Performance Evaluation of Multi-Player Online Games in IEEE 802.11g Wireless Networks

A thesis presented

by

Jing Wang

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science

The University of Manitoba

Winnipeg, Manitoba

October 2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Performance Evaluation of Multi-Player Online Games in IEEE 802.11g Wireless Networks

BY

Jing Wang

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

OF

MASTER OF SCIENCE

Thesis advisors

Dr. Yanni Ellen Liu and Dr. Michel Toulouse

Author

Jing Wang

Performance Evaluation of Multi-Player Online Games in IEEE 802.11g Wireless Networks

Abstract

Multi-player online games (MOGs) have become very popular in the last few years. Meanwhile, wireless network access has become widely used and accepted. There is now an increasing demand for playing MOGs in wireless environments. However, the unique characteristics of wireless networks raise the issue of whether MOGs are *wireless playable*, i.e., whether the wireless environments can meet MOGs' performance requirements to support quality game plays. The purpose of this thesis is to determine the capability of one specific wireless LAN protocol, IEEE 802.11g, in supporting MOGs. In particular, I adopt an experimental approach to evaluating the network-layer performance of MOGs on an IEEE 802.11g (Wi-Fi) network.

In this study, I first select five factors that may affect wireless network performance, and may potentially impact game performance. They are: the number of game clients, the amount of UDP video background traffic, the existence of FTP background traffic, the use of the WEP encryption, and the distance between wireless clients and the access point. Then I perform a 2^k factorial experimental design to determine the major factors. During this step, the WEP encryption and the distance are removed from the list of factors that can have significant impact on game perfor-

mance. After that, I select more levels for the other three factors in order to study in more detail their impact on game performance in a Wi-Fi network. The major finding of this research is that background traffic, especially the existence of FTP background traffic, significantly affects the game performance in an IEEE 802.11g network. The number of game clients, when combined with background traffic, can have a large impact on the game performance as well.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisors, Dr. Yanni Ellen Liu and Dr. Michel Toulouse, for their help on establishing and guiding this study; thank the Telecommunication Research Laboratories of Winnipeg for providing physical spaces, wireless devices, and financial support; and thank the Department of Computer Science in the University of Manitoba for providing computers to my experiments and with prompted help.

I would also like to thank my family and my boy friend for their moral support to my study.

Chapter 1

Introduction

Multi-player online games (MOGs) are computer games where multiple players simultaneously participate in a game session over a computer network. The multi-player online game market has developed rapidly over the last few years. Its popularity is substantiated by the availability of high-speed residential access networks and affordable high performance personal computers. Most game clients today are connected to game servers by wired networks. With the increased use of wireless devices, such as smart phones, wireless communication enabled Personal Digital Assistants (PDAs) and laptops, it becomes possible for players to access Internet game servers from a wireless device and thus play from those wireless devices. Over the past few years, the IEEE 802.11 (Wi-Fi) wireless networks have gained wide deployment. Wi-Fi network access points are commonly seen in coffee shops, office buildings, university campus, airports, and many residential homes. The capacity of Wi-Fi networks has also kept increasing. A Wi-Fi network interface has become a standard built-in on many of today's laptop computers. In view of these advances, it is anticipated that participating

in a MOG from a Wi-Fi environment will become more and more common.

1.1 Motivation

Wireless networks differ from wired networks in terms of capacity, media access control technologies such as collision detection methods, connection control, and security controls. These characteristics at the data link layer affect the network-layer performance in terms of various performance metrics such as bandwidth, delay, jitter and packet loss rate. Network-layer performance in turn affects the game performance at the application layer in terms of game consistency and game responsiveness [38]. To better support MOGs, the capability of underlying networks needs to be evaluated with respect to the Quality-of-Service (QoS) requirements of these games.

In this research, the performance of MOGs on an IEEE 802.11g network is studied. Differ from many existing studies in which simulation modelling was used, in this study, I take an experimental approach to analyze in detail the wireless network performance, which is then compared with some known game performance requirements to determine the capability of a Wi-Fi network in supporting games.

1.2 Objective of This Study

The purpose of this research is to study how well the IEEE 802.11g protocol supports MOGs. The performance of MOGs can be intuitively represented as game-play quality, which includes the response rapidity on any changes made to the controlled object in the game, the consistency and synchronization of the game states between

the game server (GS) and game clients (GC), and the time used to accomplish a task in the game. When MOGs are played over a computer network, network performance metrics such as the network bandwidth, latency, and jitter may have an impact on game performance. They may affect the delay observed by the GC from the time a player makes a change on the controlled object to the time the result is reported to the server and reflected back to the client side. This delay eventually leads to a change in game responsiveness, consistency, and synchronization capabilities. In addition, because most MOG implementations use User Datagram Protocol (UDP) at the transport layer, which provides a un-reliable service – instead of the Transmission Control Protocol (TCP), which provides a reliable transport, some messages sent between the GCs and the GS may be lost during a game play. This may also lead to inconsistent and un-synchronized game states among the GCs and the GS. The multi-player online game quality can be lowered greatly if the network packet loss rate is too high.

Research has been conducted on MOGs' performance in wired environments [2, 30, 34, 37]. However, less attention has been paid to investigating MOGs gaming quality in wireless networks. Since it is not known precisely how wireless communication impacts game performance, the conclusions from these studies may not be applicable to wireless environments. Therefore, it is still open whether the wireless technologies being used today offer an acceptable game performance for wireless players, and if so, in which way. My study aims to draw some conclusion about how much the IEEE 802.11g wireless protocol impacts game performance, and under which conditions such a wireless network can support MOGs. These results can be used when revising

the wireless protocols or applying QoS strategies to better support games in a wireless network environment.

In this study, I first determine which wireless communication characteristics, such as the network latency, bandwidth, and packet loss rate, affect MOGs' performance. Then I study a set of factors that may affect those characteristics, and identify the major ones among them. In this work, the initial set of factors are the number of game clients, the distance between game clients and the wireless access point (AP), the enabling of data encryption, and the inclusion of File Transfer Protocol (FTP) and video streaming background traffic. In this study, I adopt existing MOGs' performance requirements [2, 34] to determine the network conditions under which a good gaming experience can be obtained. These network game performance requirements are established for a First-Person-Shooter game, called Half-Life. I then study the major factors identified above using more factor levels to find the situations that meet these game performance requirements. My experimental results show that FTP and video streaming traffic significantly affect the game traffic performance, whereas the distance and the use of data encryption have rather minimal impact. I have also observed that when the amount of background traffic is moderate to high, the performance degrades as the number of game clients increases. A detailed analysis is presented in Chapter 4 to provide insight into how the game traffic performance is affected by the background traffic and the number of game clients. Finally, based on the observations, I propose a number of QoS strategies that may be used to better support FPS games in a Wi-Fi environment.

