

The Effect of Latency on Performance in Warcraft III

A Major Qualifying Project

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Eric Girard

Nathan Sheldon

Seth Borg

Professor Mark Claypool, Advisor

Professor Emmanuel Agu, Co-Advisor

Abstract

With the increasing interest in network games and network game traffic, understanding the effect of delay, jitter and latency on Multiplayer Computer Games can help design better networks and games. We studied Warcraft III under various amounts of jitter and latency. As a foundation for research, we separated out the tasks in Warcraft III into their basic components, and analyzed each individually. Through carefully designed experiments, we find that the effect of latency on the outcome of Warcraft III is negligible.

Table of Contents

1. INTRODUCTION.....	3
2. RELATED WORK.....	6
2.1. INTERNET TRAFFIC AND INTERACTIVE GAMES	6
2.2. QUANTITATIVE INTERACTIVE GAME RESEARCH.....	7
2.3. PREDICTING GAME POSITIONS	8
2.4. TECHNICAL RESEARCH TOOLS.....	10
3. METHODOLOGY	12
3.1. COMPUTER SETUP.....	12
3.2. PILOT STUDIES.....	13
3.3. EXPERIMENTATION	14
3.4. ANALYTICAL METHODS.....	16
4. ANALYSIS	17
4.1. APPLICATION LAYER ANALYSIS	17
4.1.1. <i>Battle Map 1</i>	20
4.1.2. <i>Battle Map 2</i>	24
4.1.3. <i>Jitter</i>	28
4.2. NETWORK ANALYSIS	30
4.2.1. <i>Incoming Data Streams</i>	31
4.2.2. <i>Outgoing Traffic Streams</i>	34
4.2.3. <i>Packet Timing</i>	37
4.3. USER ANALYSIS.....	39
5. CONCLUSIONS	41
6. FUTURE WORK.....	43
APPENDIX A: WARCRAFT 3 OVERVIEW:	45
APPENDIX B.....	49
APPENDIX C	51
APPENDIX D.....	53
APPENDIX E	55
GLOSSARY OF TERMS.....	56
REFERENCES.....	57

1. Introduction:

Over the last ten years, the Internet has been rising in popularity and capability at exceptional rates. In 1997, there were 36.6 million homes with computers and only 18 million of them had Internet access [CENS]. In 2000, there were 51 million homes with computers and 41.5 million of them had Internet access, and network speed capabilities have risen to new heights. It is becoming increasingly popular for households to have broadband Internet connections via cable modems and DSL lines.

The primary type of traffic on the Internet today is HTTP traffic [CAIDA02], and the Internet has evolved to handle this traffic very well. However, along with the growth in popularity and capability of the Internet have come many new types of Internet traffic. Much of this new traffic is generated by “real time” application, such as live audio or video broadcasts, where the computer must handle a steady flow of new information without interruption [WEB-R02]. Streaming media traffic differs from other real-time traffic because delayed or lost information packets can simply be ignored without causing the application to terminate. The lost data shows up as jitter¹, skipped frames, or sound glitches to the user. Fully interactive, real time applications, such as tele-surgery, where a doctor performs surgery remotely over a network, have stringent data integrity requirements where data should not be delayed or lost [PJMI02].

While tele-surgery and similar real-time interactive applications are becoming more common, the most popular real-time interactive applications are online multiplayer computer games. Sean McCreary and KC Claffy observed that games make up around half of all of the top 25 types of non-traditional traffic on the Internet [SMKC02]. The real time interactivity involved in many games also makes them one of the least tolerant types of traffic on the Internet to latency.

Knowing how games and other real-time applications react to latency and loss is the crucial first step in designing the next generation of network hardware and software that will be able to support their needs. General classifications of games by tolerance levels will allow users to make better-informed decisions and give engineers the ability to build networks that provide suitable quality for classes of games or other applications.

¹ Please see the glossary of terms for definitions of unknown words or acronyms.

Hundreds of thousands of people play online multiplayer games every day. Blizzard Entertainment's Battle.net service, for example, hosts tens of thousands of games at any one time, and from only five game titles [BNET]. Knowledge of how network related issues, such as latency and packet loss, affect the usability of games can be of great use to many groups including: the companies that make these games, Internet Service Providers (ISPs), and the research community at large. The ISPs, in particular, are affected because they could likely charge customers based on their requested quality of service (QoS), geared for people planning to use their Internet connection for real-time activities.

It is unlikely that First Person Shooter (FPS) games have the same network requirements as Real Time Strategy (RTS) games or Massively Multiplayer Online Role Playing Games (MMORPG), because of variations in style of play. An FPS is a game in which you control a single character, seeing the world through their eyes. FPS games are sensitive to latency, as an FPS sends coordinates as part of a command, such as "my position is (X_1, Y_1, Z_1) and I am shooting in direction Y right now." Exact positioning and timing is required, because the target must still be at the location where the player aimed in order for the shot to hit. Research into the playability of First Person Shooter (FPS) games for example, suggests that latencies of up to 150 milliseconds are tolerable for that type of game [GA01-1].

An RTS allows the player to command multiple units, typically from an overhead view. These game styles allow for different interpretations of commands than in an FPS. For instance, in an RTS game, the client sends a command to attack as "unit X attack unit Y," and the game can and will execute this command regardless of exact position as long as both units are still capable of executing the command. Therefore, slight delays generally will not cause degradation in game playability.

MMORPG games are a relatively new addition to the game world, and they are based entirely around large online multiplayer environments where players interact both with each other and with the game world. Massively multiplayer games share some aspects with first person shooters, like first person view and real time interaction, but interaction with other players is typically cooperative, reducing the effect of latency because the client and the game server are the only important nodes.

Different types of games have different quality of service requirements, but it is relatively unknown what these requirements are. Our research studies the effect of latency on the usability and playability of multiplayer computer games using a game genre that tends to be able to tolerate higher latencies, namely RTS games. Our hypothesis is that as latency increases a player's performance in a Real Time Strategy Game will decrease, but to a lesser extent than in an FPS. To test our hypothesis, we quantify the effect latency has on playability by analyzing the results of controlled research experiments designed to measure quality of service over a range of induced latencies. In order to accomplish this, we develop multiple criteria for measuring the responsiveness and playability of multiplayer games. We use these criteria, along with direct user observations, to measure a multiplayer game called Warcraft III, one of the newest and best selling RTS games available [BEPR], and which is likely to represent the state-of-the-art network usage.

The rest of this report is laid out as follows: Chapter 2, Related Work, describes current research regarding online computer games and how it relates to this project, and present tools for measuring and implementing our experiments. Chapter 3, Methodology, details the process we used for evaluating Warcraft III, including the actual layout of our test network and the tools used for emulation and measurement. Chapter 4, Results and Analysis, presents our findings from these tests. Chapter 5 draws conclusions about the effects of latency on Warcraft III specifically, and RTS games in general, based upon our results. Finally, Chapter 6, Future Work, lists what we believe to be the next logical steps necessary to expand upon the results of this project.

2. Related Work:

This chapter will provide an overview of related areas of inquiry that other researchers are exploring. We classified these works into four groups. The first group is research into ways of handling Internet and game traffic to improve performance. The second area of research is focused on quantifying the network characteristics of online games and how network performance affects games. The third area of research looks at how games can be designed to handle latency in the game, without having to modify the network. Lastly, we look at the tools we considered using to perform our research.

2.1. Internet Traffic and Interactive Games

Internet traffic can be divided into a few major categories. The first is general-use traffic, such as web browsing or email. The second is streaming media, such as video clips or video conferencing. The third is time-sensitive, interactive systems, such as Tele-surgery or multiplayer games [SURG]. Latency, jitter, and/or packet loss can adversely affect all types of traffic, but some types of traffic are more sensitive to these conditions than other types of traffic.

The first category of traffic comprises email, web browsing, FTP, and similar applications. This category accounts for the majority of all traffic on the Internet [CAIDA]. These applications have traditionally been insensitive to delay because they are not operating in real time. Furthermore, many people connect to the Internet via dial-up modems that can take time to download a particular web page. To these Internet users a few seconds of response time for such content is noticeable but acceptable.

The second category of traffic comprises streaming media and similar real time applications. These require consistent bandwidth and latency, since they rely on consistency to determine how long to buffer data before playing the media, as well as how much data can be downloaded consistently to keep the buffer as full as possible while playing. If there is a reduction in bandwidth in the middle of a rapid action scene in a streaming movie, it will be immediately noticeable.

The third category of traffic comprises real time and interactive applications, such as tele-surgery and most multiplayer games. This category of traffic requires low latency and low jitter. These applications are constantly sending and receiving data to keep the

states of all participants' games coordinated with each other. Therefore, the packets are being exchanged periodically, instead of lumped together in order to reduce the number of packets being sent, and therefore the average packet is relatively small. This also means that increased bandwidth will not benefit most games much, because the designers expected the user to have limited bandwidth.

Some groups foresee multiplayer game traffic having an impact on the Internet similar to how HTML traffic had an impact on the network infrastructure of the early Internet. Researchers have studied a variety of topics related to network optimization for multiplayer video games. Peter Key and Derek McAuly, for example, investigated a proposed system that would charge for increased quality of service [PKDM02]. This approach would allow the users to compete when congestion becomes a problem by allowing them to bid for service. Another area of research interest has been in designing routers that identify specific types of traffic and forward real-time, interactive traffic ahead of other types of traffic [PKT]. There has also been research into proxies that manage multiple different game connections to provide fairness, congestion control, latency minimization, and fast rerouting if a particular router becomes unavailable [MM02]. Another paper proposed the creation of a game protocol that would standardize common aspects of games such as location to allow special proxies to route that information faster [DB02].

2.2. Quantitative Interactive Game Research

A study done in February of 2000 observed that, on average, six of the most popular multiplayer games generate about 4% of the packets, and create 1% of the throughput on some major Internet backbones [SMKC02]. The same study showed an average packet size for HTTP traffic to be 710 bytes while the average packet size for the six most popular games was 122 bytes. Another defining characteristic of game traffic is that the majority of it is UDP rather than TCP. Most traditional Internet traffic uses TCP due to its built-in congestion control and guaranteed delivery. However, for applications such as games and streaming media, these built in functions can be a hindrance. Games often need to minimize their latency, and UDP allows designers to implement their own strategies to fulfill this requirement.

For most FPS games, players consider the highest acceptable ping times in a FPS to be around 150 milliseconds [GA01-a]. FPS games tend to be fast paced, and need precise aiming. The lower ping times a player has, the better they are able to perform because their moving targets are less likely to dodge. If there is high latency, they might have an added inaccuracy [GA01-b]. Ping times above 150 milliseconds will cause players to seek out servers where lower ping times provide a better game experience.

Different types of games allow designers to handle delayed response times differently. RTS games tend to be able to handle high latency, perhaps up to 500 milliseconds, without playability suffering too much [JF02]. This is because exact positioning is often not necessary. This allows RTS game designers the choice of slowing down all players to the slowest player's speed in order to maintain playability. This is unlike FPS games, since they are faster-paced, imposing one player's latency onto everyone else greatly affects playability.

2.3. Predicting Game Positions

One way of reducing the effect of latency is via dead reckoning to predict game positions of other players when data is delayed or missing. This technique is known as "dead reckoning." The basic premise of dead reckoning is to note the previous position and apply the distance and direction traveled since then to infer the current position. People have used dead reckoning for ocean and land navigation for centuries.

Other computer simulations have used dead reckoning. For example, the Department of Defense performed early research into computer-based dead reckoning [SIMNET]. In an attempt to run more realistic tank simulations, they used the Internet to connect multiple simulators together in a project named SIMNET. Each time a SIMNET simulation had to create a new object, it would send out a packet to the other simulators that included specifying a dead reckoning algorithm to use with the object. Each node in the simulation, including the controlling node, would then use that algorithm to predict the location of the object within the game. The controlling node would then periodically compare the actual position to the inferred, and if it exceeded a set threshold, it would then send an update to the other nodes. This reduced traffic on the network and allowed the simulation to deal with network delays. Many multiplayer online computer games

have similarly applied the principles of dead reckoning as a way to reducing the apparent effects of network delay on interactivity by predicting what might happen when data is not received in time

In principle, a well-crafted dead reckoning system could help reduce the effects of latency; however, the main difficulty is finding an algorithm that can predict movement well. Movements of objects in straight lines at constant speeds are simple to predict. However, movements controlled by human players in computer games are much harder to predict. Most games allow players a wide range of movement possibilities. Furthermore, humans – especially game players – rarely follow a simple pattern of movement. Finding algorithms to predict these erratic movements is a significant area of research into interactive computer games.

The other major issue with dead reckoning is how to determine the appropriate action to take when the predictions and reality no longer match. Early games often simply moved the object to the real location which resulted in objects moving from one location to another instantly and unrealistically. One proposed variation of dead reckoning, called targeting, attempts to address this issue by adjusting the predicted locations towards the actual location as reported by the client [CH01]. In this particular scheme, the solution is to increase the velocity of the object in order to catch up to its true position. The “targeting” method relies on knowledge of the latency present in the connections between clients and the server. Based on the observed latency, the server and other clients know the actual position reported by the player, as well as the time they received the action. The server and clients initiate the action when they receive it. They also determine the actual position. They then accelerate the object from its predicted position in the game to catch up to the real position of the object. This still leaves several other paradoxical situations to deal with, such as having an object move into the line of fire. Unit targeting, as opposed to positional targeting, however, addresses the major issue of smooth transitions from predicted to real positions in the game.

These attempts to reduce the effects of latency in games are still not sufficient by themselves. While position predictions may help some online games, other games cannot efficiently use position predictions because of the lack of predictable player behavior.

Furthermore, if designers do not carefully formulate and test dead reckoning algorithms, they can have a distinctly negative effect on game play.

2.4. Technical Research Tools

We used many tools in order to fully set up our experiments, conduct our research, and analyze the results. Among the tools are the computer systems themselves, packet sniffers, and network traffic controllers.

Battle.net Server: Battle.net [BNET] is a service provided by Blizzard Entertainment to its customers, allowing them to chat with and challenge players from all over the world. Battle.net supports all of Blizzard's most well known multiplayer games. When users play through Battle.net, the specifics of how the clients communicate are observably different from a game played on a LAN.

