ENSC 427: COMMUNICATION NETWORKS
SPRING 2018

FINAL PROJECT
Comparison of gaming Client/Server paradigms: Peer hosting vs Dedicated Server.

https://ensc427team2.weebly.com/

Randeep Shahi
301224617
rsshahi@sfu.ca

Nathan Zavaglia
301260106
nzavagli@sfu.ca

TEAM 2

Table of Contents

1. Abstract

Online video gaming accounts for a significant volume of internet traffic worldwide. Traditionally, online game applications rely on a dedicated server to provide the information processing for the player-player and player-environment interactions; while the client provides the graphics rendering. Newer models make use of distributed hosting where the clients distribute the server load for data processing and one client acts as a control node (the host). This project compares the two paradigms with respect to delay, packet loss, dropped clients. Simulations will be performed using Riverbed Modeler.

2. Introduction

When it comes to playing an online video game, the worst thing that can happen is to lose a match solely based on the fact that the network connection used between players of the game was poor. Our project is to compare the two most common paradigms that are used for communications within online video games; peer-to-peer and client-server.

The first paradigm is peer-to-peer where one of the Users acts as the host, which will be referred to as the central node. All other nodes in the peer-to-peer network will directly interface with this central node. This is typically the most common type of network that is used in video games these day because of its lower cost.

The second paradigm is where there is a dedicated server that all of the clients directly communicate with. This is agreed to be much more preferable than peer-to-peer because it prevents the central node from having an advantage and prevents a terrible central node with bad QoS.

Most games these days opt for a peer-to-peer connection between players in a match as this is much cheaper that the alternative which is having a dedicated server. However, while the dedicated server may perform better in some aspects, the cost of the dedicated server may not justify its cost.

Our motivation for this research is based to observe the differences between peer-to-peer and dedicated. A common scapegoat for players to blame is that they lost a match because either the connection type was peer-to-peer or that the dedicated server was overloaded. Between these 2 paradigm, we would like to investigate the differences in terms of delay, packet loss, client link type, and dropped clients. We plan to conduct our simulations using Riverbed Modeler and will have a couple of different scenarios for each paradigm.

Simulation were performed on a workstation with the following specifications: Intel Core i7-6700K 4.0Ghz 4-core/8-thread CPU, 16GB DDR4 RAM, running Windows 10 Professional. This choice of workstation was made as it belongs to one of the authors and is substantially more powerful than any of the workstations located in the Simon Fraser University student labs. We will be using the UDP protocol as networking games rely on high bandwidth/low latency connections.

2.1 Current State of the Art

The most prevalent network architecture for online gaming is the client-server model [1]. Players, the clients, connect individually to a central server. The server manages all game state processing and manages object replication information. Information is broadcast (or multicast) to the clients. The clients are responsible for the actual graphics rendering. The player interaction data is unicast from the clients to the server.

However, there has been increasing use of peer-to-peer networks for gaming. Most often these are used in Massively Multiplayer Online games (MMO), rather than faster paced First Person Shooters (FPS). In MMOs, players often have limited game state information, usually in an isolated area around the player, this lends itself to the peer-to-peer architecture, as players can connect to peers that are close to them - close, meaning within the game environment [1].

2.2 Literature Review

Yahyavi and Kemme [1] provide an in-depth analysis of the various network architectures used in online gaming. The paper itself goes beyond the scope of this project. However, it does provide excellent background information necessary for the analysis of the various network topologies; in particular, an overview their strengths and weakness, and ways to compensate for those weaknesses.

Neumann et al [2] does provide relevant background information as to the challenges of utilizing a peer-to-peer network for online gaming. However, the paper is not peer reviewed. Moreover, it is an editorial and not a scientific work. This paper is included for the purposes of background only.

On the other hand, Knutsson, Honghui, Wei and Hopkins [3] provide an excellent analysis of peer-to-peer online game scaling, and latency. Their work centers on a simulated peer-to-peer network with up to 4000 players. Specifically analyzing the utility of a peer-to-peer network for online gaming. They developed a simple game and used a network emulator, known as FreePastry, to simulate the player interactions. They showed it was possible to maintain a delay of 150ms in a scalable peer-to-peer network of up to 4000 players - as stated in the paper, a 200ms delay is the maximum tolerable delay for an MMO. However, the paper is rather dated, being published in 2004. Computer hardware and simulation tools have advanced considerably in the last 14 years. More importantly the networking application code used in video games has also seen substantial refinement.

The work laid out in [4] while very dated, the paper is from 2002, provides an excellent starting point for simulations of game play networks. In this paper, Farber develops a simplified model for modelling the traffic of network games. He uses the game Counter-Strike, a first-person shooter, to develop a statistical model for packet interarrival times and latency using the client-server dedicated model. The simple model presented in this work will be used to inform our simulations, allowing us to develop metrics for evaluating network performance. Additionally, the paper provides statistical distributions for accurately modelling traffic in a simulation environment. We will be using the packet size, 80 bytes, as mentioned in data analysis in [4].

The final work in this review, given in [5], is not peer reviewed research. Instead it is a somewhat akin to an industry white paper. The author is one Yahn W. Bernier a software developer for Valve Software - a well-known developer of video games, including the above-mentioned Counter-Strike, as well as the Half-Life series. In this article, Bernier is discussing latency compensation in the client-server model of online games. This work was chosen as it provides an industry perspective on how latency is handled, and provides a basis for developing simulation models where latency is a quality of server (QoS) metric.

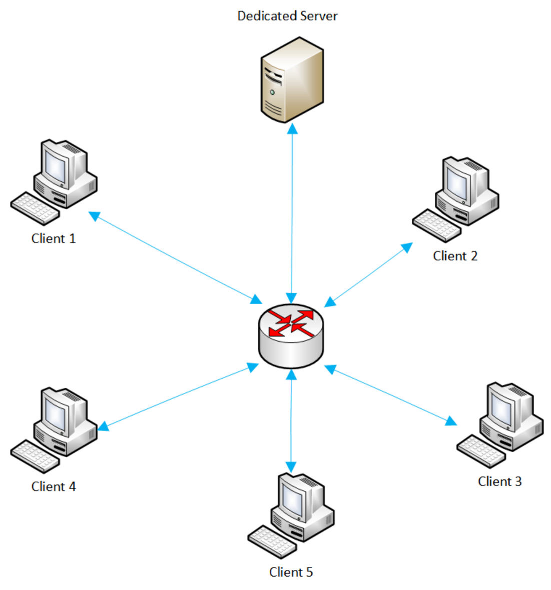
One of the difficulties in locating prior research in network gaming architectures is that most of the work is older - in some cases more 15 years old. This fact adds another motivation to this project. To produce simulations that make use of new network infrastructure models - in particular higher speed links between nodes and up to date routing protocols.

3. Main section

3.1 Problem definition

Below are the logical network topologies we will be investigating; figure 1 shows the traditional client server model where clients connect independently to the server, and the server manages the game state information and broadcasts game state updates to the connected clients.

Figure 2 shows the peer-to-peer model. Here the clients are interconnected. The major difference here is that each client is responsible for its own game state and receives updates from the other clients in the network. This means that the clients display both their own movements and virtual movements representing the movements of the other peers.



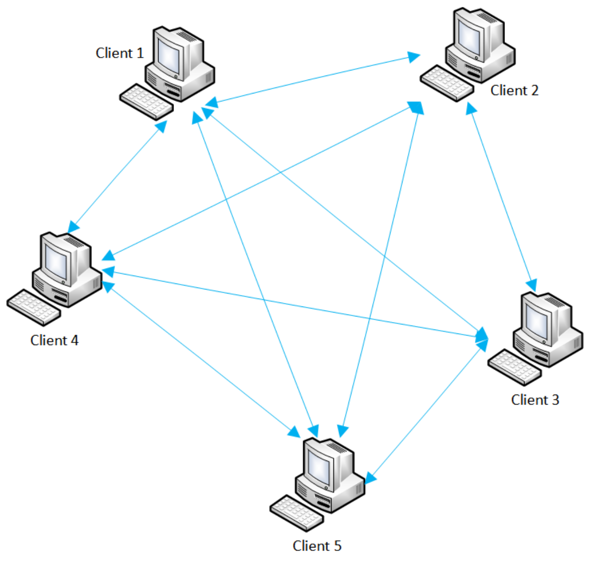


Figure 1: Dedicated server network

Figure 2: Peer to peer network

We plan to test each logical topology according to the following scenarios:

- A baseline scenario
- A scenario where some clients disconnect during the game
- A scenario where some clients join the network during the game

The qualities of interest are global average end-to-end delay, local end-to-end per client, jitter per client, and finally average dropped packets through the internet. The baseline scenario will be used as a reference.