In my investigation, I assume the game server is located on the same local area

network (LAN) as the Wi-Fi AP to which game clients are connected; the scenarios in which a Wi-Fi network acts as an access network to a wide area network (WAN) and the game server is remotely located from the game clients are not considered. It should be noted however, that my results would provide useful insight into the QoS support to games in a wide-area wireless/wired environment, when combined with results from the WAN performance studies. For example, assuming that an end-to-end latency requirement for interactivity is known, given the average delay performance on the wireless segment of an end-to-end game traffic path, one may estimate what range of average latency is required in the wired segment of the same path.

1.3 Contribution

The main contribution of this study includes the following:

1. Coming up with the experimental design and the construction of a test-bed for game performance study in Wi-Fi networks;

2. An identification of significant factors that may affect the game performance in a Wi-Fi network;

3. Understanding the impact of major factors on the game traffic performance in a Wi-Fi network;

4. An analysis of the interaction between TCP and UDP traffic in a Wi-Fi network;

5. Two QoS strategies that network service providers may use to provide quality wireless networking service to FPS game players.

1.4 Thesis Organization

The organization of this thesis is as follows. Chapter 2 describes the background and related work of this study. In Chapter 3, the research methodology and experimental design are presented. In Chapter 4, I present and analyze the experimental results. Finally, Chapter 5 contains the conclusions of this study and suggestions for future work.

Chapter 2

Background and Related Work

To realize this experimental study, I had to identify the type of MOGs which places the most demands on the network performance; study the characteristics of the chosen type of MOGs; decide which wireless protocol to study; determine the performance metrics and game performance requirements to be applied in my study; and survey network performance studies that are based on the chosen wireless protocol, especially the studies that involve real-time applications. In this chapter, these subjects are addressed in sequence.

2.1 MOGs Traffic Characteristics

I have chosen a specific type of MOGs which has the most stringent requirements on the network performance. Consequently, if my study concludes that the selected type of game is wireless playable, then it is likely that other games which demand equal or lower network capabilities will also be wireless playable. In this section,

I introduce a taxonomy of the current MOGs, and identify the type of games that generates the highest traffic load according to a number of game traffic studies.

Generally, researchers study MOGs according to the following categorization [23, 38, 40]:

- First-Person Shooter (FPS) games. In a FPS game, a player controls one character in the game, sees through the eyes of this character, who is equipped with weapons, and tries to kill as many enemies as possible. The game terminates when the winner(s) and loser(s) are decided according to pre-defined rules. FPS games are 3D games, and only allow a limited number of players (fewer than fifty) in one round. They are one of the most popular types of games [9]. Examples of this type of MOGs are Quake [19], Doom [19], Counter Strike [11], and Unreal Tournament [27].

- Massive Multi-player Online Role-Playing games (MMORPGs). Lineage2 [10] and EverQuest2 [21] are two such games. In a MMORPG, each player chooses one character from a set of characters; each type of character has distinct abilities. A character tries to collect as much experience, money and equipment as possible, and enters a higher level when the specified tasks have been accomplished or a given level of experience has been achieved. Generally, the number of game players in one game world depends on the capability of the game server, and there is no termination point for this type of games.

- Real-time Strategy games. Each player controls one team or one domain, while competing/fighting with other players. In this category of games, "Real-time

fights/matches are not as important as overall strategy" [40]. The Age of Kings [8] and Warcraft [12] fall into this category.

- Turn-Based strategy games. Examples are card games and chess games. Players make their moves in turn one by one.

- Simulators, such as car racing games. They simulate specific kinds of scenarios, for example, driving a vehicle. Players compete with each other by operating their own simulated vehicles. This type of games has a small number of attendees, and the players usually concentrate more on the competition rather than on the direct interaction with other players.

In order to determine which kind of games are the most network demanding, I first define a *high network demanding game* as a game that sends frequent messages of a size comparatively larger than that of other games. Thus, a high network demanding game should have a large bit-rate and a large packet size.

I have surveyed a set of game traffic studies according to the above categories, and have summarized the results of these studies in Table 2.1. In this table, the second column identifies the games that I have surveyed, while the first column indicates to which of the above category each game belongs. The third column shows the studies I have analyzed for each game, while the fourth one identifies the platform that has been used by each study. Finally, the fifth and sixth columns describe the game traffic of each game in terms respectively of the mean bit-rate and mean packet size of the client-to-server ("Client") and server-to-client ("Server") traffic.

There are several notes regarding to the Table 2.1. The game Lineage [35] is measured only at the server side. It only provides the duration of test (8 days 7

Table 2.1: Game Traffic Measurement

Game Category	Game	Study	Platform	Mean Bit Rate(bps)	Mean Packet Size (byte)
FPS	Counter Strike	[14]	WAN	Client 40K	Server 174.73 Client 91.28
		[13]	LAN	Server 16.4K Client 15.7K	Server 127 Client out 82
	Quake	[4]	LAN	Client 24byte/update	
		[23]	LAN LAN	Server 7.69K Client 12.8K	Server 77.9 Client 45
MMORPG	Lineage	[35]	WAN	Client 1.31K Server 1.82K	Server 36.739 Client 9.035
Real-time Strategy	Warcraft	[37]	LAN	6.8K (LAN) 4K (WAN)	Typical sizes 6, 9
	Age of King	[23]	LAN	Server 3.24K Client 3.06K	Server 33.3 Client 232.0
Turn-Based Strategy	Panzer General	[23]	LAN	Server 118 Client 85.9	Server 44.2 Client 30.0
Simulator	Grad Prix 3	[23]	LAN	Server 4.62K Client 4.62K	Server 26.3 Client 26.3

hours 42 min and 8 seconds) and the overall throughput of the game traffic. During this period, the number of connections and the durations of each connection are not fixed. To make Lingeage comparable to the other MOGs that I have surveyed, I first calculated the overall produced traffic using the given mean bit-rate multiplied by the overall length of time. Then, I multiplied the average duration of each connection with the total number of connections to obtain the aggregated connection time. At the end, I divided the overall produced traffic with the aggregated connection time. This yields the average bit-rate for each client. In Table 2.1, the study on Warcraft [37] did not provide the mean packet size but the typical size of the packets. Since I am comparing the packet size range for the majority packets in each game, I consider

this typical size comparable to the mean packet size given by other studies.

From Table 2.1, we can see that Counter Strike and Quake, which are under the FPS category, have the highest mean bit-rate and the largest mean packet size, while turn-based strategy games have the lowest mean bit-rate and the smallest average packet size. Therefore, I choose FPS games for my research.