In order to observe this and help us assess our results, we considered running a local Battle.net Server for research purposes. The BNETD Project [BNETD] was the only resource we could find that had available code for Battle.Net server simulation. At the time of our research, however, they did not yet support Warcraft III. Furthermore, Blizzard had filed a lawsuit against BNETD over intellectual property rights. Thus, we were restricted to researching mainly local LAN games.

Network Traffic Controllers: Network Traffic Controllers are common tools in the analysis of networks and client-server systems. They provide a controlled environment for research by giving the user the ability to "shape" the traffic by controlling latency, packet loss, and many other aspects of traffic flow. There are many "traffic shapers" to select from, including software solutions such as NISTNet [NIST] and Dummy Net [DUMNET], and hardware solutions, such as Packeteer's PacketShaper [PKT].

NISTNet and DummyNet are very similar in what they can accomplish. Both allow control over queue sizes, packet loss, latency, and directional distinction, or the ability to differentiate between multiple NIC cards. However, DummyNet is a FreeBSD application, while NISTNet is a Linux application. Since we are more familiar with the Linux operating system, and NISTNet requires less computational resources from the computer, we decided to use NISTNet.

PacketShaper is a high quality hardware solution that provides all the control of the software solutions, as well as many other features for which we needed to find other software. We did not use PacketShaper because the cost was too high for our budget.

Packet Sniffers: Packet sniffers, like network traffic control systems, come in a few different varieties. PacketShaper, in addition to its “traffic shaping” features, also includes packet analysis capabilities. We found a flexible solution in an application named Ethereal [ENA], which allows for sniffing packets that are traveling through a network. Ethereal allows for full capturing of all packets going into or out of a machine. It is free, open source, and runs on Windows, UNIX, and Linux systems. For these reasons, we decided to use Ethereal for our network analysis needs.

3. Methodology:

There are three main phases to our methodology. Phase 1 consisted of setting up the computer systems, testing the tools and analysis software, and general preparation for the research. Phase 2 consisted of our pilot studies, which were small-scale tests that helped us to design more effective experiments. Phase 3 consisted of the large-scale testing and data collection.

3.1. Computer Setup

Figure 1 shows a graphical description of our computer setup.

Our test-bed set-up and computer specifications were as follows:

- *Computer A:* A 300 MHz machine running OpenBSD. Its role was as a gateway to the Internet, to provide security and a source for updating the machines on our network.
- *Computer B:* A dual-processor 300 MHz machine running Mandrake Linux. Its role was as a router between the test machines “C,” “D” and “E,” which will be described in the following paragraphs. Whenever we mention the “Linux Box” throughout this document, we are referring to this machine. Network connections between this machine and C,D, and E are 100 Mbps. The connection to the gateway and the WPI network are 10Mbps.
- *Computer C:* A Pentium II 350 MHz machine running Windows 98. It had 256MB memory, a 64MB Geforce2 Ti graphics card, and was capable of running Warcraft III with frame rates above.
- *Computer D:* A Pentium II 350 MHz machine running Windows 98. It had 256MB memory, a 64MB Geforce2 Ti graphics card, and was capable of running Warcraft III without frame rate issues.
- *Computer E:* A Pentium 4 1.3 GHz running Windows XP. It had 256MB of memory and a 64MB Geforce2 graphics card, and was capable of running Warcraft III without frame rate issues.

The recommended specifications for Warcraft III are a 400 MHz Pentium II or equivalent, 128 MB of RAM, and an 8 MB 3D video card (TNT, i810, Voodoo 3, Rage 128 equivalent or better) with DirectX® 8.1 support. Although two of our machines were 350MHz machines, we felt that the graphics cards and memory that they contained made up for their deficiencies. All application level testing was performed using version 1.04 of Warcraft III, and the network traces

used 1.05 due to the requirement that Battle.net only accepts the latest version of the game.

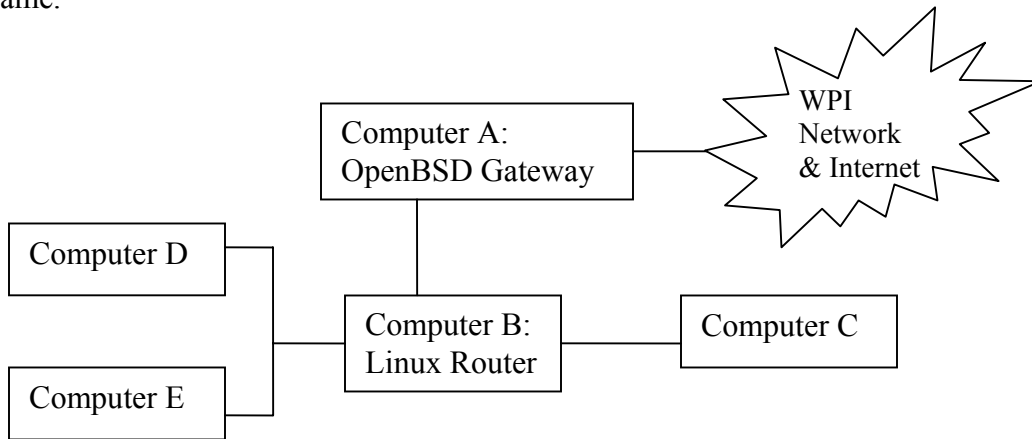


Figure 1: Test-bed setup.

A previous research group set up the OpenBSD and Linux systems in order to provide a controlled local network with Internet access for research purposes [BCMY02]. As we had this same need, we simply adapted the Linux system to our needs by installing DHCP and NISTNet, which provided us with the ability to control latency.

For our test machines, the two 350MHz systems, we first upgraded to the video cards and memory listed above to make sure that they would be able to run Warcraft III without any problems. We used a program called FRAPS that allowed us determine frame rates [FRAPS]. We felt that a minimum value of 30 frames per second was acceptable, and both 350MHz machines gave us rates between 50 and 70 frames per second during game play. We settled on Windows 98 as an operating system because it would run on the system without problems and has been on the market for several years, ensuring compatibility.

3.2. Pilot Studies

The second phase consisted of using small test maps to run various pilot studies to determine how Warcraft III worked. The first round of tests were relatively short, in order to allow us to focus our studies on areas of the game that appeared to be the most important. (Refer to Appendix A for an overview of Warcraft 3, and an explanation of the game elements used in this description of Phase 2.)

Our first test consisted of a two-player game where each player had town halls and gold mines placed equally far away from their town hall. We used map triggers to limit the game length to ten minutes, as well as to send all peasants to mine gold at the immediate start of the game. With all of these functions built into the map, it allowed us to repeat the test reliably every time. We repeated the test at several different levels of latency and jitter to get preliminary results on their effects on repeated tasks within the game. Contrary to our expectations, we discovered that both players gained gold at exactly the same rate no matter what the latency was set to.

The second test consisted of two identical units set to attack each other while doing one point of damage per hit. We again ran this test at several different levels of latency to see if there was any effect on damage dealt over time. Again, we found that there was no correlation between latency and non-interactive tasks. We also performed trials at various levels of latency and jitter to see the effect on battles and for full games.

Later in the experimentation, we observed that the Paladin's resurrect ability, and the Arch mage's "Summon Elemental" ability, added unnecessary variability to the battle by changing the number of units, and how efficiently a player used these abilities in the battle affected the outcome *very* strongly.

3.3. Experimentation

The pilot studies focused our attention on a different line of investigation than we had originally intended to pursue. The pilot studies showed that Warcraft III uses a command queuing model of some sort. In the case of resource collection, we found that once a command is given, latency does not affect how fast units gather resources. The latent computer kept the same state as the host computer no matter the latency. The effects were similar with combat. Once a unit receives an attack command, the game will carry out the order without any further communication between the client and the host. This behavior means that network problems only affect the initial command and nothing after that.

Our original methodology was to play a series of full multiplayer games under controlled network conditions to see how it would affect the outcome of the games. After completing some of the pilot studies, we decided to run smaller trials that would break

the game down into its component parts, allowing us to observe the effects of latency more easily.

We broke the game down into the three main components of an RTS game: building, exploring, and combat. We built maps to isolate each case with a limited number of players. Each map required at least two computers, one to serve as the host, and the second as a client machine. Latency was added to the network connection between the two machines during our tests, and since the host does not need to wait for a response for each command, this only affected the client machine. For the building map we were able to control the latency for each player individually, allowing us to run one test with multiple latencies.

Before we begin discussing the trials we ran, a brief discussion of the relative abilities of our test players is in order. Both players 1 and 3 are experienced gamers with considerable time spent playing RTS games. Player 1 plays a wide variety of game genres and a wide range of games within each genre. Player 3 spends more time playing RTS games in general, and the most experience with Warcraft III. Player 2 has some game experience, but most of this is concentrated in FPS, strategy, and simulation games.

The first map we tested was the building map. We divided the map into quarters using mountain ranges that units cannot cross. Each player started with a castle and four peons. Each player had unlimited gold and lumber available, and was charged with researching, building, and upgrading the complete human technology tree as fast as possible. We added triggers to the map that disabled the player's ability to build more than one building, to provide consistency and reduce confusion, as well as a trigger to display the total time since the beginning of the game. We played the map with a variety of people at a range of latencies from 0ms to 1500ms. Please see appendix D for a detailed description of our building map.

The second map we tested was the exploration map. The design consisted of a raised path which kept the player on the general course. The path twisted and turned, and had waypoints placed along the path. The player had to guide a unit through the path and step on each waypoint. Map triggers kept track of the player's time to complete the map. This map only allowed one player to run the trial at a time; the "first player" had no way to interact with the game, leaving only the second, client player, to control the unit. We

again ran the trial with various players at a range of latencies: 100, 150, 200, 250, 300, 400, 500, 600, 800, 1000, and 1500ms. Please see appendix C for a detailed description of our exploration map.

The last part of the game proved to be the most difficult to evaluate. Our original combat map was a small arena in which each player controlled a small force of units that consisted of the three human heroes, along with some lesser units. We played this map with different combatants at different latencies so we could analyze the effects of latency on who would win. Comparing the same combatants at different latencies allowed us to reduce the effect of player skill on the outcome. The selection of units that we used for this trial proved troublesome because they introduced so much variation into the results, due to a variety of reasons, that the results were very difficult to interpret.

Due to this, we designed a map with a more carefully controlled environment in order to help us see the trends due to latency. This additional battle map removed the two wildly variable heroes, and replaced them with standard units, leaving us with only one hero, a Mountain King, to work with. Please see appendix A and E for a complete description of both battle maps.

With both battle maps, we recorded multiple variables reported by Warcraft III at the end of each game in order to observe how latency affected the outcome. Our results can be found in section 4.1.

3.4. Analytical Methods

For both the building and exploration maps we recorded the game length. For the combat maps, in addition to the game length, we recorded each player's unit score, number of units killed and number of heroes killed. The number of units a player starts with plus the number of units his forces kills determines his unit score. Various units are worth various points. For a breakdown of point values refer to appendix A.

4. Analysis

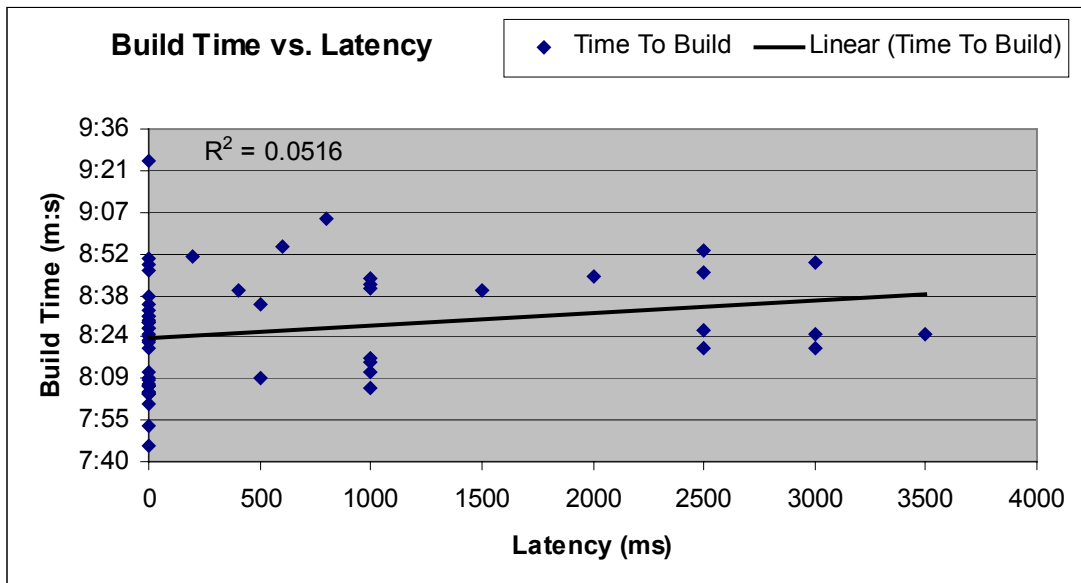
This chapter contains the analysis of the data that we collected at the application level as well as the network level. Our data is presented below in the form of annotated graphs.

Section 4.1 contains our analysis of the application level data we collected from Warcraft III. The result from each of our test maps is discussed in turn, starting with building, then exploring and battle.

Section 4.2 relates the results found during our analysis of Warcraft III network traffic. First we studied two games played over Blizzard's Battle.net, with two and four players, and then we compared them to a two player game played over the LAN.

