beginning. This behavior suggests OnLive is conservative in increasing target frame rates until it can be certain the network conditions can support it.

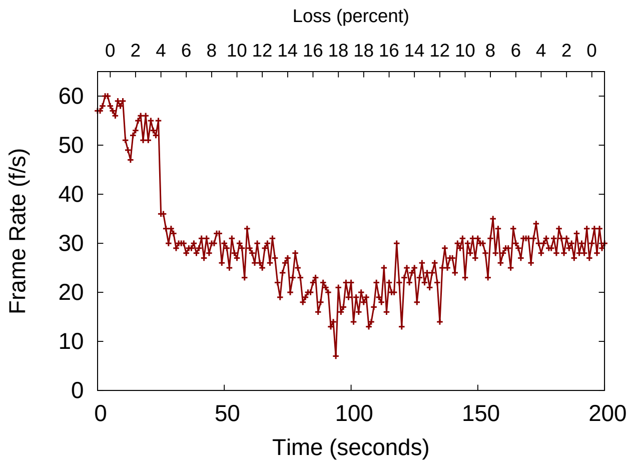

Fig. 19 Frame rate over time, loss increasing by 2% every 10 seconds then decreasing

Instead of changing loss rate mid-game, 1 minute games were run with fixed loss rates between 0% and 18%. Figure 20 shows the results, with each point the average frame rate at that loss rate, with a 95% confidence interval. Only at 0% loss does OnLive achieve nearly 60 f/s. OnLive at loss rates between 2% to 18% has frame rates of about 35 f/s to under 20 f/s.

4.4 Application Comparison

In order to compare OnLive downstream traffic to video, YouTube was selected as a representative candidate of pre-recorded streaming video. YouTube is probably the most popular video streaming service on the Internet with over 6 billion hours of video watched each month [31]. Aside from its popularity, YouTube has the technical capabilities to allow videos to be uploaded and viewed in high definition 1080p (1920x1080 pixels) resolution.

Skype was selected as a representative candidate for live streaming video. Skype's core functionality provides Voice over IP (VoIP) services, but Skype also

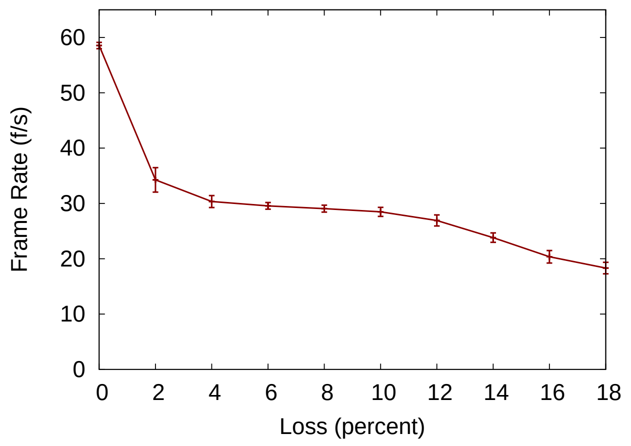

Fig. 20 Average frame rate with packet loss

provides video conferencing for camera-enabled computers. Skype is a one of the most popular VoIP applications, with 663 million registered users as of the end of 2010 [28]. Skype uses VP8 for video encoding [18], allowing high definition 720p (1280x720 pixels) resolution.

The PC in Section 3.2 runs Skype and YouTube for measuring streaming application performance, and Fraps to record application frame rates. The Skype application connects to a MacBook for two-way conferencing.

For the Skype testing, all non-essential applications were closed before the Skype video call was made. One participant was on the MacBook and the other on the PC. A short pause was given to establish the call, then the session was set to full screen mode before starting Fraps and the packet capture.

For YouTube testing, a video[7] of an omnipresent game, a two-player session of StarCraft 2, was chosen for the YouTube tests. The video could not start automatically at 1080p in fullscreen. So, using Google Chrome, the YouTube link was pasted into the URL bar at the same time the packet capture was started, and as the YouTube video came up in the browser, it was changed to full screen mode and then to 1080p.

Studies were conducted with YouTube and Skype over a range of network conditions, with the downstream bitrates analyzed. Figure 21 depicts a comparison of the bitrates for YouTube, UT and Skype. The x-axis is the time in seconds and the y-axis is the measured bitrate for each flow, calculated every 0.5 seconds. Each application is shown with a different trendline.