Table 2.1 also shows that the results for the same game may differ. For example, for the game Counter Strike, Feng et al. [14] and Färber [13], obtained different results. I suspect these differences have arisen because either the game players or the networks were not of the same skill level or capability. Nevertheless, the results from all of these studies indicate clearly that the FPS games' mean bit-rate and mean packet size are the highest among all categories. So the difference in the above studies does not affect my decision for choosing FPS games as the target of this study.

2.2 FPS Games

In order to study FPS game performance in wireless environments, I first need to understand how a FPS game is normally implemented, especially in terms of its demand on network delivery.

For implementation, FPS games usually adopt a client/server architecture, where there is one game server and a number of game clients connected to the server. Using this architecture, each client (a player's machine) is responsible for displaying the game world for that player. It records every movement of the player, e.g., firing a bullet at an enemy player, and periodically reports all the moves of this player to the game server using state update messages. The game server is in charge of gathering

the state update information from all game clients, maintaining the states of the entire game world, and periodically distributing the state update to the clients who are affected by player moves. Thus, in FPS games, state update messages are transmitted in the network along two directions: from the game server to a game client, and from a game client to the game server. The shorter the latency in delivering a state update from the client to the server and then to the affected clients, the more realistic the game play. Because of their highly interactive nature, FPS games often have the most stringent requirements on network performance. Such requirements may be in terms of the delay and loss ratio performance in delivering the game traffic.

2.3 Wireless Protocols

Network communication relies critically on the transmitting medium and the protocol used. Both the medium and the protocol may have significant impact on network performance in terms of performance metrics such as latency, jitter and packet loss rate. Network performance is critical to MOGs performance, because MOGs rely heavily on network communication capabilities. In this research, I study the MOGs performance in an IEEE 802.11g network. Prior to choosing the IEEE 802.11g Wi-Fi network, I have surveyed the main wireless LAN protocols. Also, I have reviewed the techniques used at the Media Access Control (MAC) layer of the IEEE 802.11 standards. This review is necessary in order to understand how wireless communication impacts network performance, which in turn affects MOGs' performance.

The most popular wireless LAN protocols today are Bluetooth and the IEEE 802.11 standards. Bluetooth is designed for low power-supply usage. It has some

important limitations: it transfers data with low throughput – up to 721Kbps; has a radio range of only 30 feet; and its signals cannot penetrate through walls. In contrast, the IEEE 802.11 suite, also called Wi-Fi (short for wireless fidelity), supports much higher throughput – up to 54Mbps; has a greater transmitting range – 100 or 300 feet; and its signal can transmit through walls. Because of these characteristics, 802.11 protocols are primarily used in local area network environments, while Bluetooth is more popular in exchanging small files between devices that are closely located. An example of the use of Bluetooth is transferring pictures between a PC and a digital camera. For game-play in wireless environments, I adopt the IEEE 802.11 standard for my study.

There are three extensions to the IEEE 802.11 that have been or are being widely used in wireless LAN devices. They are 802.11a, 802.11b, and 802.11g standards. Among them, 802.11b was the first one to be widely used. It provides up to 11Mbps bandwidth in the 2.4GHz band, and has a maximum radio range of 300 feet. 802.11a transmits in the 5GHz band with up to 54Mbps bandwidth. Because it uses a higher frequency band than 802.11b, its maximum range is shorter – 80 feet. 802.11g provides up to 54Mbps bandwidth for wireless LANs in the 2.4GHz band, and is compatible with 802.11b. It has the same distance capability as 802.11b and has about the same bandwidth as 802.11a. Because 802.11g and 802.11a are more advanced than 802.11b, and also because 802.11g is compatible to 802.11b with larger transmitting distance than 802.11a, 802.11g has become very popular. So I study 802.11g protocol in this research.

The IEEE 802.11g protocol supports two operating modes: infrastructure mode

and ad hoc mode. The former assumes the presence of wireless access points when forming a Wi-Fi network, and mobile nodes communicate via these access points; the latter assumes the formation of a Wi-Fi network without any AP, and mobile nodes communicate with each other directly.

In order to analyze the game performance results obtained in an 802.11g network, I have looked at the IEEE 802.11 protocol, and learned how it delivers data correctly. This knowledge also can help me in choosing the potential factors that may affect network performance in such a wireless environment. In this research, I study the game performance at the network layer, which is related to the data link layer. The data link layer can be divided into the MAC layer and Logical Link Control (LLC) layer. My study focuses on the IEEE 802.11 MAC layer protocols, which deal with access control, data transmission, reliability and security issues. In the rest of this subsection, I describe the techniques used in 802.11g MAC layer.

The IEEE 802.11g standard uses the Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) MAC protocol for access control; acknowledgement(ACK) and Request-To-Send (RTS) / Clear-To-Send (CTS) frames to provide reliability; and authentication mechanisms for security and privacy [3]. In a typical scenario, when a station, say A, wants to send data to station B, the CSMA/CA protocol listens to the medium until the medium becomes idle. Then, station A runs a back-off algorithm to decide a period of time to wait before sending the data. If the medium becomes busy before the back-off period ends, station A will restart the whole process, and the rest of the back-off time will be used in the next waiting period. If the medium is still available upon the end of this back-off period, and if RTS/CTS are enabled, the

communication will be started with a RTS frame sent from station A to station B. Station B should reply with a CTS frame if it is ready. If no CTS is received within certain period, station A will wait for another back-off time before trying again. If RTS/CTS are not enabled, the data frame will be sent. While the data is under transmission, station B should acknowledge station A with an ACK frame for every data frame arrived at B and has B as the only destination. If station A does not receive the ACK for a data frame, it will resend the data.

One purpose of using RTS/CTS is to avoid hidden node problems [41, 45], e.g. two machines that are out of the range of each other but plan to communicate to a third machine at the same time, and a collision occurred at the third machine. RTS/CTS are also used to reserve the channel for certain period of time [32]. The duration is included in both frames; other stations on the network learn this duration from these two frames, and refrain from sending during the reserved duration. Transmitting RTS/CTS frames brings overhead to the data transmission. If the transmitted data size is very large and a retransmission of the data costs more time than transmitting the RTS/CTS frames, the use of RTS/CTS will improve the performance. If the transmitted data size is very small and retransmitting the data cost less time than transmitting the RTS/CTS frames, the use of RTS/CTS may decrease the performance. IEEE 802.11g protocol uses a threshold value in bytes to determine when should use RTS/CTS. When a frame to be sent has a size larger than this threshold, RTS/CTS are enabled. Otherwise, RTS/CTS are not used.