3.2 Proposed simulation and set up

For our simulation, originally, we proposed that we would use ns3 to simulate our project. However, as we used ns-3 for our simulations, we found that we had trouble with extracting useful information from ns-3. We then decided to use Riverbed Modeler to perform our simulations instead and found that Riverbed was able to present information, using its inbuilt graphing functionality, in a very digestible way.

Our simulation consists of 2 main paradigms; client-server and peer-to-peer. We sought to compare these in regards to delay, jitter, and dropped packets. Within these 2 paradigms, we defined three variations of each:a scenario where normal operation occurs – this is the baseline, a simulation where 3 clients join, and a simulation where 3 clients leave. For each of these scenarios, we use eight hosts or peers. However, we only analyze three of the hosts across all simulations, Peer_1, Peer_2, and Peer_3. This is to simplify the data visualization and provide consistency in the results. The hosts that disconnect are Peer_5, Peer_6, and Peer_8. The choice to disconnect/connect 3 clients was to exaggerate the effect of the change in topology. Particularly for the client-server model, the effects of changing the number of clients is very small.

For each scenario, the links used between the ethernet workstations/server and the routers are 100BaseT 100Mbps links, and the links used between the router and the Internet node are PPP_DS3 44.6Mbps links.

Every simulation was run for 40 minutes of simulated time. This was chosen as a typical length of time spent per game session. However, some of the simulations ended prematurely due to the academic version of Riverbed Modeler only allowing a maximum of 50 million events per simulation. When this cap is reached, the simulation is terminated and the results shown. In all cases the simulated time was more than 30 minutes and for most simulations did reach 40 minutes of simulated time.

3.2.1 Baseline Scenario Riverbed Modeler Settings

Client server

Figure 3 shows the topology of the client-server simulation setup that was used. A notable design choice is the node titled Internet. This was used to be a simple model for the entire internet. This allows our clients to go through a network cloud which will more accurately represent a real-world example. The internet is modelled using the IP32 Cloud node (center of Figure 3) which supports up to 32 serial links and allows modification of network performance parameters such as packet delay and loss through the node. This simulates the effect of the connections taking place across the internet, without having to individually define a series of nodes between the clients and server. This is also used in the peer-to-peer network The IP cloud is configured with a 30ms average delay with an exponential distribution, and a maximum 0.5% packet loss parameter. These parameters were fixed for all simulations.

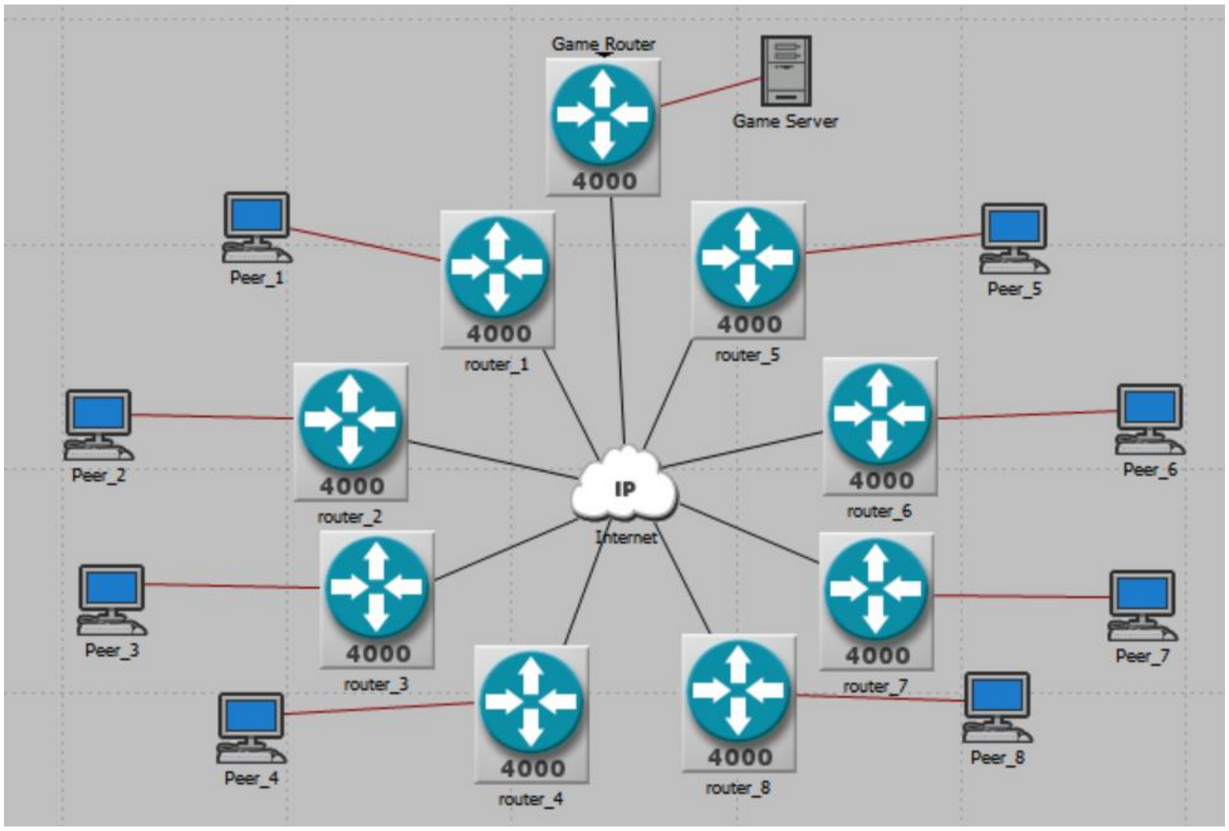
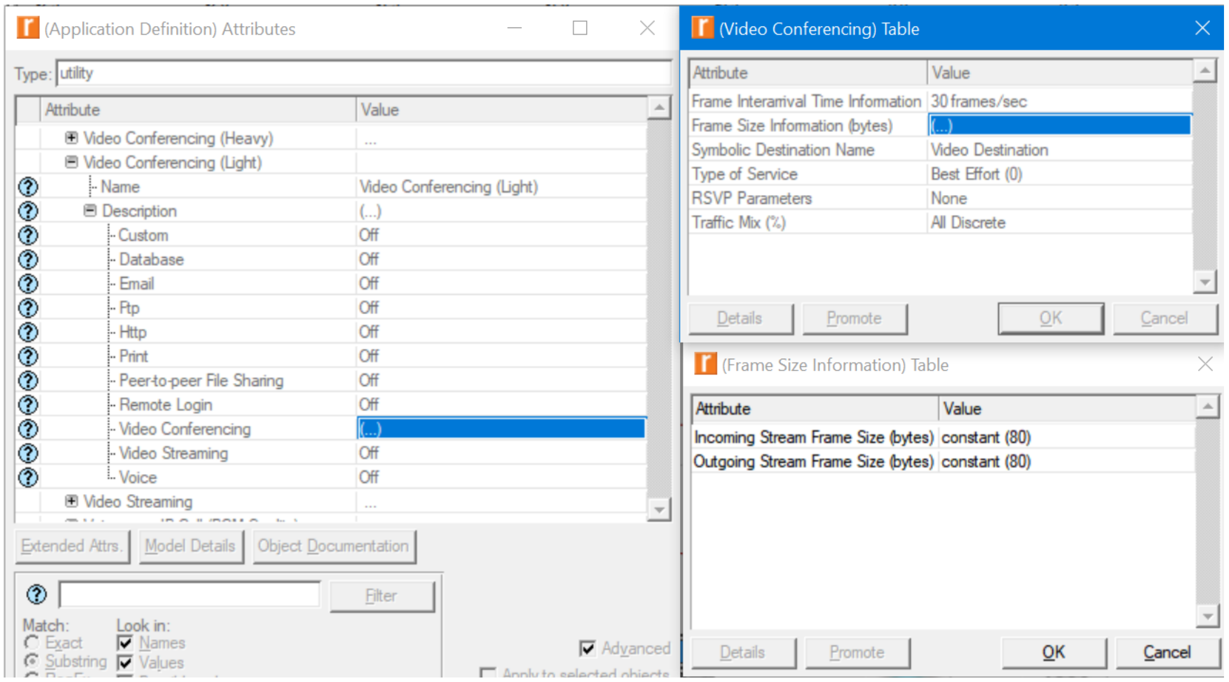


Figure 3: a screencap of Riverbed showing the network topology (Client-Server)

The server and clients for our simulation ran a modified version of an inbuilt application called Video Conferencing (Light). We modified this application in the following way to make it suitable for testing Client-Server based online gaming. Figure 4 below shows the application settings that were used to run the baseline client-server scenario with figure 5 showing the Profile that was used on each node. The other client-server scenarios have some modifications that will be brought up as necessary. The frame sizes (packets) were limited to 80 bytes for both incoming and outgoing. Video framerate was set for 30 frames/sec. This corresponds to an end-to-end delay of approximately 30ms which is reasonable for a game server that is only one or two hops away from the client. The rest of the delay is then taken up in the intervening cloud.