YouTube videos are pre-recorded, so unlike in OnLive or Skype, YouTube can potentially download as fast as the available capacity will allow. In fact, YouTube finishes the video download in about 90 seconds, so the x-axis is scaled shorter than for previous bitrate graphs. YouTube uses TCP, allowing the bitrate to expand to fill available capacity, while also having more variation due to congestion and flow control mechanisms built into the protocol.

Skype has a significantly lower bitrate than OnLive, only about 2200 Kb/s compared with about 6200 for OnLive. The bitrate ratio of OnLive to Skype, about 2.8 to 1, is on par with the ratio of their screen resolutions (1080p vs. 720p), about 2.25 to 1. Visually, the variances in bitrates for Skype and OnLive are similar.

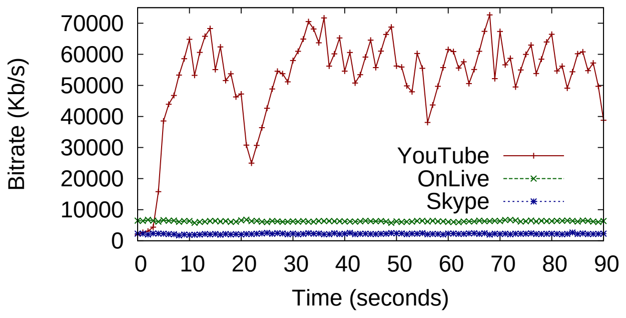

Fig. 21 Downstream bitrate for streaming applications

4.5 Traditional Game Comparison

In order to better understand what OnLive turbulence means to online games, this section briefly compares the results above with previously published results on turbulence in traditional (i.e., non-thin client) network games.

Warcraft is chosen for a third-person, omnipresent game, Madden NFL for a third-person, avatar game and Unreal Tournament (on the PC) for a first-person, avatar game, with complete published analysis available in Sheldon et al. [27], Nichols and Claypool [25], and Beigbeder et al. [1], respectively.

A complete comparison for all cases, upstream and downstream, using bitrate, packet size and inter-packet times is difficult, and, moreover, such analysis would likely not be informative because of the sheer number of details. Instead, median values are summarized for bitrate, packet size and inter-packet times. While previous analysis of the selected games only provides a summation of downstream and upstream turbulence, traditional network games are relatively symmetric downstream/upstream, anyway. For traditional network games,

[7] `http://www.youtube.com/watch?v=0NTeyF6wQUs`

general gameplay is used, as in our OnLive experiments, and not specific, isolated player actions nor other aspects of online gaming (e.g., login and server selection).

Table 4 provides a comparison of network turbulence for thin client games (OnLive), compared with traditional client games. The thin client games are shown with upstream and downstream values separated, while the traditional client games are shown with upstream and downstream values combined since separate data is not readily available. Downstream, thin client games have much greater turbulence across all genres, with about 100-700 times greater bitrates, 15-20 times larger packets, and 60-200 times more frequent packets. This is perhaps expected given that traditional game clients receive game object updates while thin game clients receive game frames. However, even upstream, thin client games have significantly greater turbulence, with 2-15 times greater bitrates, 2 times larger packets, and 5-20 times more frequent packets. Thus, any switch to thin client games from traditional client games must take into account a significant change in the network traffic both downstream and upstream.

Table 4 Turbulence for Network games (medians, thin: upstream downstream, trad.: upstream + downstream)

Game	Bitrate (Kb/s)	Pkt Size (bytes)	Inter-Pkt (msec)
Trad. Omnipresent	5	49	200
Thin Omnipresent	84 3756	130 970	9.0 1.0
Trad. Third person	14	77	75
Thin Third person	97 6339	146 1242	8.6 0.7
Trad. First person	67	75	45
Thin First person	124 6247	130 1203	8.0 0.7

4.6 Summary Comparison

A final summary seeks to put OnLive turbulence in the context of other streaming applications. As before, the focus is on downstream traffic which is often the bottleneck for many applications. Given the similarity in packet sizes and some similarity in bitrates, UT is chosen as the representative game for comparison. Second Life is selected as a representation of a multiplayer virtual environment (MVE), with complete published analysis from Kinicki and Claypool [20]. A main difference between a MVE and a traditional online game is in an MVE, users can provide custom content (e.g., avatar skins) to the game environment.