Because the data transferred in the air can be easily captured by any station in the radio range, security problems must be handled. An IEEE 802.11g network

provides two authentication methods: open and shared key methods. The former has no limitation to wireless clients to connect to the AP. The latter encrypts data using Wired Equivalent Privacy (WEP) algorithm to avoid eavesdropping [3].

The use of ACK, RTS/CTS frames, the back-off procedure, and the WEP encryption increases overhead, which could impact network performance in terms of delay, and even packet loss for real-time applications. Among them, the use of RTS/CTS and WEP are optional. In practice, most deployed Wi-Fi networks choose a large threshold value, essentially disabling the channel reservation function [22]. In addition, in my designed experimental environment, there exists no hidden node (refer to Sections 3.3 and 3.6). So I only study the impact of using WEP. For ACK frames and back-off procedure, I consider their impact on game performance when I analyze the experiment results.

2.4 Game Performance Studies

In the literature, researchers have studied the performance of FPS games at both the network layer and application layer [2, 34]. In particular, the relationship between the network-level performance and user-perceived game performance has been examined, and the requirements on the network-level performance for adequate support to FPS games have been established. These requirements are independent of the network technology used, so they can be applied to evaluate the capability of wireless networks in supporting FPS games. In my research, I use the performance metrics in those studies. I compare these performance requirements with the network-level game traffic performance in my experimentation to obtain an in-depth view of the

wireless environment's capability in supporting FPS games.

In my experimental study, I use a traffic model for FPS game Half-Life (refer to Section 3.4.1). However, none of the existing FPS game performance studies is based on this game. This adoption of FPS game performance requirements in my study is under the assumption that all FPS games have similar performance requirements on the underlying networks.

Beigbeder et al. [2] tested the effect of loss and latency on user performance for a FPS game called Unreal Tournament 2003 (UT2003). They have observed the typical packet loss and latency at UT2003 Internet game servers, and have emulated the deduced packet loss and latency in a local network environment. They monitored the performance of UT2003 game sessions at the network, application and user levels in a controllable environment. In their experiments, they tested the movement and precision shooting hit-rate under the assumption that a large portion of these two major behaviors in FPS games (movement and shooting) are loss and latency sensitive. They found that round-trip-time (RTT) between the game client and the game server is noticeable when RTT is over 75 ms; and the game is less enjoyable when latency is over 100 ms. They also found that for the game UT2003, loss rate is noticeable when it is at 3%, "with the primary artifact noticed being that the game would sometimes not display animations for shots that were fired" [2]. Their study also indicated that shooting is greatly affected by latency. With even modest latency (75-100 ms), accuracy and number of kills can be reduced by up to 50%. The authors recommended that players should avoid servers with ping times over 75 ms or packet loss ratios over 3%, although they also mentioned that users rarely notice packet loss

rate up to 5% during a typical network game. However, the effect of higher than 5% loss ratios has not been commented on. Note that high loss ratios often occur in a wireless environment. Therefore, packet loss ratio is considered a very important performance metric of this thesis study.

Quax et al. [34] measured both objective and subjective game performances of the game UT2003. As an objective measurement of game performance, they computed the number of times a player effectively kills another player minus the number of times the player is killed. As a subjective measurement, the game participants filled out a short questionnaire after every session regarding the network quality they experienced during the session. Quax et al. concluded that network impairment does have a negative influence on the affected players' perceived game quality and performance. There is an indication that if the round-trip delay is 60 ms or greater, the player experiences the impairment as disturbing.

2.5 802.11 Performance Studies

The capability of Wi-Fi networks on supporting real-time applications has been studied previously. Some of these studies are based on 802.11b protocol, and some are based on 802.11g.

Cao et al. [5] examined an 802.11b network's capability in supporting media streaming applications. In their study, a number of wireless terminals were operated in the ad hoc mode. Those terminals performed media streaming while the media file is located at one other wireless terminal. They observed that the maximum throughput achieved was 4.6Mbps, and the maximum allowed number of users

for this specific type of application was 8.

Wijesinha et al. [43] measured the throughput of UDP traffic in an 802.11g network that has single or multiple wireless terminals under various physical distance with or without the use of RTS/CTS. The packet size they used were between 100 and 1472 bytes. They found that "in almost all cases the observed throughput is well below 50% of the 802.11g maximum data rate of 54Mbps", and the use of RTS/CTS degrades the network throughput. They also concluded that the variance of wireless terminals' physical distance has little impact on throughput when the packet sizes are small. When the packet sizes become large, the impact increases inconsistently.

Nguyen et al. [28] measured the TCP performance of 802.11b network with the use of two FPS games, namely Half-Life and Quake3 [20] as background traffic. Because this study focused neither on game performance nor UDP traffic, their conclusion could not explain in which way and by how much the game play quality was affected. However, the authors found that 20 Half-Life or 10 Quake3 game players take more than 3.5Mbps of bandwidth, even though the actual required bandwidth is less than 1Mbps.

Palazzi et al. [29] performed a simulation study on 802.11g performance in a home network environment. They studied wireless network throughput, RTT and jitter in a topology that consists of a wireless LAN connected with a wired WAN, and utilized a combination of network traffic (movie stream, game traffic, video chat and FTP). The chosen topology was to represent a scenario where home users are connected to the Internet through a local AP. The factors used were the number of MAC layer retransmissions and the queue size. Also, the authors simulated different values of

the distance between the AP and wireless clients, and the thickness of obstructions in the middle. They found that, to achieve a high throughput with a relatively low latency and a small jitter, the best number of retransmissions are 3 and the queue size should be small, e.g. 50 packets. This study considered MOG traffic requirements but only from the RTT point of view. The packet loss rate, which also contributes to the game play quality, was not mentioned. This study also did not provide the maximum allowed number of wireless game players, which is one of the target in my research.