Type: utility

Attribute	Value
⊞ Video Conferencing (Heavy)	...
⊟ Video Conferencing (Light)	
Name	Video Conferencing (Light)
⊟ Description	(...)
Custom	Off
Database	Off
Email	Off
Ftp	Off
Http	Off
Print	Off
Peer-to-peer File Sharing	Off
Remote Login	Off
Video Conferencing	(...)
Video Streaming	Off
Voice	Off
⊞ Video Streaming	...

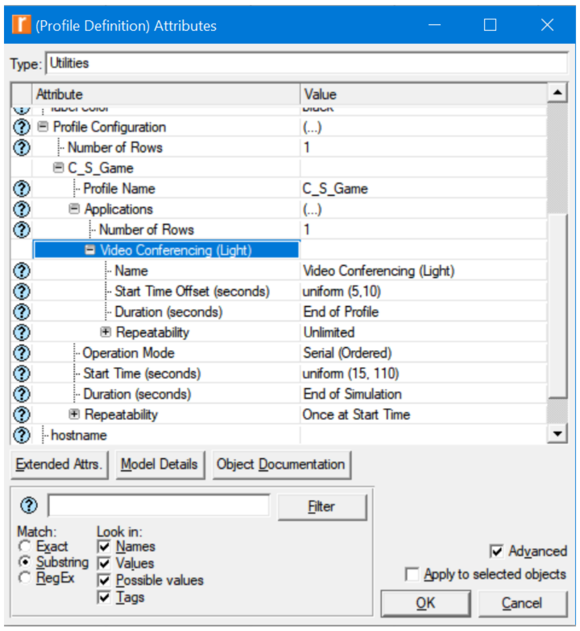
(Video Conferencing) Table

Attribute	Value
Frame Interarrival Time Information	30 frames/sec
Frame Size Information (bytes)	(...)
Symbolic Destination Name	Video Destination
Type of Service	Best Effort (0)
RSVP Parameters	None
Traffic Mix (%)	All Discrete

(Frame Size Information) Table

Attribute	Value
Incoming Stream Frame Size (bytes)	constant (80)
Outgoing Stream Frame Size (bytes)	constant (80)

Figure 4: client-server application settings



Type: Utilities

Attribute	Value
⊟ Profile Configuration	(...)
Number of Rows	1
⊟ C_S_Game	
Profile Name	C_S_Game
⊟ Applications	(...)
Number of Rows	1
⊟ Video Conferencing (Light)	
Name	Video Conferencing (Light)
Start Time Offset (seconds)	uniform (5,10)
Duration (seconds)	End of Profile
⊞ Repeatability	Unlimited
Operation Mode	Serial (Ordered)
Start Time (seconds)	uniform (15, 110)
Duration (seconds)	End of Simulation
⊞ Repeatability	Once at Start Time
hostname	

Figure 5: client-server profile settings

Peer-to-peer
Figure 6 shows the topology of the peer-to-peer simulation setup that was used in each variation of simulation. This topology is very similar to the client-server version with the exclusion of a server.

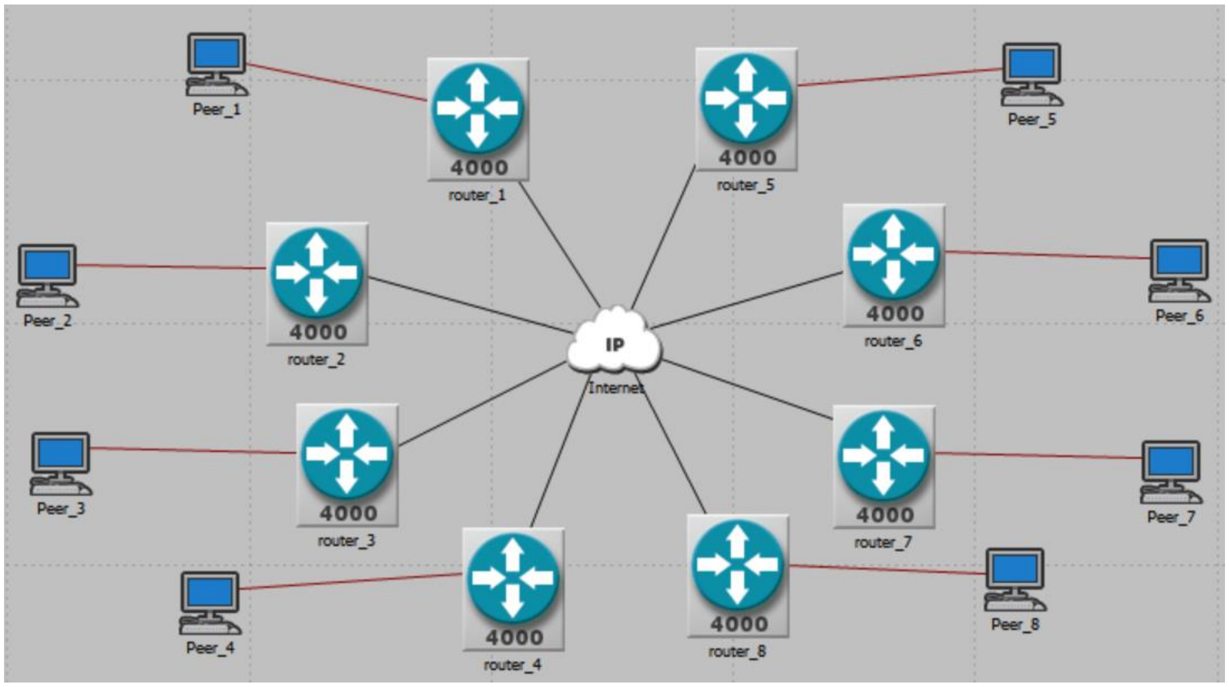
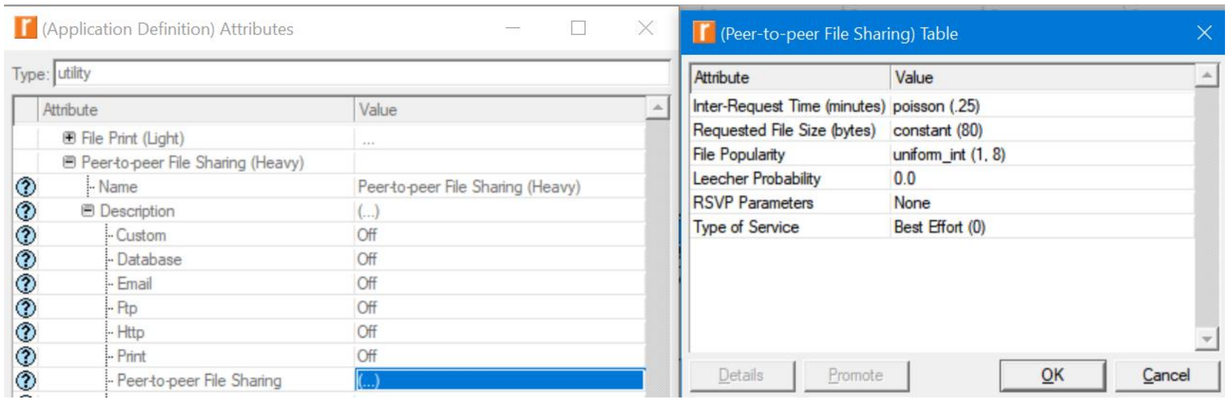


Figure 6: a screencap of Riverbed showing the network topology (Peer-to-peer)

The peer-to-peer node for our simulation ran a modified version of an inbuilt application called Peer-to-peer File Sharing (Light). We modified this application in the following way to make it suitable for testing Client-Server based online gaming. Figure 7 below shows the application settings that were used to run the baseline peer-to-peer scenario with figure 8 showing the profile's used for the baseline scenario. The other peer-to-peer scenarios have some modifications that will be brought up as necessary. The peer-to-peer statistics in Riverbed Modeler include a Download Response Time measure. This corresponds to a full round trip, from the time the request packet is sent till the first downloaded packet is received. It thus corresponds to the ping time (RTT), in video game parlance. Packet size (requested file size) is kept at 80bytes, and the inter-request time is kept as at a mean of .25min or 15sec in a Poisson distribution. This parameter only had minimal effect on network performance, so long as the inter-request time was larger than 0.0167 minutes.