4.1. Application Layer Analysis

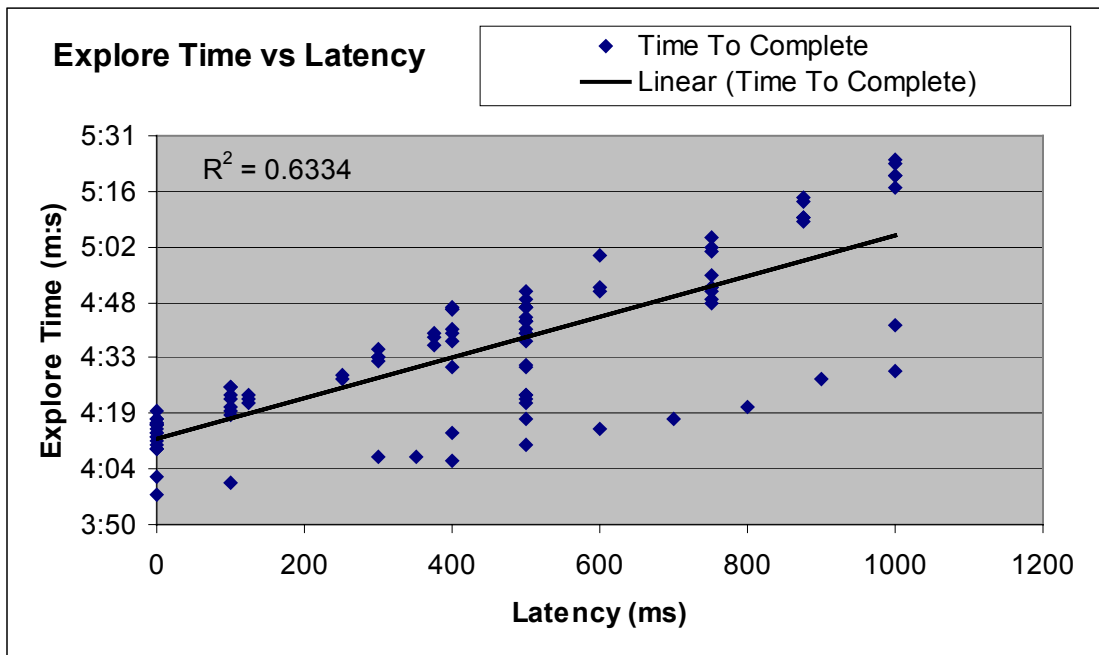
This section analyzes the data collected during the testing of the build, explore and battle maps. We first discuss the results of the build and explore maps, which had very similar results. Then, section 4.1.1 analyzes the results of the first battle map, while section 4.1.2 discusses in depth the results of the second battle map and compares and contrasts them to each other.



Graph 1: This graph illustrates the effect of latency on the total time required to build every building and research every upgrade for the human race in Warcraft III.

Graph 1 shows build time versus latency. Observing latency ranging from 0 to 3.5 seconds, the total effect on the player's required time to build the technology tree is only 14 seconds, less than 1% of a short game. Even in this concentrated experiment, over the course of only 8 minutes there is a coefficient of determination² of approximately 0.05. This low value shows that there was very little statistical correlation between latency and the time required to build the technology tree.

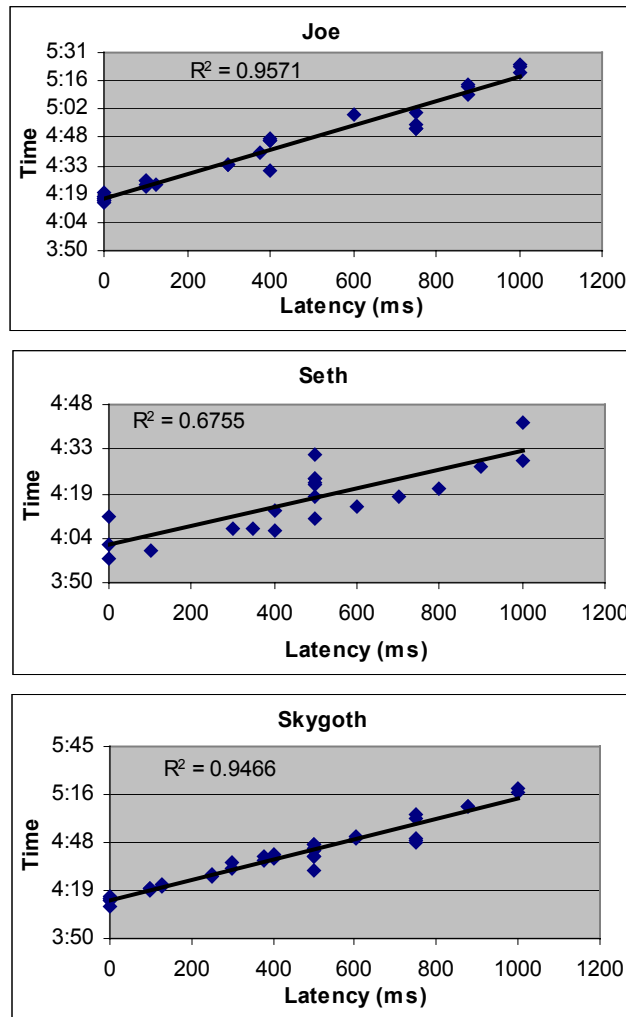
With such a low correlation in our constrained games, the correlation in a real game environment is likely to be even lower. This is due in part to a longer game time, and to the fact that different numbers of buildings would be produced (such as more than one farm). In a real game, it is improbable that a player would build the entire technology tree. In addition, in a real game, players may build their towns in a strategic layout instead of in a random pattern. Finally, time is often spent in a real game attending to other matters so the speed of building the base is not of utmost importance. Our conclusion is that any effect latency may have on building would have no significant impact on the outcome of real Warcraft III games.



Graph 2: This graph illustrates the effect of latency on the exploration of a map.

² Please see the glossary for definition.

Graph 2 shows explore time versus latency. The correlation between latency and time to explore is strong for individual users, all of whom had correlation coefficients of 0.6 or higher, as depicted by graphs 3-5 below. The first 8-10 games often show a vertical component within a single person's data, which we attribute to a learning curve. This learning curve is simply the player getting comfortable with the map, such as when exactly a waypoint is triggered and the movement speed of the unit. Once the map is known, all data shows nearly perfect linear relationships between latency and time to complete.

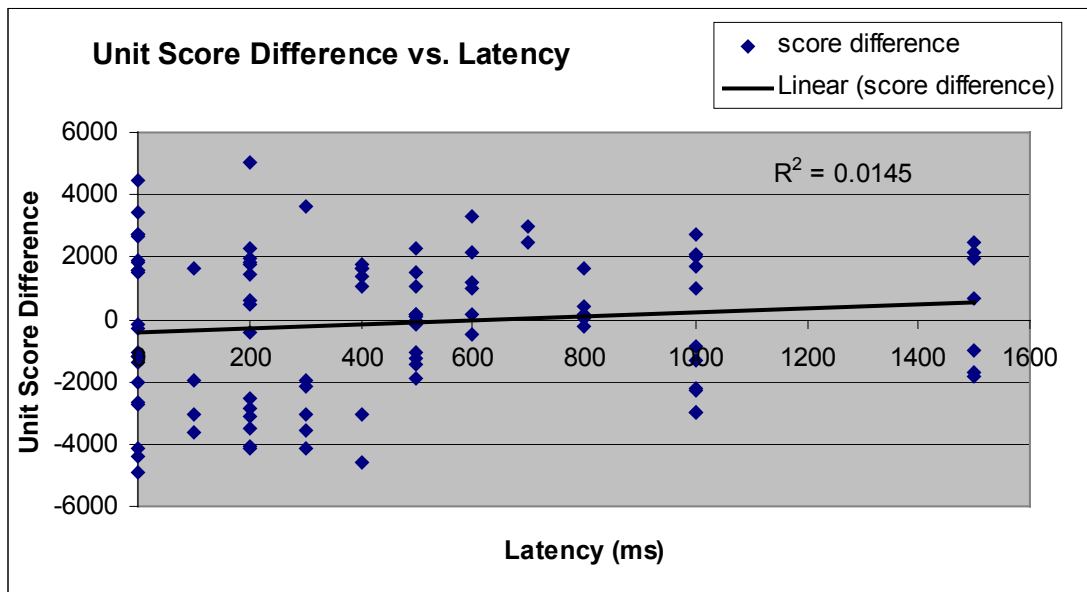


Graphs 3-5: Individual players explore times versus latency

Since most games are played on a relatively small set of standardized maps, they are quickly memorized, eliminating the need for actual exploration. Much like in the

case of building, our conclusion is that the 6 additional seconds of exploration time added for every 100ms of latency would be insignificant over the course of most Warcraft III games. In addition, latent players in a real game may figure that spending less time controlling their units for exploration yields better results and instead of micromanaging them they may send them a long distance with each move command.

4.1.1. Battle Map 1



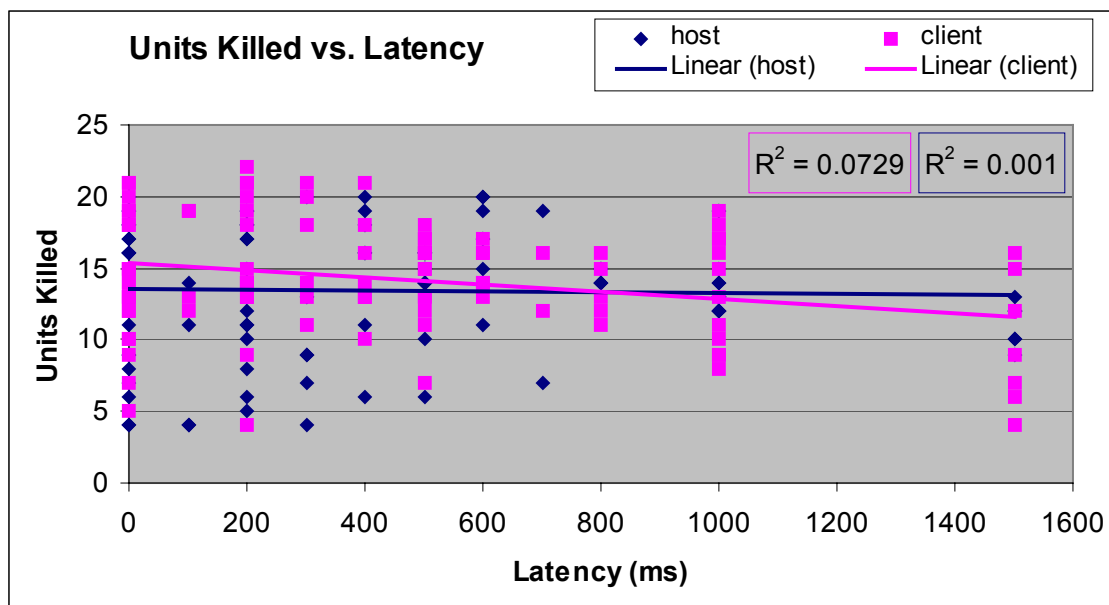
Graph 6: This graph illustrates the difference between the latent player's score and the host player's score, as reported by Warcraft III. Negative scores mean that the latent player had a higher score.

Graph 6 depicts the relationship between latency and the difference between the latent player's score and the host player's score. As the latency increases, there is a noticeable decrease in the variability of the score difference at any given latency, which can be seen in Table 1. We feel that this was due to the latency limiting the latent player's ability to adapt during the battle, and allowing the other player free reign to adjust his strategy and play as needed to win consistently.

Latency	Mean	Standard Deviation
0	-52.27	2679.65
100	136.00	1675.31
200	-356.67	2844.09
300	-75.71	2404.41
400	-17.14	2642.52
500	12.31	1242.74
600	861.67	1234.50
700	953.57	1128.04
800	647.14	905.95
1000	52.50	2181.75
1500	258.75	1781.25

Table 1: Mean and Standard Deviation versus Latency

There is also a slight upward trend in this data, but due to the large amount of variation, it is statistically insignificant. The slight upward trend is the result of the killing of two minor units that became harder to protect as latency increased. With the large range of variability in the data, the death of two units either way is not surprising.

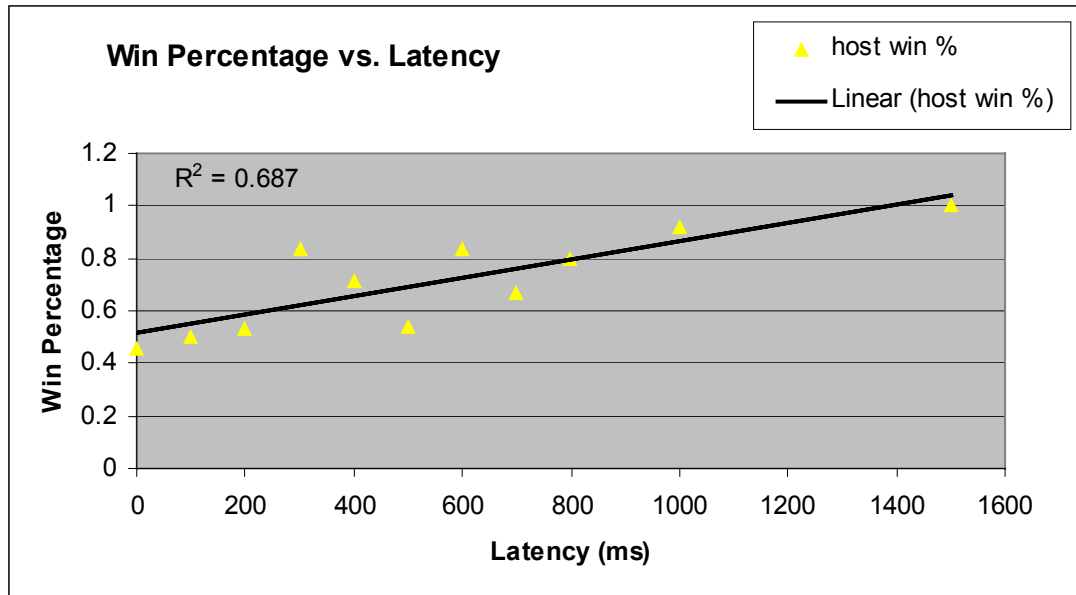


Graph 7: This graph displays the number of units killed by each player, and a trend line for each. Clearly the client does worse, while the host stays roughly the same.

Graph 7 shows how the number of units killed relates to latency. The number of units killed and the unit score are directly related, as each unit has a fixed point value based on their power. Here we can see that the host's number of units killed stays the same, while the latent player's number of units killed declines. Again in this graph as latency increases, the randomness in the data declines. There are many possible ways to explain the stability of the host's kills vs. the instability of the latent player's kills. In particular, the host's strategies are unlikely to change at all during the battles, as his system is never latent. The latent player may need to change his strategy to adapt to his lack of responsiveness. On the other hand, the latent player's game will experience a time delay between instructions. This allows his units to attack whatever unit is closest. One of the first strategies players learn is to focus damage on one unit. Killing one unit quickly means that there is less overall damage being done to your units. Therefore, allowing the computer to choose which units to attack can be an inefficient way of doing damage, especially if a large number of units spend time attacking a unit that has high hit points and does low damage. In a real game, it is likely that the latent player would quickly give consecutive *queued*-commands in order to compensate for the effects of latency.