Gretarsson et al. [16] studied the interactions between various user applications on network layer performance in an 802.11g network. This study considered the scenarios of two wireless clients connecting to the AP when one (machine A) has good connection, and the other (machine B) has either good or bad connection, where a good connection has an average signal strength $\geq -70dBm$ and a bad location has an average signal strength $\leq -75dBm$. A FTP file download session was run on the first client. On the second client, there was either a TCP file download or a streaming media application running. The streaming media application may use TCP or UDP protocol. So the applications with connectivity status are termed as good/bad TCP/UDP stream, and good/bad TCP download. Of each situation, Gretarsson et al. measured the signal strength, the WLAN channel capacity of the network, the throughput, upstream retry fraction, loss rate, and cumulative RTT. They found that, in terms of the signal strength, machine A has the same performance regardless B is in good or bad connection; while the performance on machine B decreased when the connection changes from good to bad. However, when B has

bad connection, the throughput, upstream retry fraction, loss rate and cumulative RTT on both clients were dropped. In addition, when B has a bad connection, the different type of application running on B and the related transmission protocol lead to different performance characteristics in terms of throughput, upstream retry fraction, loss rate and cumulative RTT. Comparing to a bad TCP stream and a bad TCP download, a bad UDP stream has much less upstream retry fractions, lower RTT, and extremely high loss rate. When B is running a bad TCP download, both machines have average throughput around 2.5Mbps; a bad UDP stream on B has an average throughput around 2.5Mbps, while A's throughput is near 0; a bad TCP stream leads to the average throughput close to 0 on B, and between 3 to 5 Mbps on A. Gretarsson et al. concluded that the network layer queue, the choice of transmission protocol, and the application layer behaviors have impact on the performance. This study limits the number of wireless clients to two, and mainly focused on the network performance when one client has bad network connection.

Chapter 3

Experimental Design

This chapter describes the preliminary tasks I have completed in order to carry out the actual experiments. I identify a number of factors that may impact game performance; define the performance metrics to be measured in the experiments; decide how to obtain the values for each performance metric; design a test-bed to perform the experiments; and describe one game traffic emulator and one video streaming traffic generator that I have implemented in order to generate reproducible traffic in the experiments. In addition, I determine the number of replications for each experiment by calculating the confidence intervals. After all these are completed, I present the results from a set of 2^k factorial experimental design studies. These designs are devised to determine among a set of selected factors the important ones that significantly affect the performance. Then, I complete the experimentation using more levels for each of the important factors.

3.1 The Initial Set of Factors

In this research, I had to find the factors that could have major impact on game traffic performance. Therefore, I first select a set of factors that cause low bandwidth, high latency, and high packet loss rate in wireless networks. Then in a later step, I use these factors in a 2^k factorial experimental design study to quantify the impact of each factor in order to narrow down the initial set of factors to a few major ones. In this section, I explain how the initial factors are selected. The 2^k factorial experimental design study is described in Section 3.6

There are three categories of factors that may impact network level game traffic performance: wireless network factors, game related factors, and background traffic factors in wireless environments.

3.1.1 Wireless Network Related Factors

Wireless network factors are related to the wireless network. Some factors that may impact game traffic performance include the topology of the wireless network, the use of RTS/CTS, authentication methods, and the physical distance between the wireless clients and the AP. My study focuses on a simple topology in which all the wireless clients belong to one Basic Service Set (BSS), i.e. all the wireless clients are connected to the same AP. More complex networks can be constructed using basic sub-networks. Thus, results from my study may be useful for studying the performance of more complex networks.

In my test-bed, all the machines are within the transmission ranges of each other. I eliminated RTS/CTS from the factor list because currently the RTS/CTS threshold

are not normally used in most Wi-Fi deployed networks [22]. The reason I consider distance between the game client and the AP as a factor is because radio waves fade naturally while the transmitting distance is getting larger. The IEEE 802.11 protocols use a dynamic rate shifting method that adjusts the data rate based on distance; when the distance increases, the data rate decreases [3]. Therefore, different distance levels may lead to disparate network level performance.

3.1.2 Game Related Factors

Game related factors impact directly the game traffic performance, regardless of the underlying communication network used. They may include the game map being used; the number of game players; and the game server and clients' hardware configurations. The number of game servers may be considered as a factor for some MOGs, because they may use more than one GS to support a large number of GCs. However, since FPS games usually have a small number of GCs (less than fifty), they normally only need one GS. Different game maps result in various gaming complexity, and may produce different message sizes between the GS and GCs. However, from Section 2.1, we can see that FPS games do not have high bit rates or large mean packet sizes. Therefore, variation in message sizes will not have great impact on the traffic generated by the game. So, I decide to use a map with the largest packet size in my study, and eliminated the game map from the factor list. Since I used emulated game traffic in my evaluation (see Section 3.4), graphics rendering power on the game clients is not considered in this study.

3.1.3 Background Traffic Related Factors

In this study, the background traffic (BT) in the experimented wireless network refers to the data exchanged between the AP, which also serves the wireless game players, and other wireless clients that are not related to gaming. An example of BT is the FTP traffic when a wireless device downloads/copies a file via the same wireless network. A wireless multi-player online game could be disturbed if the BT is too heavy. For my experiments, I have selected two network applications: video streaming applications and FTP applications. Both types are very popular in terms of current network usage [17]. Video streaming is a real-time application that initiates the communication using TCP, but sends real-time video frames using UDP. These video frames are sent at a constant frame sending rate. There are many video encoding schemes among which MPEG-4 and DivX video streams are two popular standards on the Internet right now. A MPEG-4 or DivX encoded video stream has a bit-rate ranging from 64kbps to 4Mbps [18]. FTP is a TCP-based protocol that may absorb all of the available bandwidth due to the congestion control mechanism of TCP. When competing with non "TCP friendly" protocols such as UDP, TCP may back-off and thus receive much reduced amount of bandwidth compared to when there is no non "TCP friendly" traffic. Because of the difference between UDP and TCP, I consider the above two types of BT separately as two factors. Although HTTP traffic is also a popular type of traffic, it is not used in this design. The purpose of have more than one background traffic factor is to discovery the impact of different transport layer protocols. Because both FTP and HTTP traffic use TCP protocol, select one from them is considered adequate to achieve this goal. Adding HTTP traffic to the

background could be the future work.

On my experimental network, there are two dedicated machines for exchanging background traffic between them: a background traffic server (BS) and a background traffic client (BC). BT is generated and sent from the BS (wired) to the BC (wireless). This direction was chosen because in real world, people use wireless devices mainly for their own pleasure, such as entertainment, but do not often intend to provide services to other clients in the local network or on the Internet. In addition, Internet Service Providers usually provides much lower bandwidth for upstream traffic than downstream traffic.

3.1.4 The Preliminary Factors

The identified initial set of factors that are selected for this study includes:

Factor-I: NC: The number of wireless GCs.

Factor-II: $Video$: The amount of background video streaming traffic.

Factor-III: FTP: The number of background FTP sessions that are kept alive during the entire game session.

Factor-IV: WEP: Authentication methods.

Factor-V: $Distance$: The distance from the GCs to the AP.

3.2 Performance Metrics

Previous studies of game performance requirements established the range of packet loss rate and round-trip delay for acceptable FPS game play. Based on these studies, I define the performance metrics of my experiments as follows:

1. the packet loss ratio, LR_{s2c}, for game packets sent from the GS to the GCs,
2. the packet loss ratio, LR_{c2s}, for game packets sent from all the GCs to the GS, and
3. the average round-trip-time, RTT, for game packets sent from a GC to the GS.