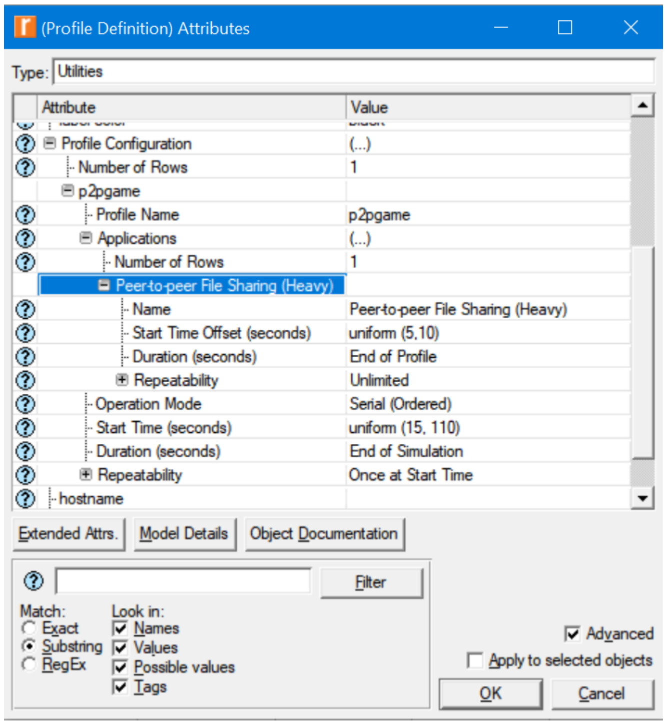

(Application Definition) Attributes
(Peer-to-peer File Sharing) Table
Details
Promote
OK
Cancel

Type: utility

Attribute	Value
⊞ File Print (Light)	...
⊟ Peer-to-peer File Sharing (Heavy)	
Name	Peer-to-peer File Sharing (Heavy)
⊟ Description	(...)
Custom	Off
Database	Off
Email	Off
Ftp	Off
Http	Off
Print	Off
Peer-to-peer File Sharing	(...)

Attribute	Value
Inter-Request Time (minutes)	poisson (.25)
Requested File Size (bytes)	constant (80)
File Popularity	uniform_int (1, 8)
Leecher Probability	0.0
RSVP Parameters	None
Type of Service	Best Effort (0)

Figure 7: peer-to-peer application settings


(Profile Definition) Attributes

Type: Utilities

Attribute	Value
Profile Configuration	(...)
Number of Rows	1
p2pgame	
Profile Name	p2pgame
Applications	(...)
Number of Rows	1
Peer-to-peer File Sharing (Heavy)	
Name	Peer-to-peer File Sharing (Heavy)
Start Time Offset (seconds)	uniform (5,10)
Duration (seconds)	End of Profile
Repeatability	Unlimited
Operation Mode	Serial (Ordered)
Start Time (seconds)	uniform (15, 110)
Duration (seconds)	End of Simulation
Repeatability	Once at Start Time
hostname	

Match: Exact [✓ Substring] RegEx

Look in: ✓ Names ✓ Values ✓ Possible values ✓ Tags

✓ Advanced

Apply to selected objects

Figure 8: client-server profile settings

3.3 Simulations and Results

3.3.1 Client-Server: Baseline

Figures 9 to 12, show the resulting graphs from running the baseline client-server scenario for about 33 minutes of simulated time. The simulated lasted about 33 minutes and from this we can reference a baseline for all of the modified client-server scenarios. The 4 graphs show, are:

- Figure 9, this graph is used to see the average end-to-end delay throughout the simulation, this delay is half the RTT, or half the ping time. Converging to about 60 milliseconds (ms)
- Figure 10, this graph shows the end-to-end delay, Here the delays converge to between 37 and 38ms, or 74 – 76ms ping.
- Figure 11, this graph shows jitter, or delay variation. Here the jitter is quite small and is converging to between 1.05-1.10ms
- And Figure 12, which shows the average traffic that is dropped through the internet. This trails off towards a value of 2 packets/sec dropped.