As in graph 6, there is a lot of variability in graph 7. Each player was given 12 units to start with. However, in one game in particular, 23 units were killed by one side. This is due to the use of the paladin's resurrection spell which allows a player to resurrect up to 6 dead units. If only one player casts the spell, they are more likely to win decisively, preventing the opponent from killing as many of their units. If both players successfully cast the spell, then both players will kill more units than normal, even though the difference might not be significantly higher. In addition to simply missing a chance to resurrect, there is also the variability of resurrecting 6 of the least useful units or casting the spell outside of the area where the units died, resulting in fewer or no units being resurrected. As latency increases it is less likely that the latent player will utilize the full potential of the resurrect spell, leading to a decrease in the number of units in the battle. The archmage and the paladin are two units that had very time sensitive spells that would sway the outcome of the test battle if a player did not use them to full effect. This was particularly true of the paladin's resurrect spell. Thus, our data tended to be unfairly

balanced towards these units' timing-important maneuvers. In a full game, battles are likely to be fought with more units, especially by the time a hero has reached level 6. We chose 12 units for testing purposes, as that is the maximum number of units that could be selected at once.



Graph 8: This graph shows the host's win percentage as latency increases, out of 146 games.

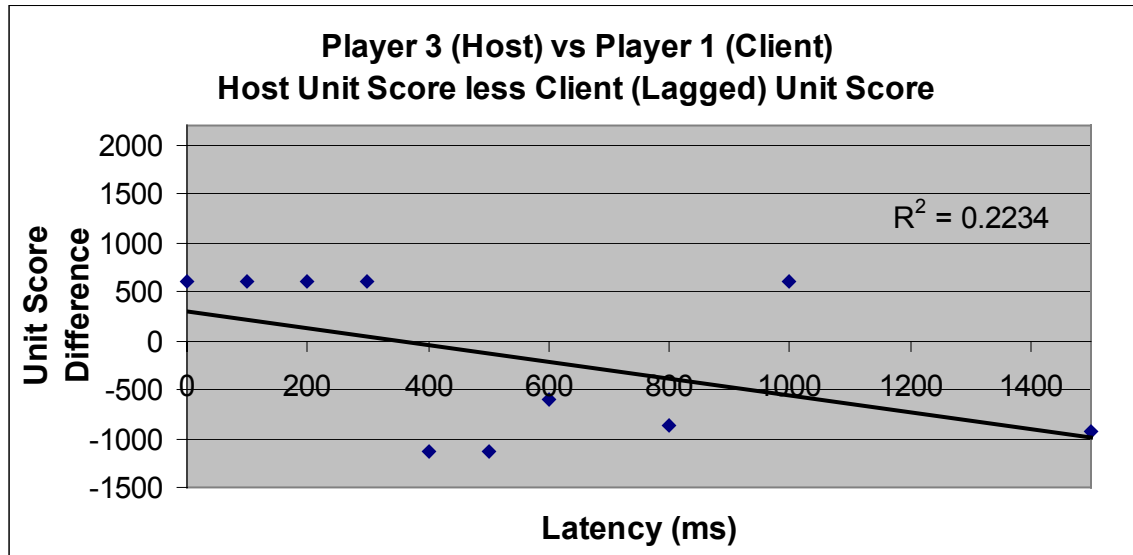
While this last graph from our first battle map does show a clear upward trend with a reasonably good line of fit. However, we felt that this started to push the limits of interpretation for our data, as it failed to show some significant details. The win percentage fails to present the manner of the victory, such as whether the victor had one unit left or four. This, in addition to our analyses of the other results, led us to our second round of battle research.

Graph 8 clearly shows a strong correlation between latency and the percentage of wins. However, we feel that this does not indicate an overall correlation between latency and performance under real world conditions. This is due to the effect the paladin has in controlling the outcome of these battles. In a real game there is less reliance on any one hero or unit, as there is usually a strategy available to counter the advantages of any individual unit. In this, the game is well balanced. This leads to developing strategies

based on a variety of units where each serves a specific purpose, thus negating any latency effects. This could not be captured in our trials.

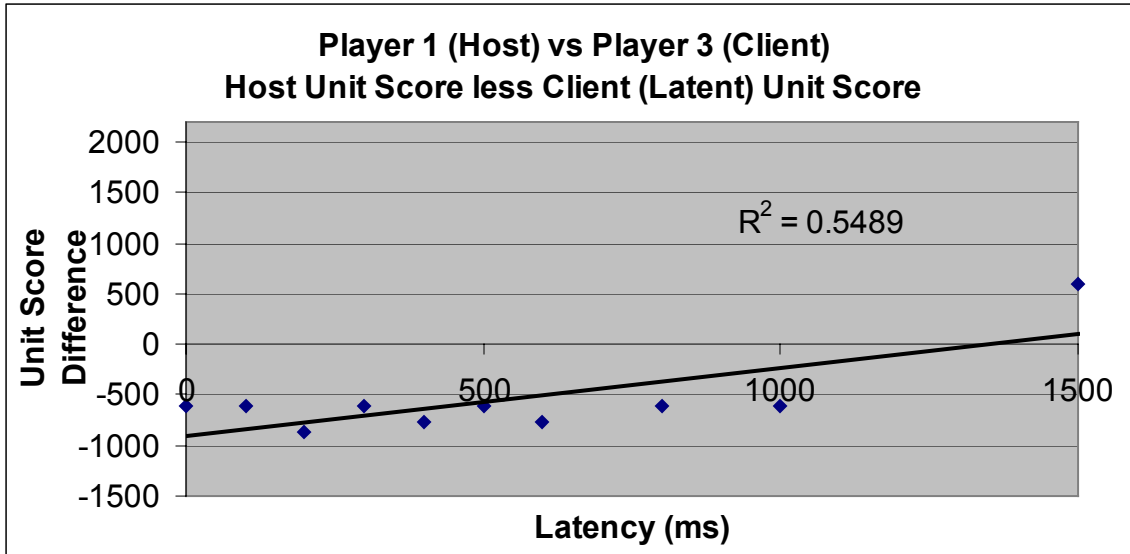
4.1.2. Battle Map 2

In map 2, we removed the paladin and Archmage from the battle, leaving the dwarf as the only available hero. The highest possible score for this map is 6040, while the lowest is 3020, because the unit score is the value of your units plus the value of units you kill.



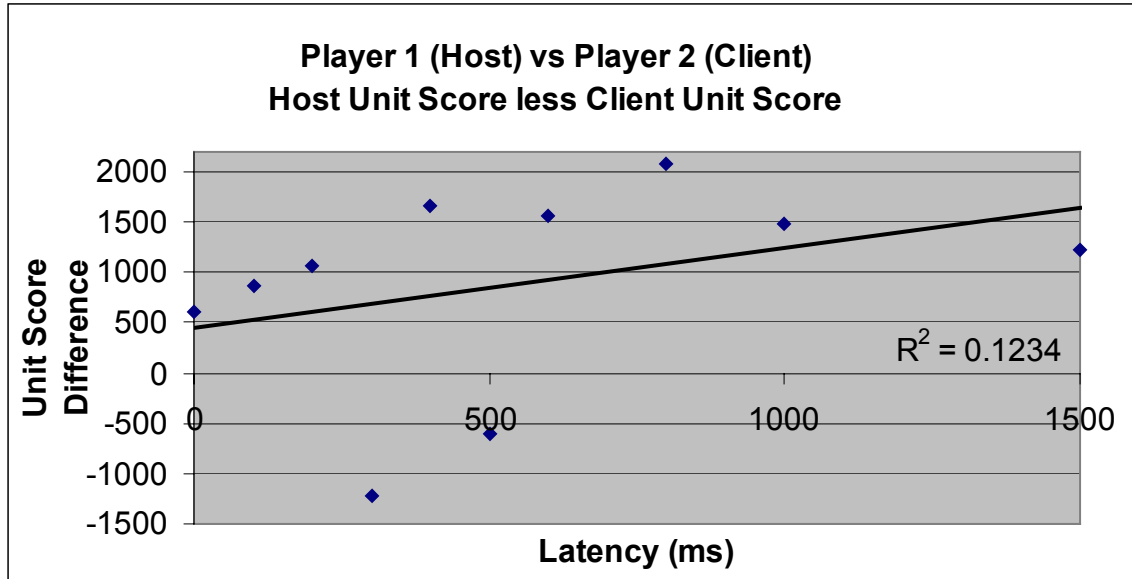
Graph 9: The difference in score versus latency for Player 3 as the host and Player 1 as the client. With a 2-point moving average.

As graph 9 shows for the battles between “Player 3 versus Player 1,” there is almost no correlation between latency and score difference in our second trial, and there are no discernable trends. Results such as these, with every round ending in almost precisely the same way with a variance of 1 or sometimes 2 units either way, showed up in every pairing of players, meaning that the fights had similar endings, regardless of latency.



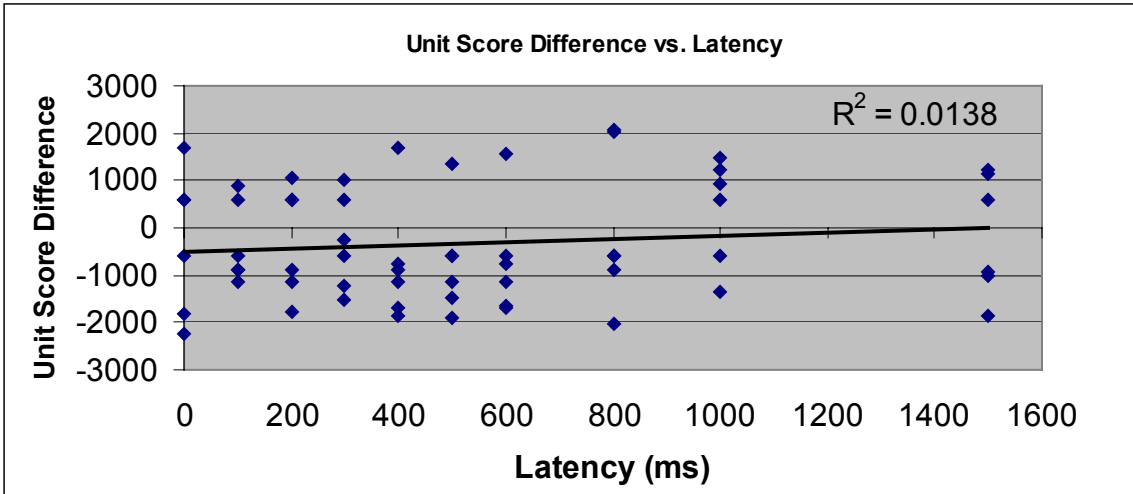
Graph 10: The difference in score graphed against latency for Player 1 as the host and Player 3 as the client. With a 2-point moving average.

Graph 10 shows that the battles with “Player 1 versus Player 3” all ended with either 1 hero standing, or 1 hero and a single extra unit. There were two exceptions in the two cases of 200 and 1500 ms latency where there were two additional units and where Player 1’s hero was left standing instead of Player 3’s, respectively. This could appear as a relationship between latency and battle efficiency, or it could simply be a random event. Taking into consideration the other two graphs, we believe this to be a random event.



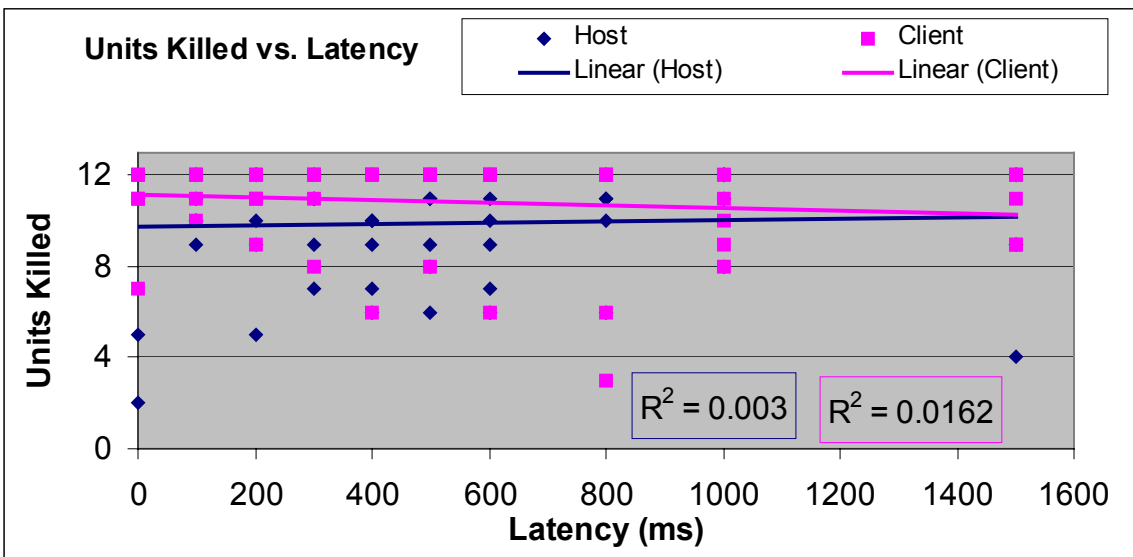
Graph 11: This graph shows the difference in score between the host and client as reported by the game with a 2-point moving average line for the data.

Finally, graph 11 shows that in “Player 1 versus Player 2,” we see much more variability than in either of the other sets of data. Player 2 was the least experienced player among the three players and so his strategy in battle was not as well organized as Player 3’s or Player 1’s. As a result, the battle ended in a larger variety of ways, ranging from a hero and a unit on the latent side to a hero and two units on the host side. However, even with this range, one can see a slight trend with nearly every battle ending with a hero and a unit left on the host side.