Note that these metrics are at the network layer. It is different from the game performance at the application layer above, or frame level performance at the data link layer below. For convenience, the performance experienced by the game traffic at the network layer is called "game traffic performance" in this thesis.

As described in Section 2.4, both latency and loss ratio performance at the network layer may have a major impact on the game performance at the application layer. The importance of latency requirement has been established in the literature. For loss ratio, although it is found that a loss ratio of up to 5% has minimal impact on a user's experience in the literature, some of my experimental results showed that the loss ratio could be significantly higher than 5%. These loss ratio results may be important and therefore are reported in this thesis.

To calculate LR_{s2c}, two counters, N_s and N_c are used. N_s records the total number of game packets that are sent by the GS to all the GCs in an experiment. N_c is the total number of game packets received by all the GCs from the GS in an experiment.

The game traffic generator on each GC records the number of packets received. N_c is obtained from the sum of all these values. Then, LR_{s2c} is calculated by

$$LR_{s2c} = \frac{N_s - N_c}{N_s} \times 100\%$$

Both N_s and N_c are collected by the game traffic generator.

Similarly, to calculate LR_{c2s}, the total number of packets that are sent by all the GCs, N'_c, and the total number of packets that are received by the GS, N'_s, are also collected by the game traffic generator. LR_{c2s} is then calculated by

$$LR_{c2s} = \frac{N'_c - N'_s}{N'_c} \times 100\%$$

RTT is collected using the "ping" utility that is provided by the operating system. To reduce the negative impact that is brought in by the additional ping traffic, the time interval between consecutive ICMP Echo Request packets of the ping utility is set to 1 second. This is much longer than the 41.5 ms time interval between consecutive game packets that are sent from a GC to the GS. This interval is not too large either. So it can accurately capture the RTT encountered by the game traffic.

In order to determine whether the RTT should be obtained from one GC or more than one GC, and in case of the former, which GC should be chosen, I first performed a set of experiments all with 8 GCs (the maximum number of GCs used in this design), and ping the GS from each of those GCs. The standard deviation of the RTT observed at all GCs is less than 5% of the average round-trip time. So I conclude that RTT is independent of the GC machines. Therefore, I decide to use only one GC to obtain the RTT. When there is more than one GC in an experiment, the GC that last joins the game session is used to collect the RTT.

3.3 Test-Bed

To carry out my study, I have set up a network environment to perform experiments, and to observe the game traffic performance during these experiments. It is a LAN environment using the IEEE 802.11g standard for wireless communication. A number of machines communicate through this network. In this research, I consider regular PCs or laptops equipped with wireless LAN Network Interface Cards (NICs). I do not use smart phones, PDAs, or portable game consoles because currently they are in general not used for high-end 3D FPS games yet. (With the advances of cellular phone processors and flash memories, it is expected that smart phones and PDAs will be able to support sophisticated 3D games in the near future. Then, my study could be extended to those devices as future work.) In real world, there is always some other network traffic when a game is played, so I have included background traffic besides game traffic in my test-bed environment.

My test-bed, depicted in Figure 3.1, has eleven machines, which fall into four different categories:

1. A dedicated GS, – it has a Pentium III 1.7 GHz CPU, 512 MB RAM, and 60 GB hard disk;

2. Eight GCs – each has a Pentium III 733 MHz CPU, 256 MB RAM, and 13 GB hard disk;

3. A background traffic server (BS) – it has a Pentium III 1.7 GHz CPU, 1 GB RAM, and 40 GB hard disk;

4. A background traffic client (BC) – it has a Pentium III 733 MHz CPU, 256 MB

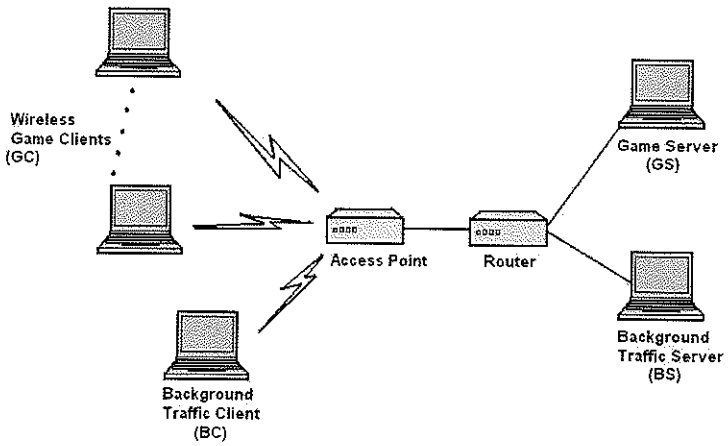

Figure 3.1: Test-bed

RAM, and 13 GB hard disk.

The GCs and the BC are on a wireless network, while the GS and the BS are on a wired network. The GC machines are equipped with Linksys WMP54G Wireless-G PCI adapters. The BC machine is equipped with a D-Link AirPlus G High-Speed 2.4GHz DWL-G510 wireless PCI adapter. Both GC and BC machines are associated with a wireless AP (CISCO AIRONET 1200 series, model No.: AIR-AP1231G-A K9). The advantage of using networking equipments from different vendors is that my test-bed might closely reflect real-life scenarios. On the wired portion of the test-bed, the GS and the BS are connected with the wireless AP via a U.S. Robotics 8054 router. All 11 machines are installed with the Linux operating system (Fedora

3, Kernel 2.6.9-1.667). The entire test-bed was set up in an office environment in a one-floor building, where there are offices and cubicles, all on the same ground floor.

The wireless AP was configured using its default settings except that the access control by MAC addresses on the AP was turned on and the broadcasting SSID was disabled. It is worth noting that besides my Wi-Fi network, there are two other Wi-Fi networks that are in operation in the same building. I used the wireless network discovery application provided by Microsoft Windows XP operating system to measure the signal strength of each Wi-Fi network as seen from my test-bed. The result shows that, my Wi-Fi network is rated 10 out of 10, and those two other Wi-Fi networks were rated 3 to 5 out of 10. To further reduce the interference from those two networks, I configured my AP to make use of the least busy channel and performed the experiments in evenings and weekends. I consider such an environment adequate for my purpose because in a practical wireless gaming environment, co-existence of multiple Wi-Fi networks is likely, and some (low) degree of interference may be present.