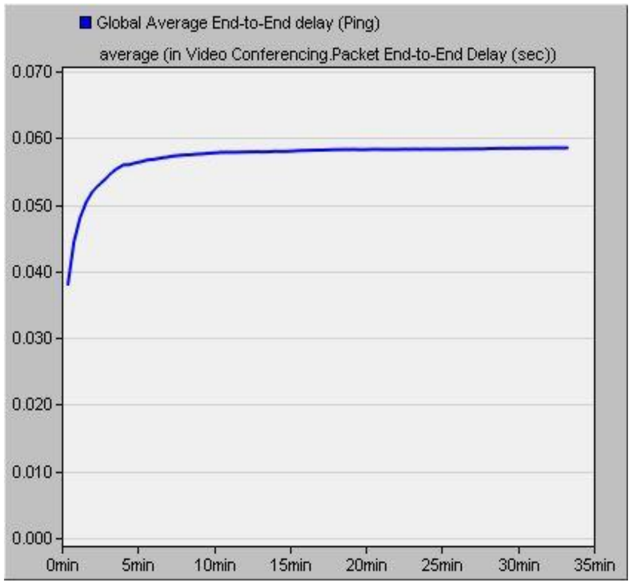


Figure 9: global delay during the simulation

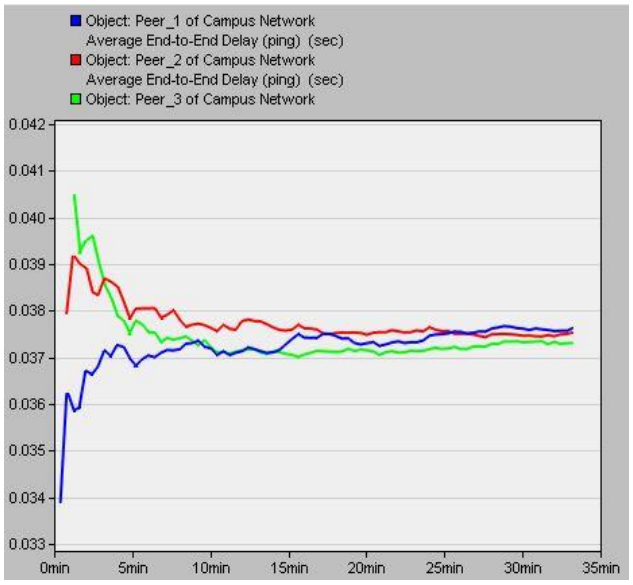


Figure 10: local delay per client

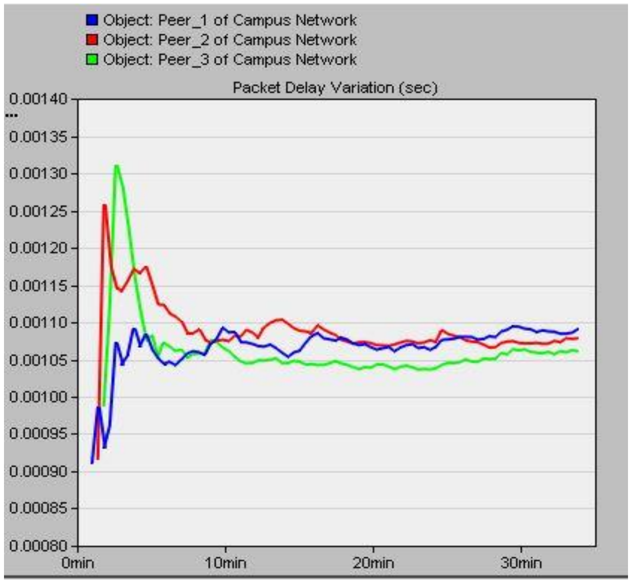


Figure 11: jitter per client

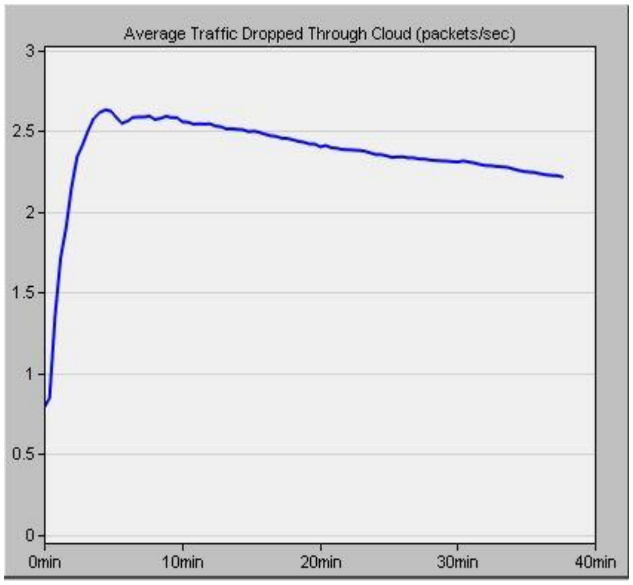


Figure 12: average traffic dropped through internet

3.3.2 Client-Server: 3 nodes disconnect from the network

In this scenario, clients Peer_5, Peer_6, and Peer_8 are dropped for the simulation each respectively at 10 minutes, 20 minutes, and 30 minutes respectively. Below, figures x to x show the graph that were produced from the simulation. Some notable differences between the below figure and the equivalent baseline graphs are the following:

- As the clients dropped out of the network, the global average end-to-end delay slowly began to drop (figure 13). This is the expected result.
- The local delay per client dropped very slightly (figure 14), but still remained approximately 37ms or 74ms ping
- The jitter per client dropped very slightly (figure 15).

- No differences were observed with average traffic dropped through the internet (figure 16).

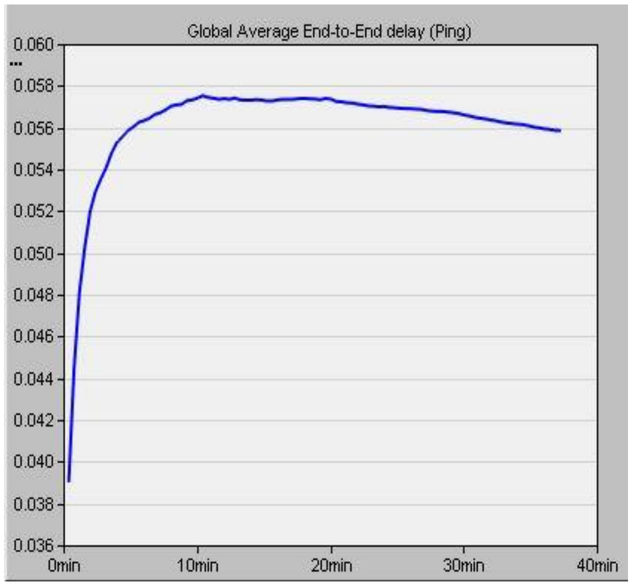


Figure 13: global ping during the simulation

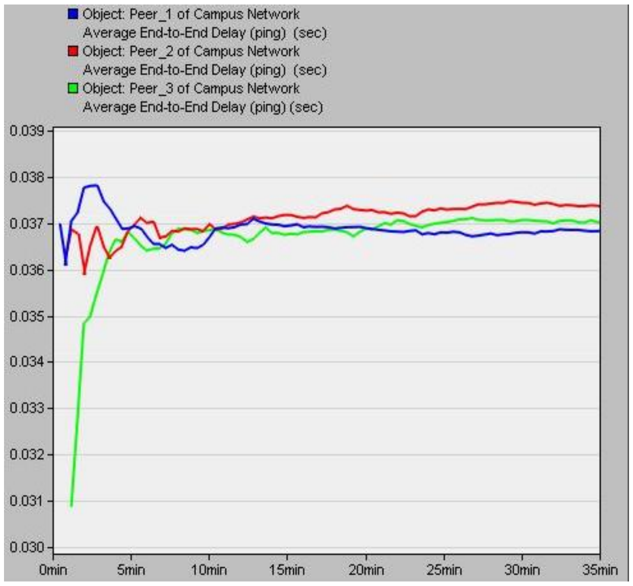


Figure 14: local ping per client

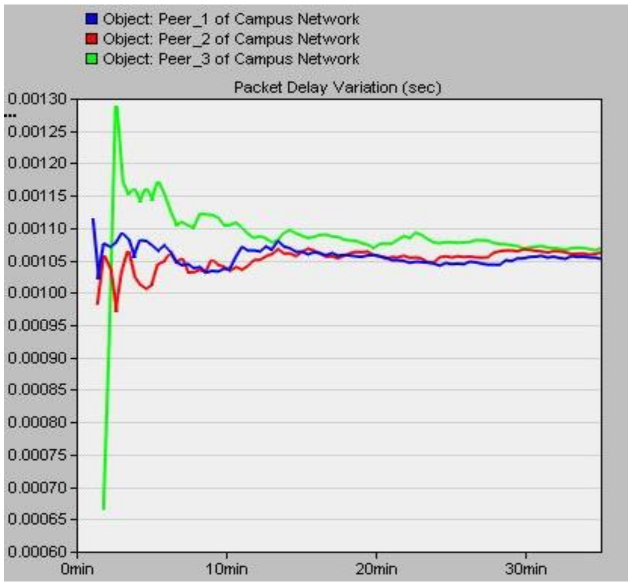


Figure 15: jitter per client

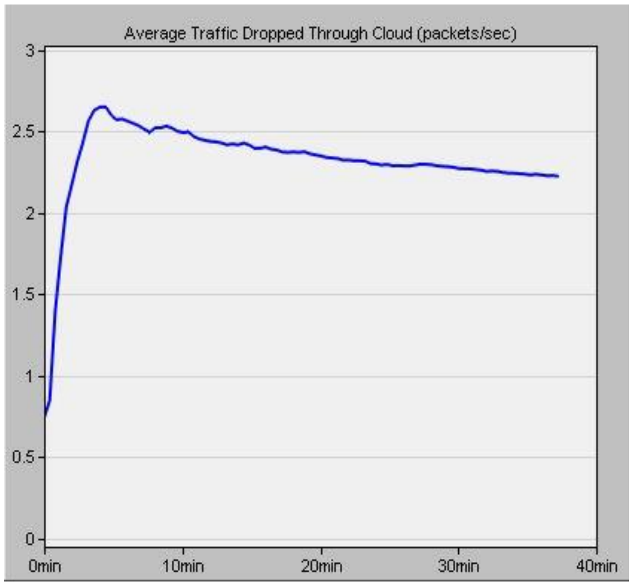


Figure 16: average traffic dropped through internet

3.3.3 Client-Server: 3 nodes connect to the network

In this scenario, clients Peer_5, Peer_6, and Peer_8 are added to the simulation each respectively at 10 minutes, 20 minutes, and 30 minutes respectively. Below, figures 17 to 20 show the graphs that were produced from the simulation. Some notable differences between the below figure and the equivalent baseline graphs are the following:

- As the clients added to the network, the global delay went down (figure 17).
- No changes were noticed for local delay time (figure 18).
- The jitter per client dropped very slightly (figure 19).
- Average traffic packet loss though the cloud dropped very slightly (figure 20).