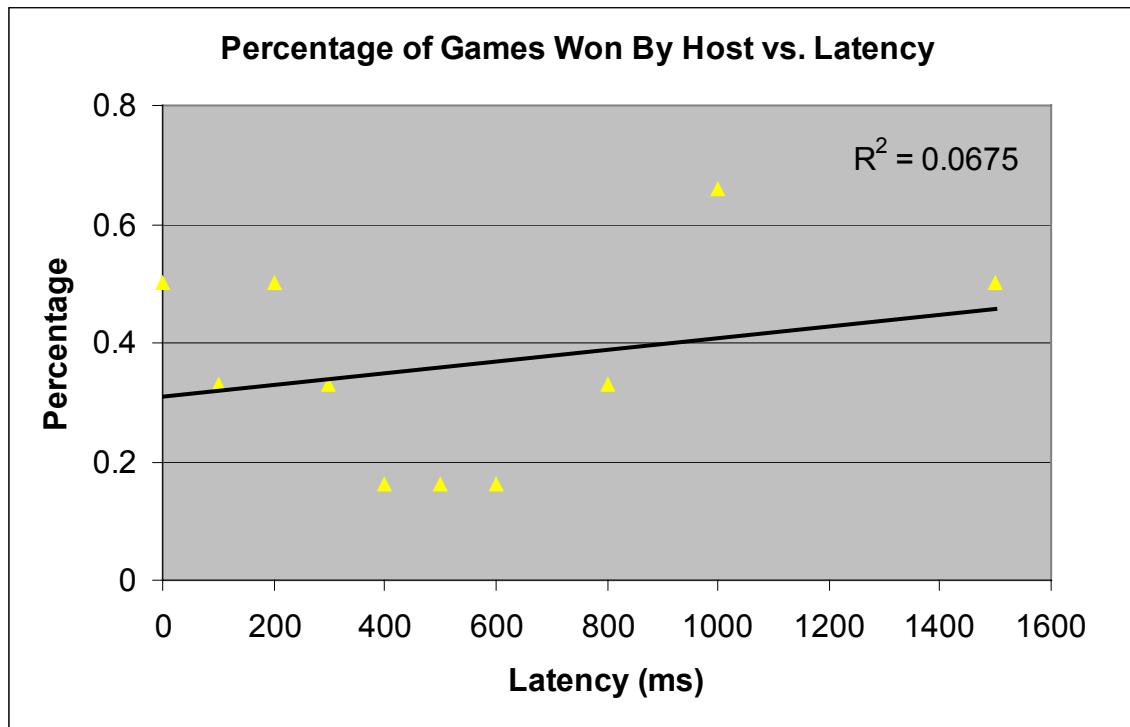
Graph 12: This graph shows the unit score difference versus latency for all trials in the second battle run, as well as a linear line of best fit for the data.

Graph 12 has the same trends as that of our first run of battle trials. Much like our first battle map, this graph shows an upward trend that is even smaller than graph 6, which ranges over the value of a single unit. Also the variability dropped by about half from the previous trial and that there is no narrowing trend as latency increases.



Graph 13: This graph shows the number of units killed by each player, and a corresponding trend line.

Again, in graph 13, we can see similar differences between this run and the earlier battles, where here there is much less variability as latency increases. The trend lines are very shallow and not statistically significant.



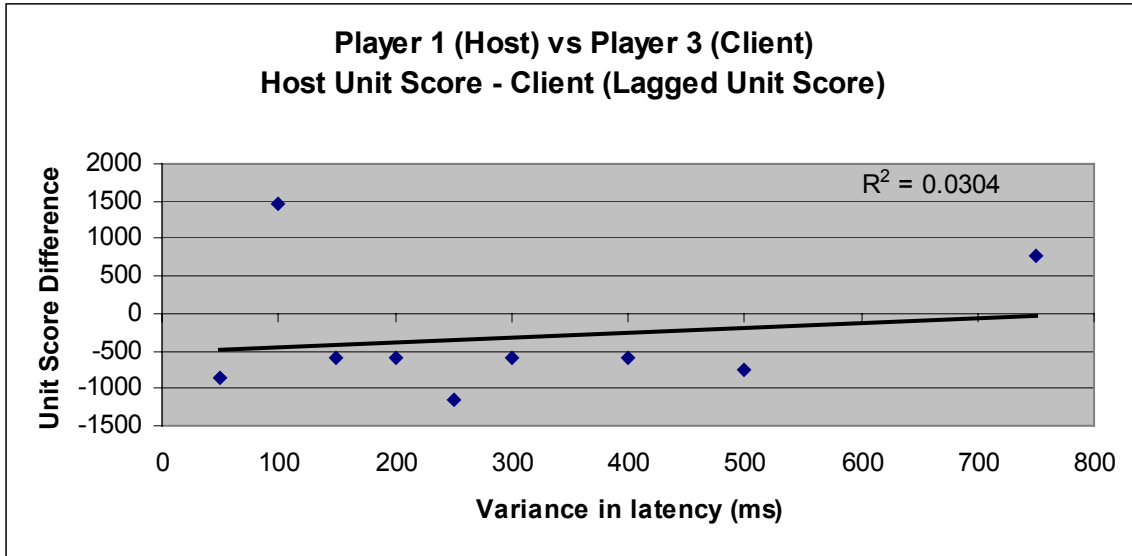
Graph 14: Percentage of games won by the host versus latency.

The only clear indicator of a correlation between latency and performance during the previous set of trials was in graph 14, win percentage versus latency. With the new data there is obviously little to no trend and this can be reinforced statistically by the much lower coefficient of correlation of .0675 versus .687 for the previous set. The trend line is weak and should not be seen as any sort of concrete result because only six percent of the change in win percentage can be attributed to a change in latency, thus leaving the majority of the change to some other factors.

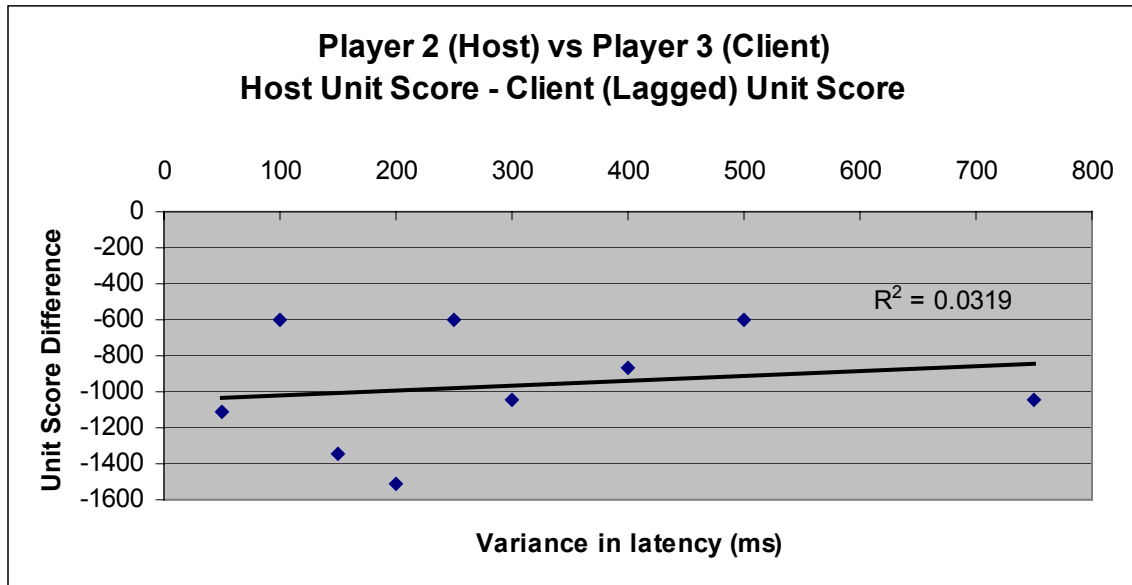
4.1.3. Jitter

Due to our observations of latency's effect on the player's performance in the game, we also examined variable latency to more closely emulate real-world network conditions

and to see if it was the change in latency, rather than latency itself that was responsible for poor game play. The latency shown in graphs 15 and 16 represents the standard deviation of a normal distribution with mean zero.



Graph 15: Unit score difference for Player 1 vs. Player 3 under variable latency, along with a 2-point moving average line.



Graph 16: Unit score difference for Player 2 vs. Player 3 under variable latency, along with a 2-point moving average line.

Here, once again we see the same variable results. Graph 15 shows Player 1 winning two games, one at 100ms, and then again at 750ms, while losing the games in between, all by similar margins of 1 or 2 units. In graph 16, Player 3 consistently beats his opponent in every game, but by varying margins. Neither graph shows a significant statistical relationship between latency and success in battle, just as we saw earlier with constant latency.

Our research has shown that, contrary to our original hypothesis, there is very little relationship between latency and game performance of Warcraft III, the Real Time Strategy (RTS) game we selected. Both as a direct conclusion from our data and with extrapolation into a complete game, we found that the effect of latency on the outcome of a Warcraft III game is negligible over a range of practical latencies.

An important aspect to our research is that the trials were done in a highly controlled environment, with limited commands and strategic options in comparison to a full online game, even a quick one. This plays some important roles in the interpretation of the data, as will be discussed later.

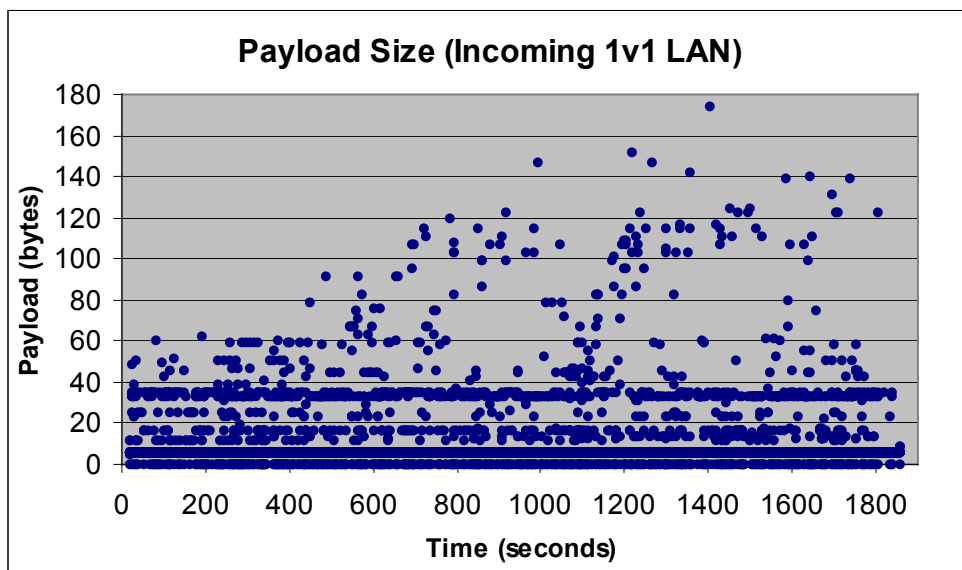
4.2. Network Analysis

Warcraft III appears to use a weak client-server architecture, in which all data is sent to a host machine, where it is combined and then redistributed to the other players in the game. Even though there is one machine acting as a host, it appears from our pilot studies that each machine maintains a complete copy of the game state, and to an extent, all outcomes are predetermined upon initiation of the action. This is backed up by the observation that data is only transferred upon the issuance of a command, and never again during the life of the event. For example, the commands to initiate a large-scale battle result in an increase from the base payload size, but the battle itself has no effect on traffic unless there are further commands being issued as the battle is carried out.

All packet traces shown in this section were captured using Ethereal. We packet traced three full games, ranging in length from approximately 20 to 30 minutes. We traced two games played through Battle.net, and one game played over a LAN. All of the games showed fairly similar overall traffic patterns that differed slightly with the number

of players and outcome of the games. Warcraft III uses TCP and uses port 6112 as the server port. All IP traces were performed on the client machines, and “incoming” and “outgoing” are appropriately labeled as such. In the case of the Battle.net games all data is sent and received from Battle.net servers that during game play act as Network Address Translation servers to protect users’ identities. Battle.net games use a separate channel to communicate directly with the Battle.net servers, which appears to be used as a ‘heartbeat’ during game play, the purpose of which we could not determine. This separate data stream was not taken into account because it featured a much larger inter-arrival time and we felt that it was insignificant enough to eliminate it from actual game traffic.

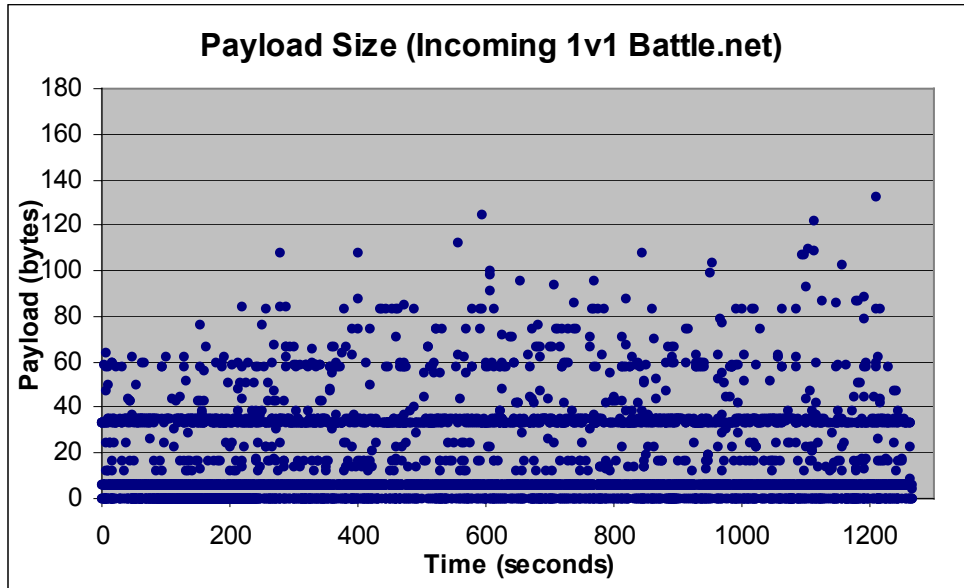
4.2.1. Incoming Data Streams



Graph 17: Payload size (incoming) for a 1vs1 LAN game.