At the beginning of an experiment, the GS is first started; it waits for a specified number of GCs to make connections. If there is any background traffic, the background traffic transmission would be started before the GCs are started. Each GC initiates a connection with the GS after it is started. After all GCs have done so, the GS sends packets to the GCs back-to-back in a row at each timeout, following the order in which the connections were first initiated. At the same time, the GCs send packets to the GS periodically without concerns about the other GCs. The background traffic applications are terminated after the game session completes.

3.4 Traffic Generator

Network traffic models define the patterns of observed network usage showing the packet activities for specific network applications. They are usually used in simulations to represent typical network workload by the specified applications. Researchers can then study, test or evaluate the capability of a designed environment to support those applications, or estimate network throughput by running relevant network traffic models in a simulation. A network traffic model emulates the network behavior of every station in the network environment for a selected application. It describes the distribution of the packet size and the packet inter-arrival time from one station to another station, i.e., the stochastic nature of the traffic [31, 44].

In this study, I implemented and made use of emulated FPS game traffic, as well as some emulated background traffic. Using emulated game traffic is easier to control and costs much less than having human players. So I implement a game traffic model to emulate game traffic. In addition, because of resource limitation, I implement a streaming video traffic generator so that I can use one pair of background traffic server and client to emulate the scenarios in which a number of video streams are transferred in the air. Using those two traffic generators, all foreground and background traffic are statistically identical among experiments.

In order to compare the experimental results under various network conditions,

3.4.1 Game Traffic Emulator

Interactive game traffic has different network traffic characteristics compared with other real-time applications, such as video conferences and IP-telephony. Game com-

munications are highly interactive but the transmitted packets are smaller than that in video conferences and IP-telephony. Currently, only a few FPS games' traffic models have been built. They are traffic models for the games Quake1 and Quake2 by Borella [4]; Quake3 traffic models by Lang et al. [25]; and Half-Life traffic models by Lang et al. [24] and Färber [13]. Among those games, Half-Life is the newest, and is currently the most popular FPS game. Therefore, I decide to implement a Half-Life traffic model.

In both of the existing Half-Life traffic models [13, 24], messages are sent between every individual GC and the GS. Färber [13] assumed the independence (i) among clients' behaviors, (ii) between the server traffic per client and the number of clients, and (iii) between the client traffic and the corresponding server traffic. In this model, the GS-to-GC inter-arrival time and the packet size of both directions follow an extreme distribution with different parameters, and the GC-to-GS inter-arrival time is deterministic and is fixed at 40 ms. (An extreme distribution is the distribution of the extreme order statistic [42]. It has a shape that is similar to log-normal distribution.)

In the traffic model by Lang et al. [24], the server-to-client packet inter-arrival time is deterministic: 50% at 50 ms and 50% at 70 ms, with 50ms and 70ms strictly alternating. The GS-to-GC packet size follows a log-normal distribution with different parameters for the different maps being played on. The GC-to-GS packet inter-arrival time is also bimodal but differs based on the client's graphic rendering methods. For those client machines using OpenGL Application Programming Interface (API) [36], the inter-arrival times are 50% at 33 ms and 50% at 50 ms with 33ms and 50ms strictly alternating; for Software users, the inter-arrival time is 41 ms. The GC-to-GS

packet size fits a normal distribution with parameters: mean $\mu = 71.57$ and standard deviation $\sigma^2 = 6.84$ in bytes.

Although these two traffic models are not exactly the same, the probability distribution and the distribution parameters used are very similar in shape, and close in bounds. Both models would likely generate very similar network traffic. For my experiments, I use the latest model produced by Lang et al. [24]. I have implemented this model using the C programming language on the Linux operating system. In this game traffic emulator, I have chosen the map that has the largest average packet size, thus a mean of 203 bytes and a standard deviation of 0.31 bytes for the GS to GC direction are used.

3.4.2 Streaming Video Traffic Generator

Currently, most video streaming applications use UDP protocol to transfer their packets. Using UDP, the loss of any single packet does not affect the sending rate of the rest of the video. To better represent the current state of this field, I have implemented the streaming video traffic generator using UDP as well.

This generator generates video frames of various sizes according to video trace data provided by Fitzek [15]. Multiple video streams are included in my experiments. For each video stream, one video frame is sent every 40 ms. Based on the average bit-rate of each trace, the number of video streams is varied to produce various levels of background video traffic. Based on the average video bit rates, in my experiments, the number of video streams coincides with the expected video load level in Mbps. Thus, there are 4 video streams for 4Mbps video stream load, and 16 streams for 16Mbps

load. A total of seven video trace files are used in this study, and their bit rates range from 850Kbps to 1.2Mbps. The videos are categorized as TV shows, movies, sports programs, and Cam video streams. For each *Video* level, I try to include video files from all categories to represent the situations in which network users have various interest on videos, and thus, may generate different kinds of traffic.

In each experiment, all video streams are started in the first 40 ms, within which the actual start times are random. Similar to the FTP traffic, video streaming traffic is sent from the BS to the BC. This is under the assumption that users in a Wi-Fi environment may be more likely to act as content consumers rather than content producers.

3.5 The Duration and Number of Replications

Each experiment is performed for a duration of 8 minutes. This length is considered representative in a typical FPS game session [6].

In order to reduce the impact of un-controllable parameters, I repeat each experiment 6 times. This number of replications is determined by calculating a confidence interval [26]. A confidence interval estimates how well a given set of independent sample data can represent the un-observed population. It gives the probability that a newly-observed result will have the same value as those in the sample data. I define a typical scenario in my experiments as: $NC = 4$, $Video$ =16Mbps, $FTP = Yes$, $WEP = Yes$, and $Distance = Level1$. Then I perform this experiment 6 times, and compute the 95% confidence interval from the obtained results. The calculation results show that the width of the confidence intervals are extremely small when com-

pared to the sample mean, so I only report on the sample mean of my performance metrics.

3.6 The 2^k Factorial Experimental Design Studies

As described in Section 3.1, I select a set of factors that may cause low bandwidth, high latency, and high packet loss rate in wireless networks.

To simplify the experimentation by omitting unimportant factors, I need to narrow down the initial set of factors to a few major ones. So I perform a 2^k factorial experimental design study; estimate the importance of each factor; and identify the major factors.

Other than a full factorial experimental design, the purpose of a 2^k factorial experimental design is to use the least number of experiments to determine the effect of k factors [26]. In such a design, each factor has two levels, i.e. values. These two levels are an upper-bound and a lower-bound level, which represent the highest and the lowest values of the factor. This determines if a factor affects the game traffic performance significantly when it reaches its limit.

In this 2^k design study, the defined two levels of initial factors are as follows:

Factor-I: *NC*. The smallest number of game players is 1, and the maximum number of players for game Half-life is 32. However, because of the resource limitation, the maximum number of wireless clients I can reach is 8. So this factor has a range from 1 to 8. Therefore, the two levels used in 2^k factorial experimental design are 1 and 8.