Table 5 presents a comparison of the relevant data. Traditional network games have relatively low amounts of turbulence, considering bitrates, packet sizes and inter-packet times. Virtual environments have an order of magnitude more turbulence, reflecting the flexible, dynamic nature of the player interactions. Live video has still three times more turbulence than virtual environments, owning to the frequent updates required to display video in realtime. While pre-recorded video has flexible data rates since there are fewer time constraints on clients watching the video, pre-recorded video generally expands to fill available capacity. Thin client games appear most similar in turbulence to live video, requiring frequent transmission of large packets in order to maintain a smooth frame rate depicting the game. This large turbulence, coupled with the real-time nature of online games, suggests meeting the quality of service requirements of thin client games is a challenge in terms of network planning similar to high-quality video conferencing.

Table 5 Turbulence for online applications (medians)

Application	Bitrate (Kb/s)	Pkt Size (bytes)	Inter-Pkt (msec)
Traditional Game	67	75	45
Virtual Environment	775	1027	9
Live Video	2222	1314	0.1
Thin Game	**6247**	**1203**	**0.7**
Pre-recorded Video	43914	1514	0.1

5 Conclusions

The growth in connectivity and capacity of networks presents the opportunity for thin clients, clients that primarily handle input and output and not computation, to be used for computer games. Understanding the network traffic characteristics, the network *turbulence*, is an important component for designing systems to support this new kind of traffic. Moreover, analyzing the frame rates of thin client under different network conditions can help determine appropriate network conditions to provide for acceptable player performance.

This paper provides a detailed study of the network turbulence of a prominent, commercial thin client game system – OnLive. Carefully designed experiments allow for comparison of network turbulence across game genres and streaming video applications and a range of network conditions. Frame rate analysis provides insights into the quality of experience over the network conditions, with predictions of player performance over network capacity constraints. Leveraging previous research allows comparison with traditional network game turbulence.

Analysis shows OnLive traffic has downstream turbulence most similar to high-definition, live video, with large, frequent packets and high bitrates. OnLive upstream traffic has far less turbulence than downstream traffic, but still significantly higher bitrates and packet rates than traditional upstream game traffic. OnLive turbulence does not respond to network perturbations much, but does adapt to changes in downstream capacity and in the frame rates provided to players when there is packet loss. Under even moderate conditions of loss and latency, OnLive does not respond to signals of congestion, using significantly more capacity than TCP. To avoid frame rate degradation impacting play, players need capacities of 7 Mb/s or higher. Playing OnLive even at the recommended 5 Mb/s results in about a 20% degradation in player performance, and the minimum of 2 Mb/s results in about a 50% degradation in player performance.

Future work can include testing additional games, possibly expanding the selection in the genres chosen or selecting other genres. Additional work could measure and analyze OnLive performance under other network loss conditions, such as packet loss over wireless networks. Other thin client game systems can be evaluated, such as GameNow. Research to classify and then potentially treat thin client traffic can build directly upon the results and analysis presented in this paper, with impacts assessed for actual thin client traffic. While this paper includes predictions of the effects of frame rate on player performance in OnLive based on past measurements of traditional clients, future work might include detailed user studies measuring the effects of frame rate on player performance in OnLive. The measurement data gathered in our experiments can be used to build thin client game traffic models. For this and other purposes, such as trace driven simulations, the traces from our experiments are available at:

`http://perform.wpi.edu/downloads/#onlive`

References

1. Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, Portland, OG, USA, September 2004.
2. M. Billinghurst, S. Bee, J. Bowskillb, and H. Kato. Asymmetries in Collaborative Wearable Interfaces. In *Third International Symposium on Wearable Computers (ISWC)*, 1999.
3. Fabrizio Biscotti, Brian Blau, John-David Lovelock, Tuong Huy Nguyen, Jon Erensen, Shalini Verma, and Venecia Liu. Market trends: Gaming ecosystem, 2011. ID Number: G00212724.
4. Mike S. Borella. Source Models of Network Game Traffic. *Elsevier Computer Communications*, 23(4):403 – 410, February 2000.
5. M. Carbone and L. Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communications Review*, 40(2), April 2010.
6. Yu-Chun Chang, Po-Han Tseng, Kuan-Ta Chen, and Chin-Laung Lei. Understanding the Performance of Thin-Client Gaming. In *Proceedings of IEEE CQR*, May 2011.
7. Wu chang Feng, Francis Chang, Wu chi Feng, and Jonathan Walpole. Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, November 2002.
8. Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the Latency of Cloud Gaming Systems. In *Proceedings of ACM Multimedia*, Nov 2011.
9. Kajal Claypool and Mark Claypool. On Frame Rate and Player Performance in First Person Shooter Games. *ACM/Springer Multimedia Systems Journal (MMSJ)*, 2007.
10. Mark Claypool. Motion and Scene Complexity for Streaming Video Games. In *Proceedings of the 4th ACM International Conference on the Foundations of Digital Games (FDG)*, Florida, USA, April 2009.
11. Mark Claypool and Kajal Claypool. Perspectives, Frame Rates and Resolutions: It's all in the Game. In *Proceedings of the 4th ACM International Conference on the Foundations of Digital Games (FDG)*, Florida, USA, April 2009.
12. Mark Claypool and Kajal Claypool. Latency Can Kill: Precision and Deadline in Online Games. In *Proceedings of the First ACM Multimedia Systems Conference (MMSys)*, Scottsdale, Arizona, USA, February 2010. (Invited paper).
13. Mark Claypool, Kajal Claypool, and Feissal Damaa. The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games. In *Proceedings ACM/SPIE Multimedia Computing and Networking (MMCN) Conference*, San Jose, CA, USA, January 2006.
14. Mark Claypool, David Finkel, Alexander Grant, and Michael Solano. Thin to Win? Network Performance Analysis of the OnLive Thin Client Game System . In *Proceedings of the 11th ACM Network and System Support for Games (NetGames)*, Venice, Italy, November 2012.
15. J. Faerber. Network Game Traffic Modeling. In *Proceedings of the ACM Network and System Support for Games (NetGames)*, Braunschweig, Germany, April 2002.
16. Sally Floyd and Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, February 1999.
17. P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube Traffic Characterization: A View From the Edge. In *In Proceedings of the ACM Internet Measurement Conference (IMC)*, San Diego, CA, USA, October 2007.
18. The H. Skype Moves to VP8 for All Video Calls, August 2011. http://www.h-online.com/open/news/item/-Skype-moves-to-VP8-for-all-video-calls-1318315.html.
19. Joeng Kim, Ricardo A. Baratto, and Jason Nieh. pTHINC: A Thin-Client Architecture for Mobile Wireless Web. In *Proceedings of the World Wide Web Conference (WWW)*, Edinburgh, Scotland, May 2006.
20. James Kinicki and Mark Claypool. Traffic Analysis of Avatars in Second Life. In *Proceedings of the 18th ACM*

International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Braunschweig, Germany, May 2008.

21. Albert Lai and Jason Nieh. On the Performance of Wide-Area Thin-Client Computing. *ACM Transactions on Computer Systems*, 24(2):Pages 175 – 209, May 2006.
22. T. Lang, G. Armitage, P. Branch, and H-Y. Choo. A Synthetic Traffic Model for Half Life. In *Australian Telecommunications Networks & Applications Conference (ATNAC)*, Melbourne, Australia, December 2003.
23. T. Lang, P. Branch, and G. Armitage. A Synthetic Traffic Model for Quake 3. In *ACM SIGCHI Advances in Computer Entertainment (ACE)*, Singapore, June 2004.
24. Yue Lu, Yong Zhao, Fernando Kuipers, and Piet Van Mieghem. Measurement Study of Multi-party Video Conferencing. In *Proceedings of the 9th IFIP TC 6 International Conference on Networking*, pages 96–108, 2010.
25. James Nichols and Mark Claypool. The Effects of Latency on Online Madden NFL Football. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Kinsale, County Cork, Ireland, June 2004.
26. Keith Packard and James Gettys. X Window System Network Performance. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, pages 207 – 218, San Antonio, TX, USA, 2003.
27. Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, Redwood City, CA, USA, May 2003.
28. Telecompaper. Skype Grows FY Revenues 20%, Reaches 663 Mln Users, March 2011. http://www.telecompaper.com/news/skype-grows-fy-revenues-20-reaches-663-mln-users.
29. Davy De Winter, Pieter Simoens, Lien Deboosere, Filip De Turck, Joris Moreau, Bart Dhoedt, and Piet Demeester. A Hybrid Thin-client Protocol for Multimedia Streaming and Interactive Gaming Applications. In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Newport, RI, USA, June 2006.
30. Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video Telephony for End-consumers: Measurement Study of Google+, iChat, and Skype. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, Boston, MA, November 2012.
31. YouTube. Statistics, 2013. http://www.youtube.com/-t/press_statistics.
32. Sebastian Zander and Grenville Armitage. A Traffic Model for the Xbox Game Halo 2. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Stevenson, WA, USA, June 2005.
33. Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose. Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications. *Elsevier Computer Networks*, 53(4):501–514, March 2009.