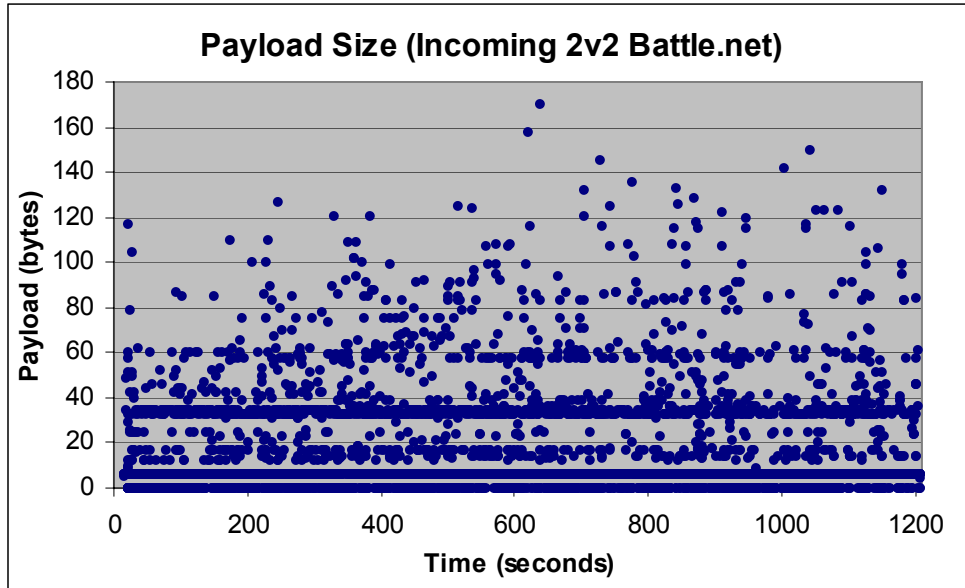
Graph 17 shows the trace of a two player “1 versus 1” game played over our local area network with no induced latency. There are three distinct bands of traffic located at 0 bytes, 6 bytes, and from 33 to 35 bytes of payload. By far the most dominant band is the one at 6 bytes of payload, which corresponds to no commands being issued in the game by any player. The 0-byte band is TCP ACK packets which were not piggybacked. The random scatter of other payload sizes we found to be the result of various commands

issued by each player, with the band at 33 to 35 bytes of payload corresponding to a particularly frequent command, or combination of commands.



Graph 18: Payload size (incoming) for a 1vs1 Battle.net game.

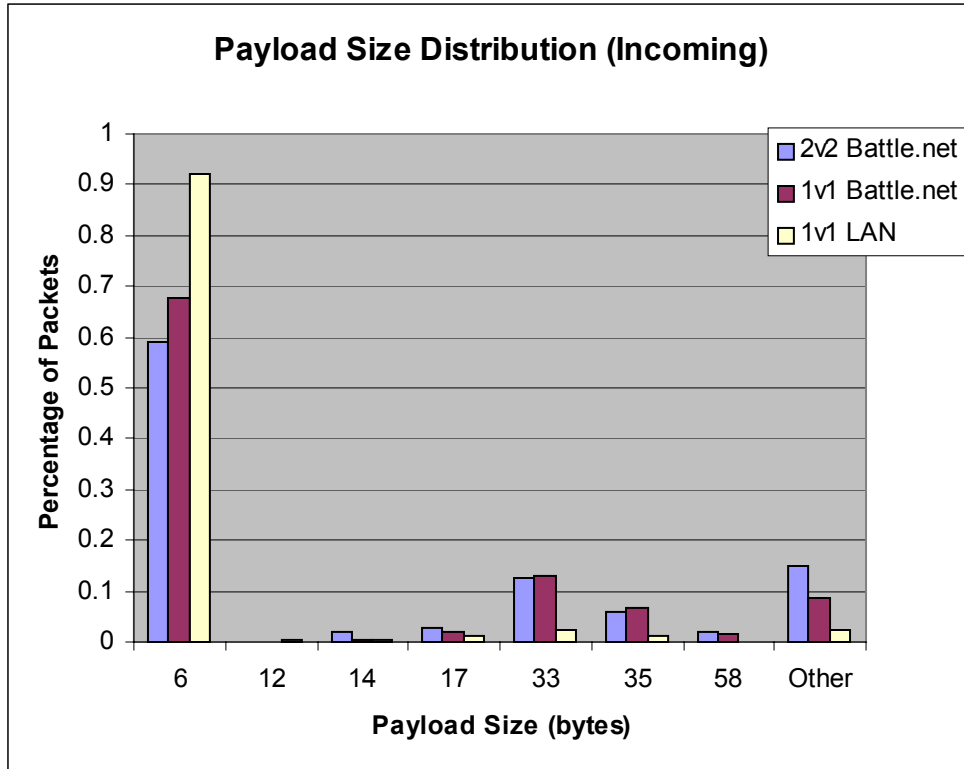
Graph 18 showed the same concentrated bands at 0, 6, and 33 to 35 bytes of data. The higher frequency of other payload sizes is the result of a faster paced game played by more experienced players, resulting in more commands being issued. The game was also much shorter.



Graph 19: Payload size (incoming) for a 2vs2 Battle.net game.

In graph 19 we see the same prominent bands of data with a still higher frequency of other sizes due to the fact that each packet from the server contains the commands of four players instead of two. This makes it more likely that a command will have been issued by one of the four players at any given time, resulting in more packets above the base payload of 6 bytes.

It is easier to see these trends in a graph of the relative frequency of each payload size as shown below. The “other” category includes many different packet sizes which arrived relatively infrequently. We believe they can be explained by combinations of commands being sent in a single packet, or rarely used commands. For example, in the “2 versus 2” game, the 7 most frequently sent payload sizes account for approximately 88% of the packets, with 109 unique payload sizes making up the remaining 12%.

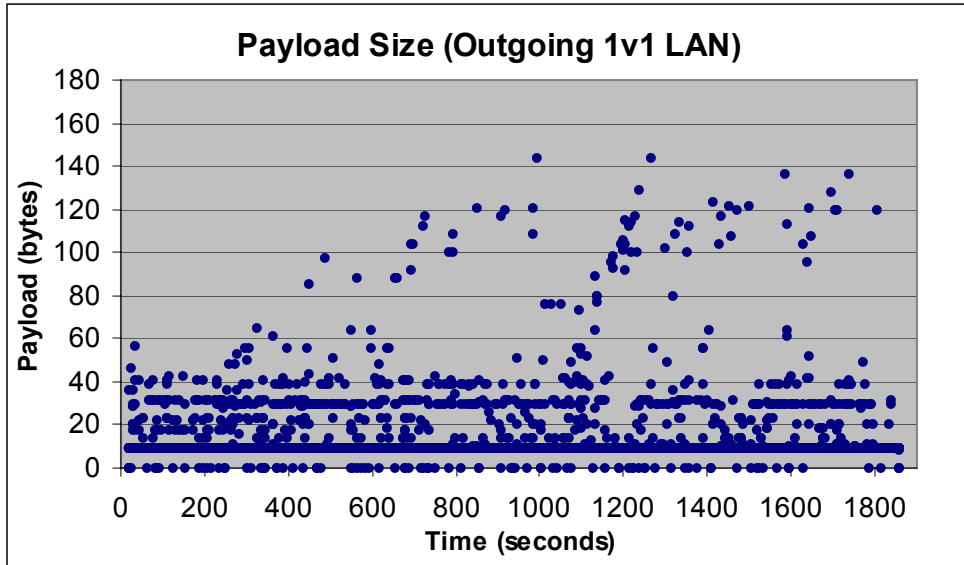


Graph 20: Percentage of Packets versus Payload Size for each game type.

It can also be seen in this graph that there is a large difference in the percentage of packets that have a 6 byte payload between the three games. These differences are accounted for by the higher percentage of packets in the other payload ranges.

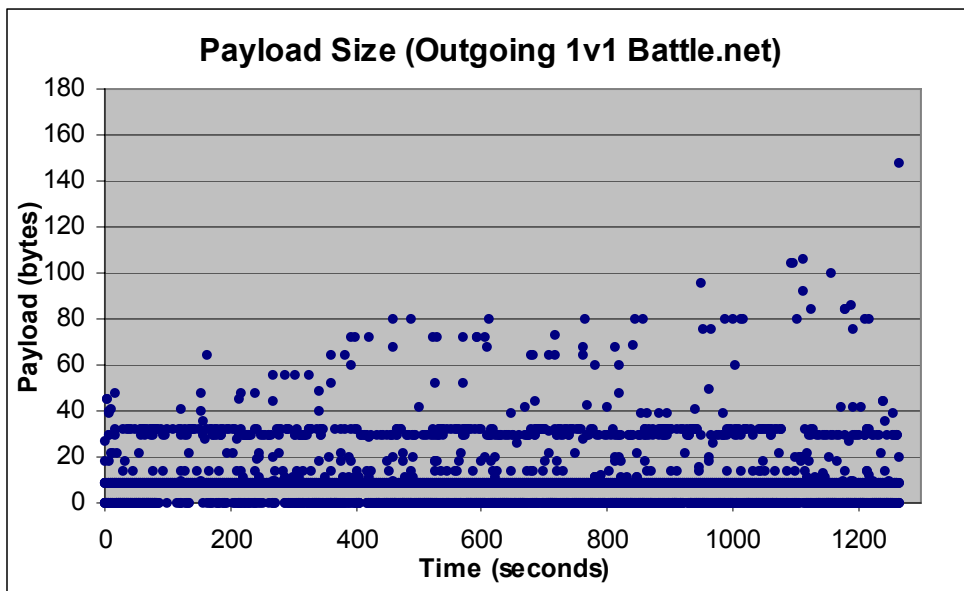
4.2.2. Outgoing Traffic Streams

The outgoing data streams conform to much the same patterns as the incoming, with a few minor differences. As was stated above, Warcraft III's client server architecture means that each player will receive all commands, but a player will only send out the commands that they issue. In the 1 versus 1 game, this means that roughly half of the commands in the incoming stream will have originated from our client and show up in the outgoing stream. This trend is somewhat visible in the graphs of the slower paced LAN game, but nearly impossible to see in the Battle.net games.



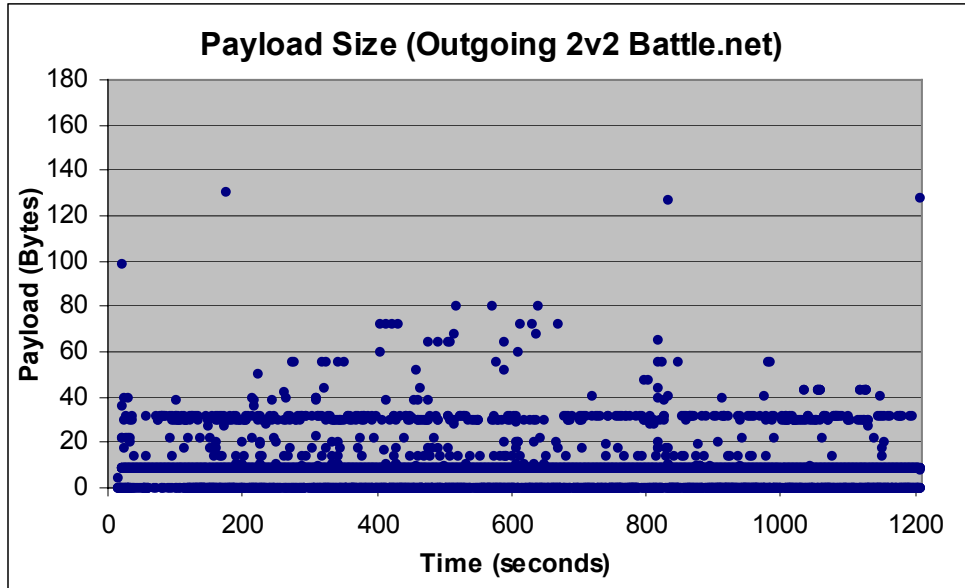
Graph 21: Payload size (outgoing) 1vs1 LAN game.

Graph 21 depicts the distribution of outgoing payload sizes during the “1 versus 1” LAN game. Another difference between the incoming and outgoing data streams are the locations and densities of the bands. In the outgoing data the base payload size is increased to 9 bytes, the higher band moves down to 30-32 bytes, and again the band at 0 bytes of payload are due to solitary ACKs.



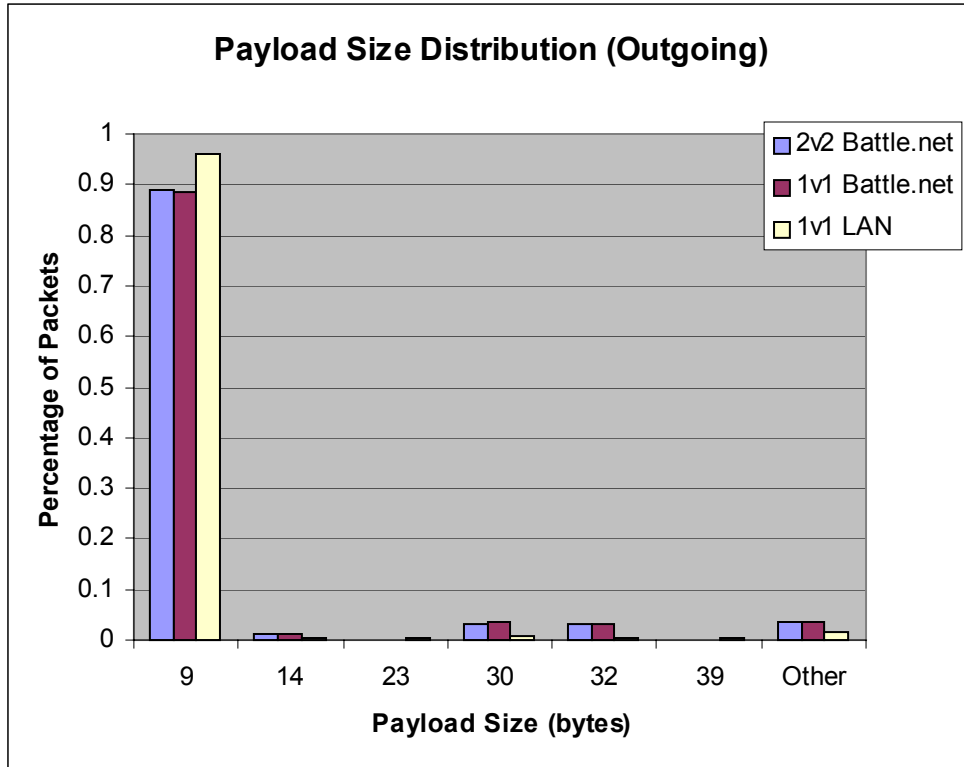
Graph 22: Payload size (outgoing) 1vs1 Battle.net game.