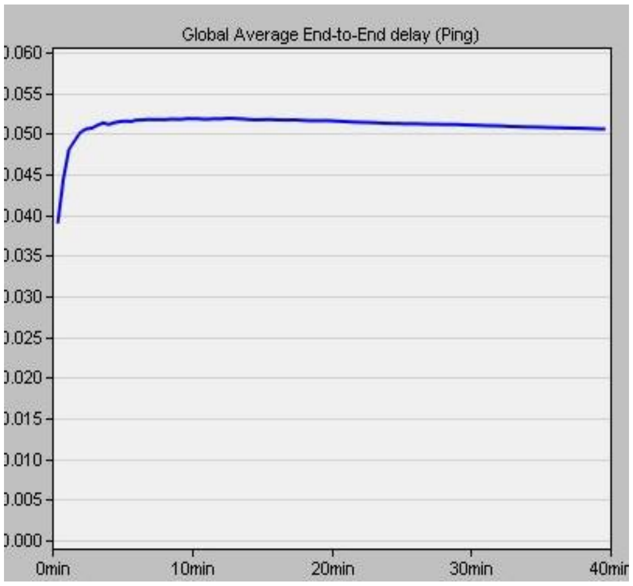


Figure 17: global ping during the simulation

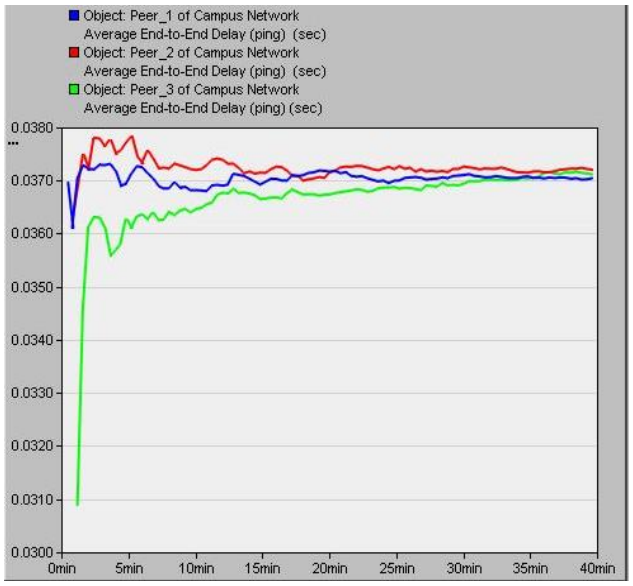


Figure 18: local ping per client

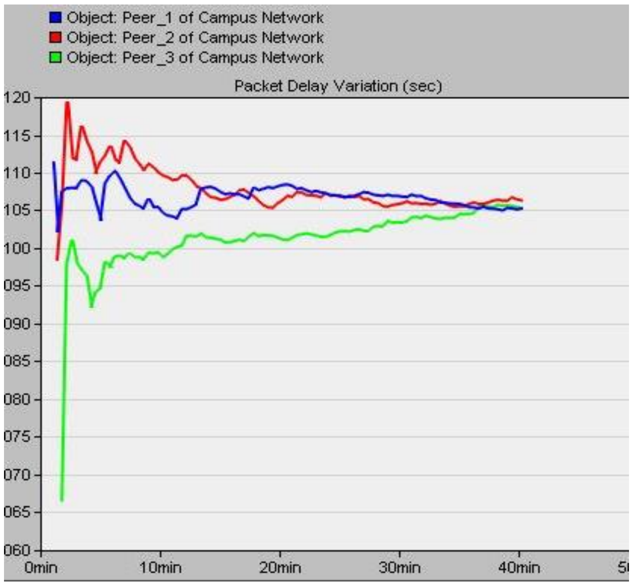


Figure 19: jitter per client

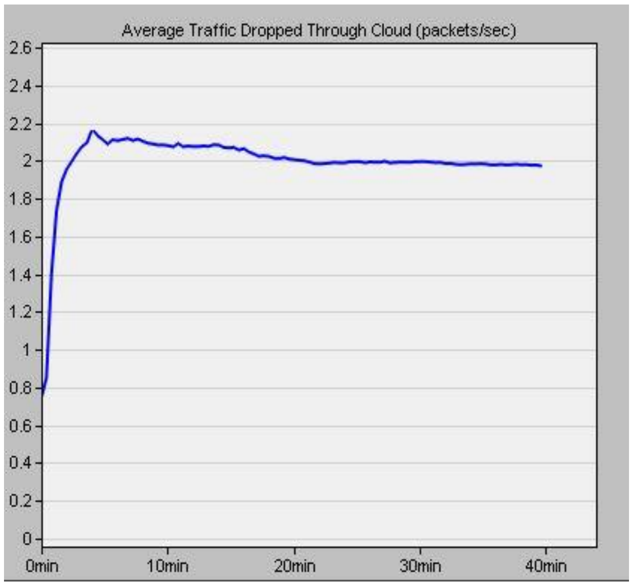


Figure 20: average traffic dropped through internet

3.3.4 Peer-to-Peer: Baseline

Figures 21 to 24, show the resulting graphs from running the peer-to-peer baseline scenario for about 33 minutes of simulated time. The simulated lasted about 33 minutes and from this we can reference a baseline for all of the modified peer-to-peer scenarios. The 4 graphs shown, are:

- Figure 21, this graph is used to see the average ping throughout the simulation which is converging to 90ms and is consistent with other simulations showing 150ms for peer-to-peer gaming [3]
- Figure 22, this graph shows the ping time for Peers 1 to 3. And can be seen to be between 90-95ms

- Figure 23, this graph shows jitter, which again is somewhat larger than the client-server, at an average of 25ms (see Discussion for why this graph is not a Riverbed Modeler graph)
- And Figure 24, which shows the average traffic that is dropped through the internet. Notable is that this is less than that observed for the client-server paradigm, converging to 1.2 packets/sec