Graph 22 shows the 2 and 4 player Battle.net streams and we can see the higher command rate in the higher density bands of 30 and 32 bytes of payload and an increase in the number of large payloads.



Graph 23: Payload size (outgoing) 2v2 Battle.net game.

Once again, the trends are easy to see in graph 24, with a small change in the relative frequency of payload sizes. The base payload at 9 bytes accounts for a larger percentage of the stream, with approximately 5 to 10 percent more packets at that size than the incoming streams. This increase in the 6 byte packets is mirrored in the lower percentages of larger packet sizes.



Graph 24 Payload size distribution (outgoing)

4.2.3. Packet Timing

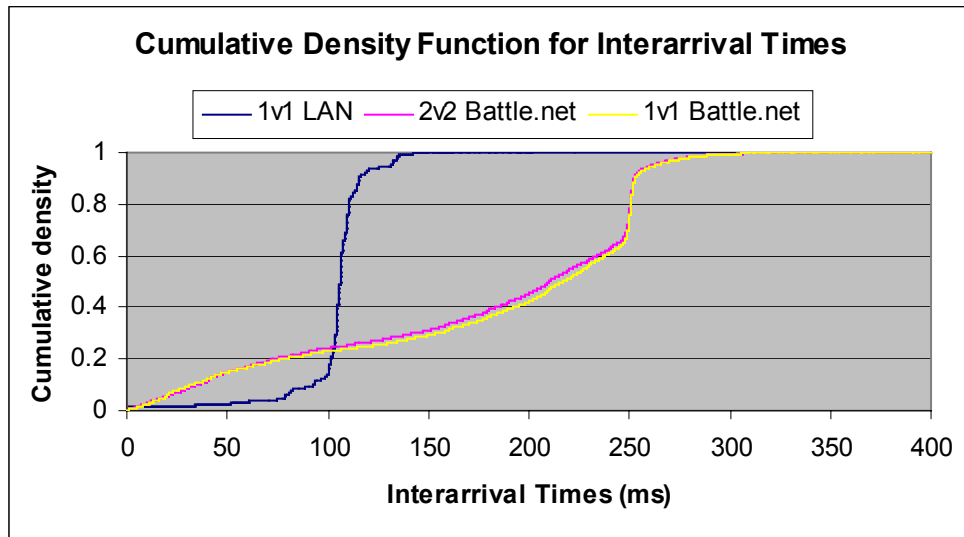
One of the features of Warcraft III's network architecture is that it sends out packets at set intervals. Table 2 depicts the intervals that we observed for incoming and outgoing inter-arrival times during the games we traced. In our local area network game, Warcraft maintained a very steady inter-arrival rate of one packet approximately every $1/10^{\text{th}}$ of a second both incoming and outgoing. With our Battle.net games the timing interval was slower, down to one packet every 200 ms incoming and every 160 ms outgoing.

	1v1 LAN	1v1 Battle.net	2v2 Battle.net
Incoming Mean	0.1041	0.2013	0.2009
Incoming Standard Deviation	0.0186	0.0791	0.0781
Outgoing Mean	0.1038	0.1647	0.1593
Outgoing Standard Deviation	0.0194	0.0874	0.0882

Table 2: Mean and Standard Deviations of Inter-arrival Times in seconds

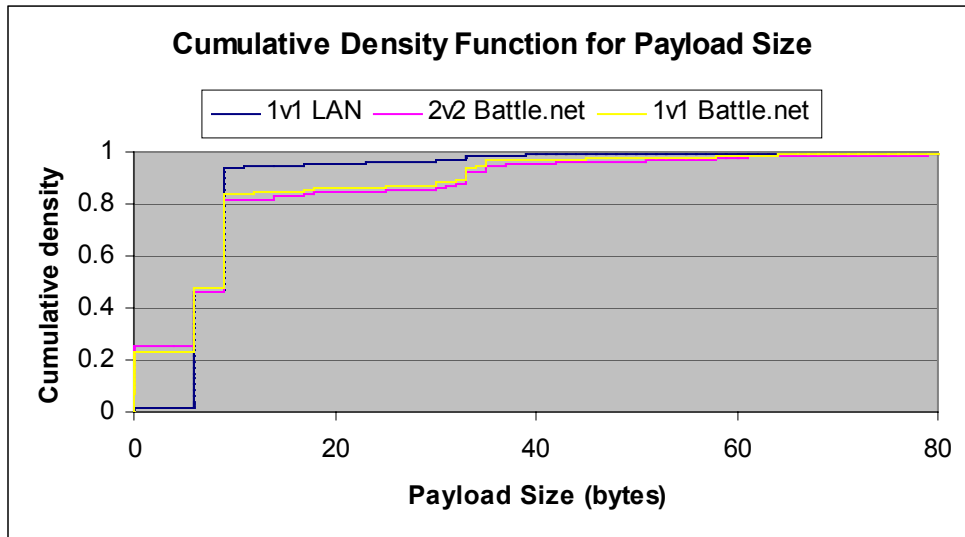
4.2.4. Comparisons

To make a final comparison between the different games that we traced, and to compare and contrast our result with other network games, we used cumulative density functions to summarize the points made earlier in this section.



Graph 25: CDF of Inter-arrival Times for all three games

In graph 25 we can clearly see the difference between the consistent LAN connection and the less consistent Internet connections, as the LAN game has a much steeper curve because the inter-arrival times are centered more closely around the mean value. We also noticed the heavier tail that you would expect with Internet traffic, with a small number of packets arriving much later than the rest.



Graph 26: CDF of Payload size for all three games

With the cumulative density functions shown in graph 26, you can see the difference in the numbers of ACK only packets sent in the Internet games in comparison to the LAN game. We can also clearly see all three games with large vertical sections that correspond to the 6 and 9 byte payloads that make up the majority of the traffic, and the much smaller numbers of larger payloads that correspond to commands being issued.

Comparing our network traces to traces taken of StarCraft games, which is another Blizzard RTS, we see a substantial difference in the packet sizes sent by each game. The two most common packet sizes in Starcraft are 122 and 132 byte packets, while Warcraft III packets are most commonly 60 or 63 bytes in size [DLJW01].

4.3. User Analysis:

This section is solely based on observed player reactions, and we do not provide an analytical way to quantify player perceptions. Latency does not necessarily affect game play, but *perceived* latency, whether real or imaginary, does affect the gaming experience. The point at which latency begins to be noticed can vary between individuals.

As the games were played, players observed that it was relatively easy to adjust their strategy to compensate for levels of latency between 0ms and 500ms. At levels of low constant latency such as these, the game still ran smoothly. Although the delays in

executing commands were perceptible, it was relatively easy to estimate this delay and react accordingly to account for it.

At levels of fixed latency above 800ms, the game started to appear erratic, and this made for a poor game experience. Without a smooth flow of information, it was difficult to try to implement a strategy. None-the-less, the statistical results showed that the strategies were not rendered totally ineffective. As mentioned above, this only became noticeable at much higher latencies, and the exact point at which a player perceived a poor game experience varied from person to person.

In dealing with variable latency, the game experience was usually perceived as acceptable, due in large part to the difficulty of determining any change in the connection. Players had trouble determining if a latency spike was in fact a latency spike or if it was something else like a decrease in the frame rate. The only difference between levels of fixed latency below 800ms and variable latency was the noticeable pauses that occurred when units at a standstill were given move commands. The delay between issuing the command and the unit moving was approximately the same as the latency.

Player strategy and skill level are two of the factors that contribute to how noticeable latency was during game play. A strategy that relied heavily on micromanagement suffered the effects of, and was more able to clearly point out latency than a strategy that was less focused on individual unit control. What the player chooses to micromanage also has an effect on how perceived latency affects his gaming experience. A player whose strategy involves micromanaging his town rather than his units is much less likely to be aggravated by latency than a player who micromanages his units in battle. Also, a mistake during a battle that appears to be the result of latency will be much harder to forget about than a mistake during building such as canceling a building.

From all the observations that we have made, it has been the user's perceptions of the game that are most effected by latency. In many cases, it is because they feel that they are doing poorly as a result of the latency, due to a single noticeable mistake, or a mistimed maneuver. However, it is rare that small mistakes result in a direct loss, as supported by data in section 4.1.

5. Conclusions:

The field of network games is a growing area of computer science that has largely been ignored by researchers until recently. Results obtained from research in this field can benefit game makers by providing knowledge of what conditions their games must work under, as well as network designers, by providing information about what types of traffic their networks will carry. We set out to extend previous network game research with a study of how latency affects the playability of Warcraft III, with the prospect of extending our findings to real-time strategy games in general.

Our primary focus in this project was to research the affect of latency on player performance, and to have our data available to compare to other similar studies of other games. We decided to split Warcraft III into three components of play: building, exploring, and combat. Our secondary area of focus was to characterize the network traffic generated by Warcraft III, to provide data for interested parties to simulate or emulate the effects of the game on a live network.

At the application level, we find that latency appears to have little effect on the final outcome of building, exploring, and most battles. While we did see some effect of latency on the metrics used to measure performance in our experiments, the effects observed would be insignificant during a regular game. For instance, in the equations for our lines of fit for building and exploring, the slope term was on the order of 1 microsecond and 10 microseconds, respectively. Even at high latencies of up to 800ms, this would result in an incredibly small effect, even in a short game.

In our battle experiments, we found that in a fair fight, the effectiveness of certain units is influenced by latency more than others, because they have an element of timing that is not present in a majority of the units of the game. We found that the effectiveness of certain strategies that involve precise timing of events are more influenced by the amount of latency, but that very few such strategies prevail in typical Warcraft III games. Generally, only the most advanced players use such strategies, as they require relatively precise knowledge of the game.

On the other hand, most battles in Warcraft III are far from fair, with one side outnumbering the other, or higher technology levels on one side, or some other previously implemented strategic advantage that makes even the aforementioned tactical

battle strategy somewhat irrelevant. In this case, we feel that the only practical effect of latency is to limit the ability of a player to adapt his/her strategy as a direct response to the strategies of their opponent.

Overall, we feel that strategy plays a much larger role in determining the winner of a game than does latency. This is because Warcraft III is designed to allow players to plan out their strategy, and implement it. While Warcraft III is in real-time, quick movements and superb reaction times only play a very small role when compared to understanding the game, knowing a map, and having good strategies. Having the capabilities to produce powerful units, a well protected base, and a good battle strategy will do much more to determine victory than a fast, close connection.

In our network analysis of Warcraft III, we discovered an interesting game mechanism that may help mitigate the effects of latency. Warcraft III regulates the action based on the relative locations of items in the game. This means that players issue commands to attack another unit and not a location specifically. And furthermore, in the case where players choose to attack a location, the command is usually interpreted as “attack enemies near this location.” A single consistent state across all players is not required, as the unit should still exist in all clients, and all that is required is for the unit to close with the enemy and attack. This mechanism is supported by data in the network traces.

Warcraft III sends the commands out as they are issued, but it does not send any data about damage and position during the game. For example, if a player starts a major battle, the only network traffic will be the commands, so if no commands are issued, then only base size packets are sent. Thus, the outcome of a command is essentially predetermined, and each client uses this to calculate an identical copy of the game state. These identical copies are not necessarily kept in perfect synchronization with each other, but that at any given moment in game time, which might not be the same moment in real time, all the states are identical. We also saw indications of this during our application level testing. For example, a battle with 1500ms of latency would end on the host approximately 1.5 seconds before it ended on the client, but the actions carried out on each screen were identical. One possible system for implementing this is sending a

random number seed along with the command or creation of a unit, and this seed is used in each state to reach the same outcome.

Overall we feel that most Real Time Strategy games, if implemented correctly and balanced well, would behave in a similar manner while dealing with latency. We feel this is the case because the game mechanics that make up Warcraft III are very similar to those of most RTS games. One of the ways in which we believe latency affects Warcraft III, perhaps more than other RTS games, is through the addition of the hero characters with unique, time-sensitive abilities that make the game more interactive than other RTS games.

6. Future Work:

There are several areas for possibly interesting future work. Due to limitations of time and other resources, we made decisions throughout the project such as to limit the number of trials.

Some Warcraft III-specific research we feel should be pursued are:

1) Additional research verifying and extending our “three basic components” approach to Warcraft III, such as additional trials similar to ours. We feel that, in order to be fully confident in our results, more tests should be conducted with more players and more maps.

2) Extensions of the “three basic components” theory to larger portions of the game environment, such as large-scale wars, building and battling, and even to entire games played on full-sized maps. This is important to determine if our results for our three small slices of the game are also present in larger trials as well as full games.

3) Determining if latency does present a barrier to adjusting strategies. This could be done by playing games in which players are not allowed to change their strategies with and without latency, and then comparing this to games where players are allowed to try and adjust their strategy.

4) Evaluating how the formation of strategies is affected by latency, which may require in-game adaptations to the opponent’s strategy.

We also feel it would be beneficial to research or emulate other RTS games to see if our conclusions hold true for these games, as well, and under what conditions. It

would be quite useful to be able to characterize a game by the style of play and what kinds of game play elements it contains, both to allow game designers to see what options they have in making online games and for networks to be able to plan to handle the growing traffic load.

We also think researchers should look into other game genres, such as first person shooters or massively multiplayer online role playing games, to see how they handle latency, and explore how the game genre could possibly predict network usage characteristics.

Appendix A, Warcraft 3 Overview:

This is a brief overview of the recently released Warcraft III, a real time strategy game produced by Blizzard Entertainment (<http://www.blizzard.com>). Real time strategy games most often use a third person perspective. Warcraft III has four “races,” which a player can choose between when a game starts. While these races vary in many ways, the game is considered to be fair, so our research focuses mainly on one of the races, the “Humans,” as opposed to the “Orcs,” “Undead,” or “Night Elves.” Two major aspects to the game are *unit control*, and *town control*.

Town control consists of selecting what buildings are to be built or upgraded, what units are to be produced and what technologies are to be researched. To accomplish these tasks gold and lumber are required which are gathered by peasants. The Human buildings range from farms, which provide food, to town halls, which are the central building in any human town. Buildings such as barracks produce standard army units such as footmen and knights while buildings such as arcane sanctums produce units such as priests and sorceresses. Town control involves a lot of strategy in knowing when and where to build, upgrade, and research.