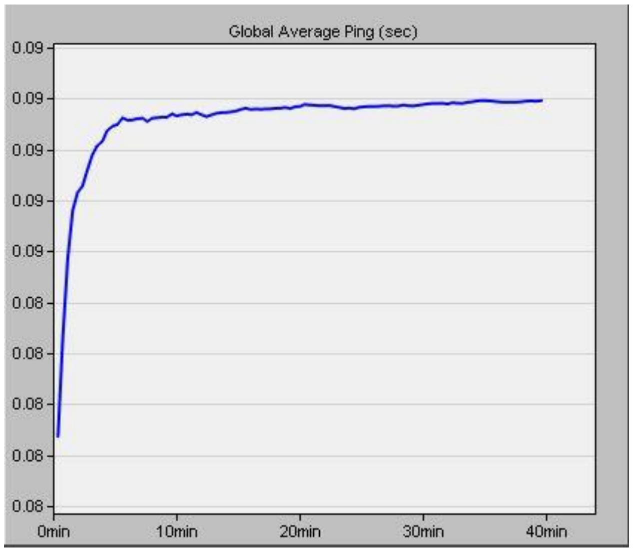


Figure 21: global ping during the simulation

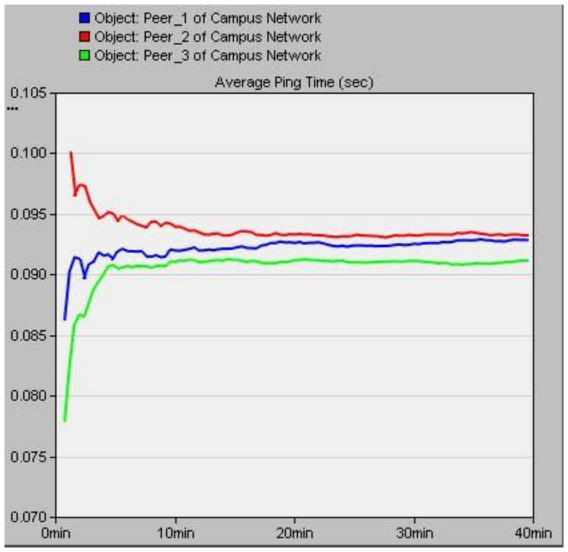


Figure 22: local ping per client

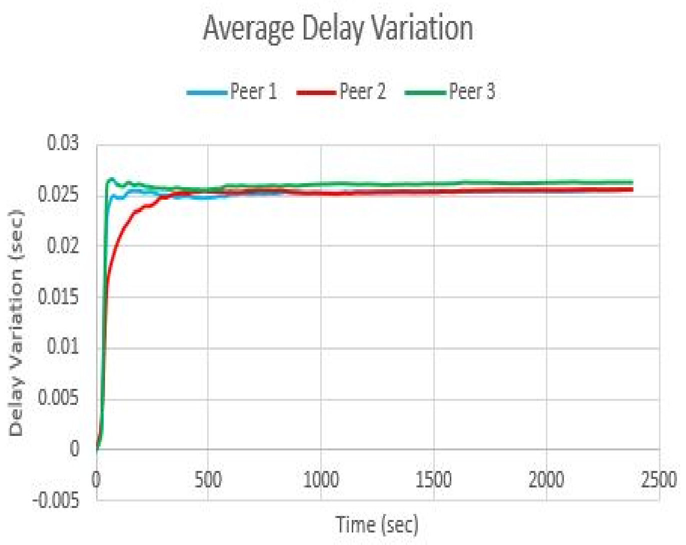


Figure 23: jitter per client

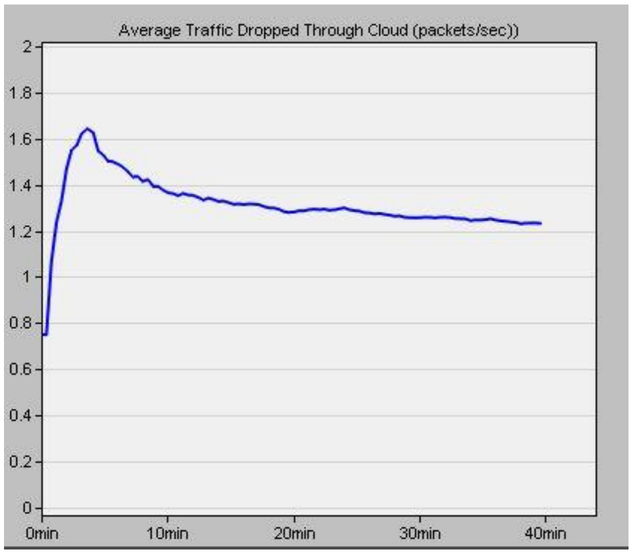


Figure 24: average traffic dropped through internet

3.3.5 Peer-to-Peer: 3 nodes disconnect from the network

In this scenario, clients Peer_5, Peer_6, and Peer_8 are dropped from the simulation each respectively at 10 minutes, 20 minutes, and 30 minutes respectively. Below, figures 25 to 28 show the graph that were produced from the simulation. Some notable differences between the below figure and the equivalent baseline graphs are the following:

- As the clients dropped out of the network, the global average ping was substantially higher, over 200ms (figure 25).

- The local ping per client also converges to approximately 200ms (figure 26).
- No changes were seen in local jitter (figure 27).
- Average number of dropped packets reduced quite considerably (figure 28).

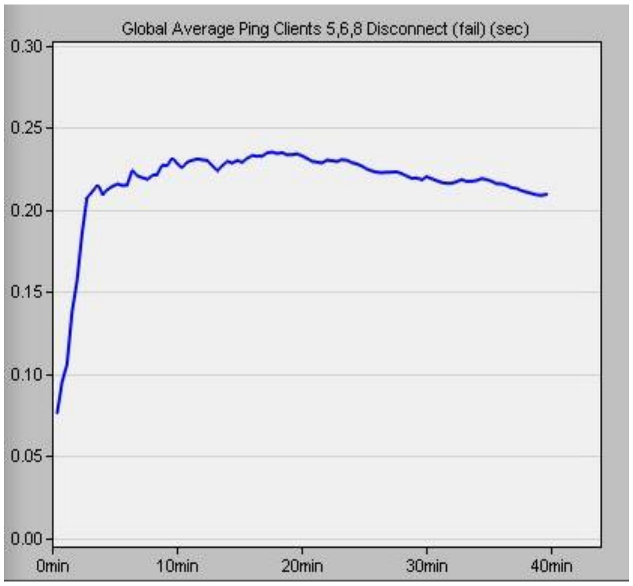


Figure 25: global ping during the simulation

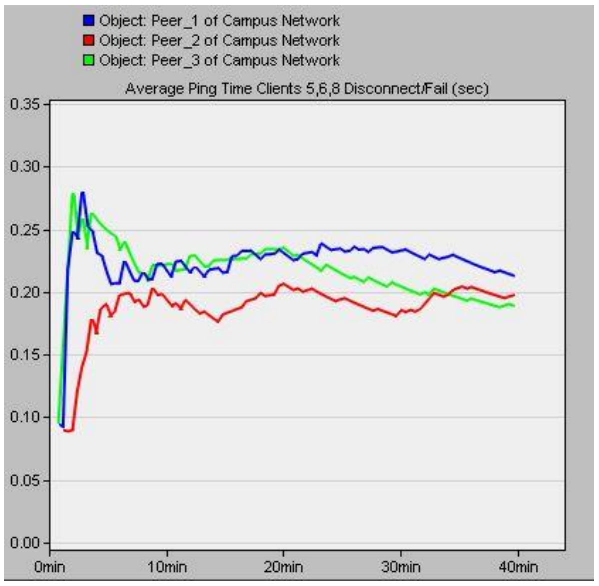


Figure 26: local ping per client

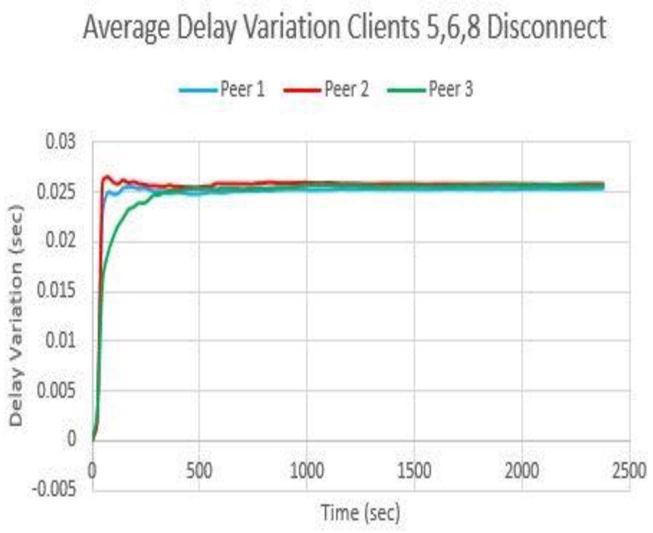


Figure 27: jitter per client

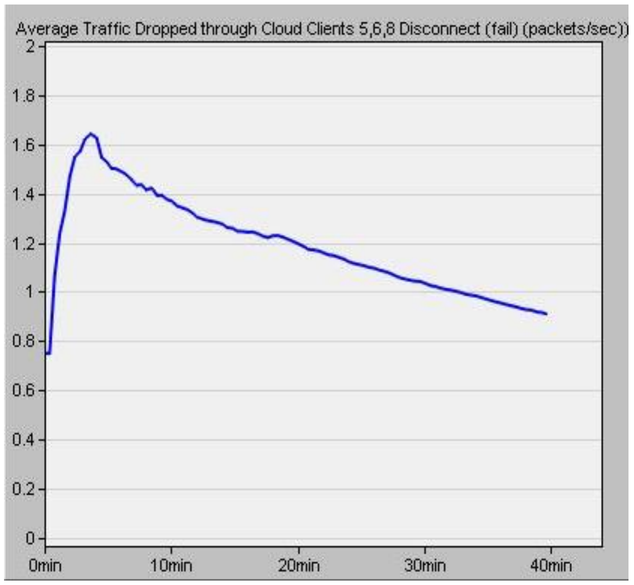


Figure 28: average traffic dropped through internet

3.3.6 Peer-to-Peer: 3 nodes connect to the network

In this scenario, clients Peer_5, Peer_6, and Peer_8 are added to the simulation each respectively at 10 minutes, 20 minutes, and 30 minutes respectively. Below, figures 29 to 32 show the graphs that were produced from the simulation. Some notable differences between the below figure and the equivalent baseline graphs are the following:

- As the clients were added to the network, the global average end-to-end delay slowly increased and stayed above 220ms (figure 29).
- The local delay per client dropped noticeably as clients were added appearing to converge to 220ms (figure 30).
- The jitter per client increased very slightly but remained approximately 25ms(figure 31).
- Average traffic dropped increases slightly as clients were added to the network (figure 32).