Unit control can be broken up into three subcategories: peasant control, exploration and battle. Peasant control overlaps a bit with town control as it is the management of peasants harvesting gold and lumber As well as building and repairing buildings.

Standard Warcraft III maps have something known as the fog of war which covers any unexplored area of the map. To reveal the terrain a player must move one of his units into the area. Exploring allows players to find things such as creeps (neutral enemies on which to gain experience for their heroes), enemy towns or units, and neutral shops at which their heroes can buy various items.

One of the most important features of any real time strategy game are battles. There are various strategies that can be used while fighting battles but at a minimum the player can let the computer’s artificial intelligence handle most of the work for him. Warcraft III has special units called heroes which are more powerful than standard units, and have additional abilities. The Humans have the following three heroes available to them:

The Paladin: The paladin is a healer-warrior, who has good hand-to-hand fighting capabilities, as well as the special abilities “heal,” “30 second invincibility,” “passive increase all friendly units’ armor,” and “resurrect 6 units.”

The Mountain King: The mountain king is a strong warrior, who is very good at hand-to-hand combat, and has the special abilities “ranged damage attack,” “area damage attack,” “passive increased damage,” and “temporary battle enhancement.”

The Arch mage: The arch mage is a frail magic user, who has a decent ranged attack, and has the special abilities “targeted area damage spell,” “summon water elemental unit,” “passive increased mana regeneration”, and “teleport 24 units to safety.”

Knight	350
Footman	160
Rifleman	270
Priest	170
Sorceress	200
Level 6 Hero	600

Table 3: Unit Score Values

Screenshots



Figure 1: Attacking a town in Warcraft III

The purpose of these screen shots is to provide an understanding of what the game looks and feels like to play the game. This screenshot shows a human town under attack from an undead army. The undead are in the upper left area of the screen. You can see some human peasants carrying lumber to the town hall, and various other goings-on.



Figure 2: Player and Neutral Units in Warcraft III

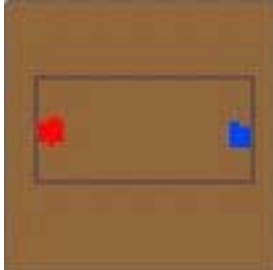
This screenshot shows a small group of human footmen, led by a paladin, battling some neutral enemies. This is a low level paladin with a heal spell and a passive armor bonus ability, as depicted by icons in the lower right of the screen.

Appendix B, First Battle Map:

Map Name: “Battle Test”

Map File: (2)Arena_v5.w3m

This map was designed starting with the “small” map grid and adding a fenced in arena that spanned from the left side of the map to the right side. The first player (Red), which is always the host, starts on the left side and the second player (Blue) starts on the right. Each player starts with the same twelve units arranged in a square. The players have the full technology tree researched before the map starts. All of the units are from the “human” race, and the break down is: two Footmen, three Priests, two Knights, two Riflemen and three level six heroes. The heroes are the Paladin, Archmage, and Mountain King. We evenly distributed the heroes’ skill levels before the map began, to help with consistency.



The map also contains a trigger to end the game if the battle takes longer than two and a half minutes, as our pilot studies on the map showed that, in that case, something had almost always gone wrong for this to happen. Figure 3 is a screenshot of the arena, the red and blue circles over each group of units denote the player’s start area and cannot be seen in the actual game.



Figure 3: Screen shot from Warcraft III Map Editor of Battle Map 1



Figure 4: In-game screen shot of our first battle map, mid-battle

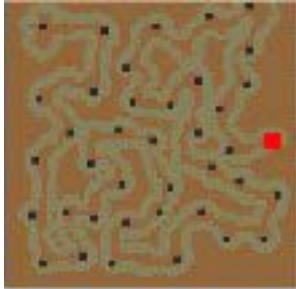
Above we have a screen shot from within Warcraft III showing the interface and the early stages of a battle.

Appendix C, Explore Map:

Map Name: "Explore Test"

Map File: (1)Explore_v1.1.w3m

Our exploration map uses a medium sized map and it consists of a winding raised cliff.



Player 1 has no units, and only serves to host the game. Player 2 (Red) has only one unit, an Undead Shade, an invisible spying unit with no attack capabilities in the game, which starts upon the raised cliff. The player must move the shade along the cliff, stepping on the waypoints in order. The waypoint graphic is called a Circle of Power within the game, and the map contains

triggers that detect when each circle is passed over. The map triggers count each waypoint as it is stepped on and once the player has moved over all 38 waypoints the game ends. During the game the player can only see the areas that they have explored. We felt that 38 waypoints were enough that players would not memorize the locations without a conscious effort, and also long enough to show results.



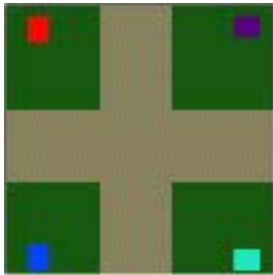
Figure 5: Screen shot from within the map editor showing the starting position of the shade.

Appendix D, Build Map:

Map Name: "Build Test"

Map File: (4)Build_v9.w3m

Our building map is split into four quadrants by mountains to keep the players separated.



Each player starts with four peasants and a town hall. They must build one of each building in their technology tree, and research each upgrade available at each building. There are three exceptions to this rule. The first is that a player is required to build a basic Scout Tower, but not to upgrade it. The second is that the player can build three blacksmiths in order to reduce the time needed to finish. (The blacksmith was a severe bottleneck otherwise.) What is the third exception? Conditions built into the map track all of these requirements. Triggers disable each type of building after it has been built once, or in the case of the blacksmith, three times. Each player is timed using triggered timers for accuracy.

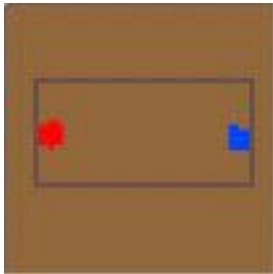


Figure 6: Screen Shot of peasants building three buildings at the start of a “build” run.

Appendix E, Second Battle Map:

Map Name: "Battle Test 2"

Map File: (2)Arena_v6.w3m



This version of the arena map differs from the previous version only in number of units and unit abilities. The units are: a Mountain King (level 6), two knights, four footmen, two Riflemen, a Sorceress, and two Priests. Again, the hero abilities were spread evenly.



Figure 7: Screen shot of the revised battle map, mid-combat.

Glossary of terms

Bandwidth: The amount of data that can be transmitted in a fixed amount of time. [WEB B]

Delay: See “latency.”

Jitter: The variation in delay between two packets. [JP02]

Lag: An noticeable problem in the performance of the network.

Latency: The amount of time it takes for a packet of data to go from source to destination. [WEB L]

Packet loss: Packets that do not reach their destination. This most commonly occurs when the network is congested, causing routers to drop packets.

Packet sniffer: A device or piece of software that logs packets traveling across a network.

Ping: A special tool used to check response time by sending packets.

Ping time: The total time, usually in milliseconds, from the time a ping packet is sent until a response is received.

Real time: A situation where the computer must handle to a steady flow of new information without interruption. [WEB R02]

Response time: The total round-trip latency.

TCP: Transmission Control Protocol: TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and guarantees that packets will be delivered in the same order in which they were sent. [WEB T]

Throughput: The actual amount of data transferred from one place to another taking into account latency and loss.

UDP User Datagram Protocol: A connectionless protocol that, like TCP, runs on top of IP networks. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. [WEB U]

References

[BCMY02] Brian Conway, Charles McAuley and Jonathan Yurek. *Evaluation of DCBT*. Spring 2002. <<http://www.cs.wpi.edu/~claypool/mqp/dcbt-eval/>>

[BEPR] *Blizzard Entertainment – Press Release*. 11 Oct. 2002 <<http://www.blizzard.com/press/020722war3.shtml>>.

[BNET] *Battle.net*. 4 Oct. 2002 <<http://www.battle.net>>.

[BNETD] *The BNETD Project :: Building the game servers of tomorrow*. The BNETD Project. 11 Oct. 2002 <<http://www.bnetd.org>>.

[CAIDA02] *Characterization of Internet traffic loads, segregated by application*. Cooperative Association for Internet Data Analysis. 3 Oct. 2002 <<http://www.caida.org/analysis/workload/byapplication>>.

[CENS] *Home Computers and Internet Use in the United States*. U.S. Census Bureau. Sept. 2001 <<http://www.census.gov/prod/2001pubs/p23-207.pdf>>.

[CH01] Haag, Chris. *Targeting, a variation of dead reckoning*. 17 May 2001 <<http://www.gamedev.net/reference/articles/article1370.asp>>.

[DB02] Bauer, Daniel et al. *Network Infrastructure for Massively Distributed Games*. IBM Research. 3 Oct. 2002 <<http://portal.acm.org/citation.cfm?id=566506&coll=portal&dl=ACM&CFID=4483302&CFTOKEN=34347262>>.

[DLJW01] LaPointe, Dave and Winslow, Josh. *Analyzing and Simulating Network Game Traffic*. 19 Dec 2001 <<http://www.cs.wpi.edu/~claypool/mqp/net-game/game.pdf>>

[DR] *Dead Reckoning for Latency Hiding in Internet Games*. 1 Oct. 2002 <http://hamsa.unl.edu/~byrav/CSCE462/Project/old/spr00/rstrasbu/Dead_Reckoning.PDF>.

[DUMNET] *DummyNet Homepage*. 27 Oct. 2002 <http://info.iet.unipi.it/~luigi/ip_dummysnet/>.

[ENA] *Ethereal Network Analyzer, The*. 24 Oct. 2002 <<http://www.ethereal.com>>.

[FRAPS] *FRAPS Homepage*. 23 Dec. 2002 <<http://www.fraps.com>>.

[GA01-a] Armitage, Grenville. *Lag over 150ms is unacceptable*. 17 May 2001 <http://www.geocities.com/gj_armitage/things/quake3-latency-051701.html>.

[GA01-b] Armitage, Grenville. *Sensitivity of Quake3 Players to Network Latency*. 1 Nov. 2001 <http://www.geocities.com/gj_armitage/q3/imw2001/poster110101.pdf>.

- [JF02] Farber, Johannes. *Network Game Traffic Modeling*. University of Stuttgart. 3 Oct. 2002
<<http://portal.acm.org/citation.cfm?id=566508&coll=portal&dl=ACM&CFID=4483302&CFTOKEN=34347262>>.
- [JP02] Perser, Jerry et al. *Terminology for Benchmarking Network-layer Traffic Control Mechanisms*. Network Working Group. 3 Oct. 2002
<<http://www.ietf.org/proceedings/01mar/I-D/bmwg-dsmterm-00.txt>>.
- [MM02] Mauve, Martin, Fischer, Stefan and Widmer, Jörg. *A Generic Proxy System for Networked Computer Games*. University of Mannheim et al. 3 Oct. 2002
<<http://portal.acm.org/citation.cfm?id=566504&coll=portal&dl=ACM&CFID=4483302&CFTOKEN=34347262>>.
- [NIST] *NIST Net Home Page*. 27 Oct. 2002 <<http://snad.ncsl.nist.gov/itg/nistnet/>>.
- [PJMI02] Popovic, Aleksandra, Jovic M., Milutinovic V., Inic M.. *Infrastructure for Surgery over the Internet*. 2 Oct 2002
<<http://www.ssgrr.it/en/ssgrr2002w/papers/227.pdf>>.
- [PKDM02] Key, Peter and McAuley, Derek. *Different QoS and Pricing in Networks: where flow-control meets game theory*. Microsoft Research Limited. 3 Oct. 2002
<<http://www.research.microsoft.com/network/publications/ieeproc.pdf>>.
- [PKT] *Application Performance Solutions from Packeteer*. 3 Oct. 2002
<<http://www.packeteer.com/products/packetshaper>>.
- [SERZ02] Schaefer, Christian, Enderes T., Ritter H., Zitterbart M. *Subjective Quality Assessment of Multiplayer Real-Time Games*. 3 Oct, 2002 <<http://www.ibr.cs.tu-bs.de/events/netgames2002/presentations/schaefer.pdf>>.
- [SIMNET] *An Introduction to the Internet Networking Environment and SIMNET/DIS*. 1 Oct. 2002 <www.web3d.org/WorkingGroups/vrtp/dis-java-vrml/DISIntro.ps>.
- [SJ02] Joyce, Sarah. *Traffic on the Internet*. Waikato Applied Network Dynamics. 3 Oct. 2002 <<http://wand.cs.waikato.ac.nz/wand/publications/sarah-420.pdf>>.
- [SMKC02] McCreary, Sean and Claffy, KC. *Trends in Wide Area IP Traffic Patterns*. University of California, San Diego. 9 Mar. 2002
<<http://www.caida.org/outreach/papers/2000/AIX0005/AIX0005.html>>.
- [SURG] Mitsuishi Mamoru, Watanabe T., Nakanishi H., Hori T., Watanabe H. and Kramer B.. *A Tele-micro-surgery System across the Internet with a Fixed Viewpoint/Operation-Point*. The University of Tokyo et al. 3 Oct. 2002
<<http://www.computer.org/proceedings/iros/7108/volume2/71082178abs.htm>>.

[WEB B] *Bandwidth* – *Webopedia.com*. 3 Oct. 2002
<<http://www.webopedia.com/TERM/B/bandwidth.html>>.

[WEB L] *Latency* – *Webopedia.com*. 3 Oct. 2002
<<http://www.webopedia.com/TERM/L/latency.html>>.

[WEB R02] *Real Time* – *Webopedia.com*. 11 Oct. 2002
<http://www.webopedia.com/TERM/r/real_time.html>.

[WEB T] *TCP* – *Webopedia.com*. 18 Feb. 2002
<<http://www.webopedia.com/TERM/T/TCP.html>>.

[WEB U] *User Datagram Protocol* – *Webopedia.com*. 18 Feb. 2003
<http://www.webopedia.com/TERM/U/User_Datagram_Protocol.html>.

[WOB] Wobus, John. *DHCP FAQ*. 26 Oct. 1998 <http://www.dhcp-handbook.com/dhcp_faq.html>.