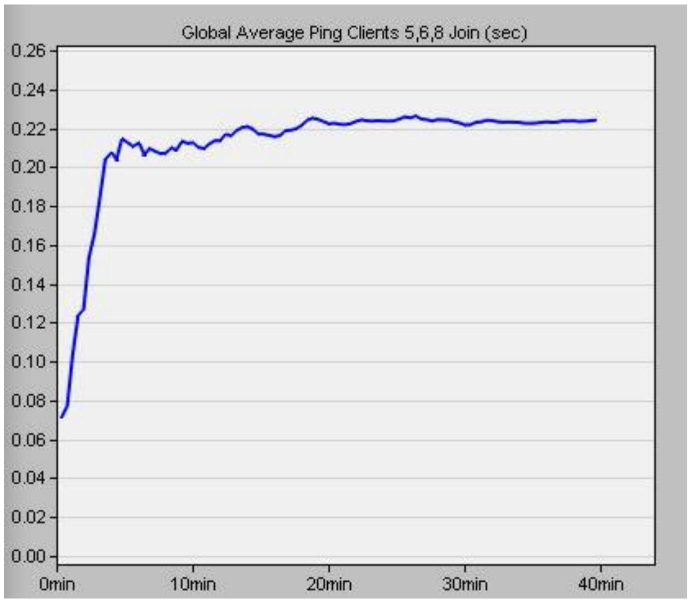


Figure 29: global ping during the simulation

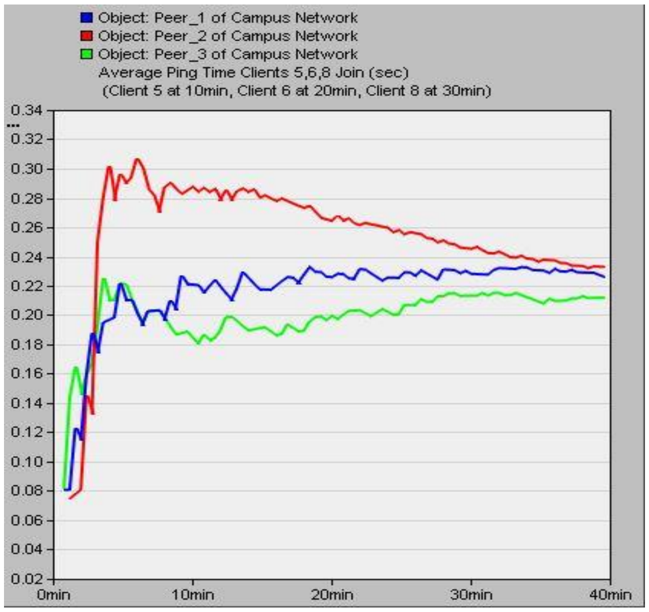


Figure 30: local ping per client

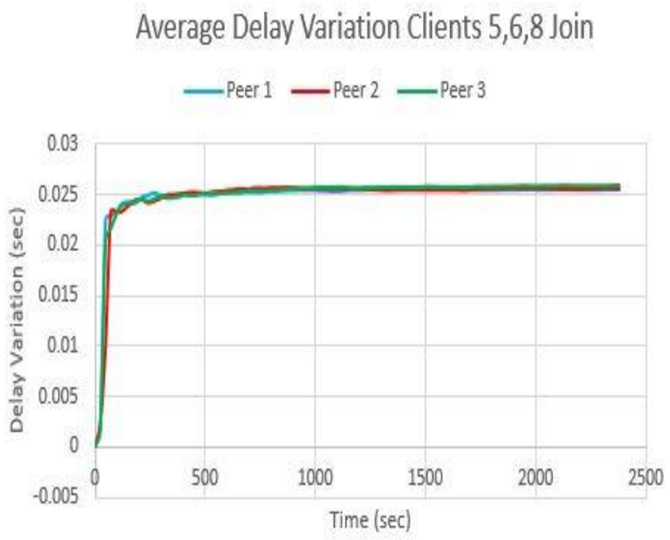


Figure 31: jitter per client

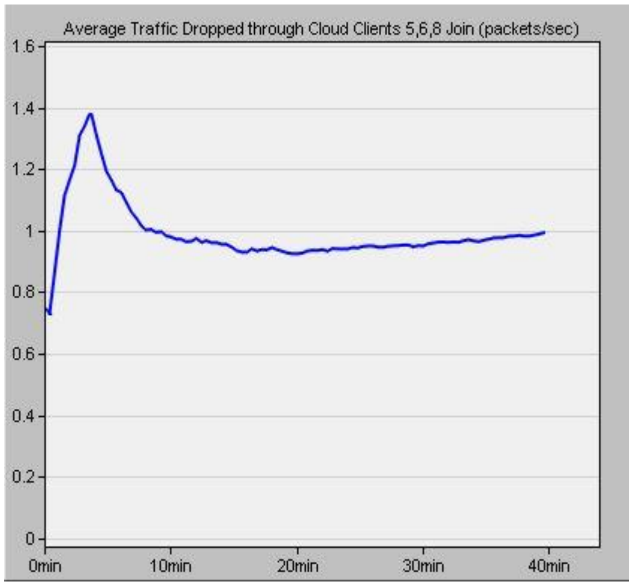


Figure 32: average traffic dropped through internet

3.4 Discussion

Taking these observations into consideration, some generalizations can be made.

Regardless if the topology is client-server or peer-to-peer, there were some similar trends in the respective scenarios. Whenever a client is added or dropped from a scenario. The global ping appears to increase or decrease respectively. This result makes sense as when clients are added to the simulated network, the server and/or seeding peers need to communicate with the newly added nodes which will increase global ping. A similar trend can be seen with ping per clients, though on a much smaller scale. Of note is the dramatic increase in delay, or ping, when a client joins or leaves the peer-to-peer network, from 90ms to approximately 200ms. It is unknown at this time why this occurs, but it may be due to the rebalancing of the network as the numbers of peers change, though this would require much deeper analysis to confirm. Riverbed Modeler Academic Edition 17.5 is limited to 50 million events per simulation, this caused some simulations to terminate early, and prevents running extended simulations in which several hours of game play are simulated. This makes analysis of these increased delays extremely difficult at the current time.

When comparing the jitter per client, once again we see an expected observation where additions of clients into a simulation tends to slightly increase the overall jitter and the removal of clients decreases the jitter as expected. Similar observations can be seen with packet loss on the internet. In order to properly analyze the results for the peer-to-peer jitter, the graphs had to be made outside Riverbed Modeler. This was due to the way in which the software prepares and displays results. Riverbed modeler shows the jitter for each individual peer connection. This makes graphical analysis cumbersome with many peers. Instead the results are averaged for each of Peer_1, Peer_2, and Peer_3 and then graphed in a similar fashion to the other results – all three hosts on the same graph. This was done using Microsoft Excel.

What our simulation have proved is that both a client-server and peer-to-peer network will have very similar performance characteristics given that neither of the scenarios are over saturated. However*,* the RTT, or ping, has much higher jitter on a peer-to-peer network, and is in general higher than the client-server model. The ping can also change dramatically as peers join or leave the network and may take substantial time to settle back down. We theorize that the reason that gamers complain against using peer-to-peer networks for gaming is due to these factors. Additionally, the player who is hosting the peer-to-peer match may not have sufficient network bandwidth for the number of peers – further confounding network latency. One outcome to note is that the results confirm the findings in [3], that peer-to-peer gaming networks are better suited to slower paced games, such as MMO games, due to the intrinsically higher delays between peers.

4. Conclusion

Online gaming is a large market and having a perfect balance with respect to delay, packet loss, jitter, and added/dropped clients. Taking these parameters into consideration, a game design team can implement the best solution. As of this writing, the team has grasped a solid understanding of how peer-to-peer is handled with video games and likewise with dedicated servers. The results in the paper above agree with the findings by Knutsson et al [3]. Peer-to-peer gaming networks are better suited to games where latency is not a primary concern for game performance. The RTT, or ping, in the peer-to-peer network rose to 200ms when peers left or joined the network – while this is sufficient for role-playing type games such as MMOs, this is simply not sufficient for faster paced action oriented games where delay becomes an important factor for game performance and personal enjoyment. However, more work is needed to investigate the cause of the dramatic increase in ping when peers leave or join a peer-to-peer game network. The academic edition of Riverbed Modeler lacks the functionality to run extended simulations of more than 50 million events, so it is impossible at the current time to determine if the delays settle back down after a period of time, or if they remain high.

5. References

[1] A. Yahyavi and B. Kemme, “Peer-to-peer architectures for massively multiplayer online games,” ACM Computing Surveys, vol. 46, no. 1, pp. 1–51, Jan. 2013.

[2] C. Neumann, N. Prigent, M. Varvello, and K. Suh, “Challenges in peer-to-peer gaming,” SIGCOMM Comput. Commun. Rev., vol. 37, no. 1, pp. 79–82, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1198255.1198269

[3] B. Knutsson, Honghui Lu, Wei Xu and B. Hopkins, "Peer-to-peer support for massively multiplayer games," IEEE INFOCOM 2004, 2004, pp. 107. doi: 10.1109/INFCOM.2004.1354485

[4] Johannes Färber. 2002. Network game traffic modelling. In Proceedings of the 1st workshop on Network and system support for games (NetGames '02). ACM, New York, NY, USA, 53-57. doi: http://dx.doi.org/10.1145/566500.566508

[5] Y. W. Bernier, “Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization,” Valve Developer Community. [Online]. Available: https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization. [Accessed: 09-Feb-2018].