Factor-II: *Video.* In practice, due to protocol overhead, the maximum throughput that an application can achieve in an 802.11 network is much lower than the protocol's theoretical bit-rate [5, 7]. In fact, 802.11g network-level performance is well below 50% of its maximum data rate of 54Mbps at the physical layer [43]. Consequently, I choose the two levels as following: No video streaming and 22Mbps video streaming traffic, where 22Mbps represents a heavy load condition on an 802.11g network.

Factor-III: *FTP.* The two levels are no FTP and one FTP session respectively. For the second level of this factor, because an FTP session that transfers a very large file may absorb all available bandwidth in the network, one FTP session is adequate to demonstrate the impact of FTP applications. For this level, I activate a FTP client application that came with Fedora, called "ncftpget", at the BC. It starts to download a large file from the BS before the game session starts. This application will be terminated after the game session completes.

Factor-IV: *WEP.* The two levels are with and without the use of WEP encryption. In the second level, the Wi-Fi network uses a 128 bit shared key with WEP encryption.

Factor-V: *Distance.* At the first glance, for each level of the distance factor, all GCs are located at the same distance from the AP. But in the real world, wireless clients should have varying distance from the AP. To facilitate the investigation of the distance factor, I consider the following two cases.

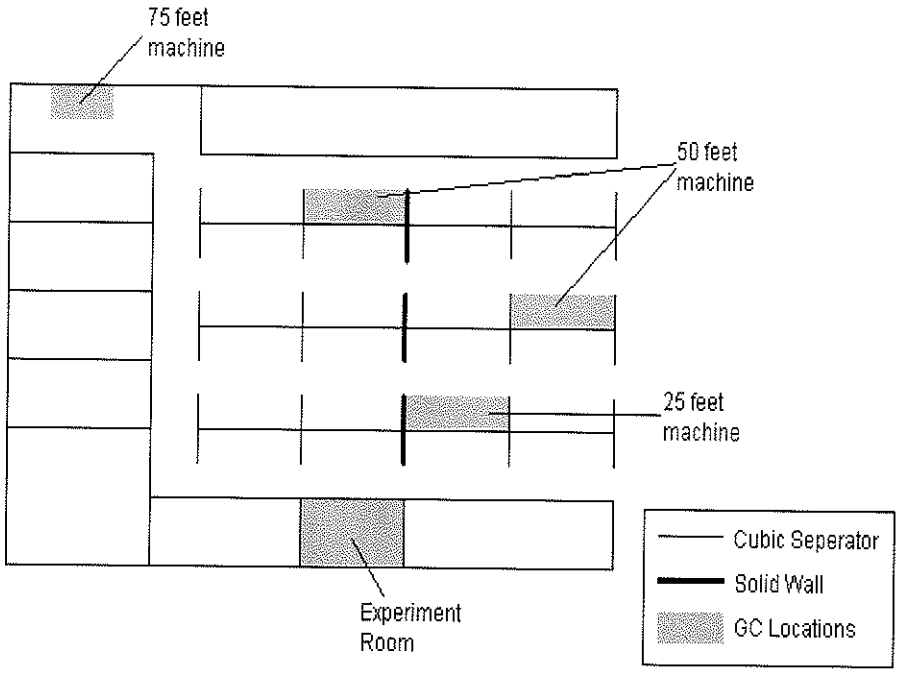

Figure 3.2: Distance Factor Evaluation Environment

In Case I, I have one game client (NC=1) and I vary the distance between two levels. At distance level 1, the GC is placed in an "experiment room" where the AP is located, and the distance to the AP is around 10 feet. (See Figure 3.2 for the layout of the office environment where the experiments were conducted.) At distance level 2, the GC is placed in a room (at the top left corner in Figure 3.2) that is around 75 feet away from the AP. Case I is to see the impact of distance when there is only one game client in the Wi-Fi network. In Case II, I increase the number of game clients to 8 (i.e., NC=8). Similar to Case I, two distance levels are under consideration. At distance level 1, all 8 GCs are located in the experiment

room. These GCs form a circle with a diameter of around 10 feet and the AP is positioned in the center of the circle. There are no physical obstacles between the AP and the GCs. At distance level 2, four GCs are remained in the experiment room. The other four GCs are spread out in the office environment (see Figure 3.2); one is 25 feet away from the AP, two are 50 feet away, and one is 75 feet away. I attempt to use this placement of GCs to model a Wi-Fi gaming environment where players may sit at different locations within the Wi-Fi network. As shown in Figure 3.2, there are solid walls and cubicle dividers between these four GCs and the AP. Finally, in both cases, the GS, BS, and BC are located in the experiment room.

Since the distance factors are studied in two different ways, their results cannot be combined together for studying the impact of NC. So, I performed three 2^4 factorial experimental designs for the situations that have one client with varying distance, or eight clients with varying distance, or one / eight clients that are all in the same distance, respectively. They are called Design 1, Design 2, and Design 3. In addition, for Design 2, I have measured the RTT from every GC to the GS. Using these results, I further study the latency difference between the average RTT of the four within-10-feet GCs and the 75-feet GC within one game play. This forms the fourth 2^4 factorial experimental design. The fourth design is different from the others in two ways:

1. Design 4 uses only RTT as performance metric. This is due to the limitation of my game traffic generator. In the game traffic model, the state update messages sent from GS to the GCs are in bursts. Because of lacking the implementation

detail of how Half-Life sends the messages from the GS to the GCs, I implement my game traffic generator by sending the GS messages to the GCs one by one in each burst. Then the messages that are sent later will also arrive at the AP queue later, thus, have higher probability to be lost than the messages that are sent earlier. In this case, the loss ratio on each individual GC is not only related to the transmitting procedure, but also related to the sequence in which the messages are sent to the GCs. When the distance factor is used, the impact of distance cannot be separated from the impact of the message sending sequence. Therefore, I decide not to consider loss ratio in this design.

2. In Design 4, the number of GCs is not fixed at one value. The latency result for within-10-feet GCs is the average RTT of all four GCs at that distance, while the 75-feet latency is obtained from one GC only.

Each design has 16 settings. In all the designs, the BC is located within 10 feet from the AP. The factors and their two levels used in the three designs are shown in Table 3.1. Every design yields $2^4 = 16$ settings. Because both Design 1 and Design 2 have half of the settings in the same settings as Design 3, and also because Design 4 uses the results from Design 2, there are a total of $16 + 8 + 8 = 32$ settings for all four designs. As the experiment of each setting has 6 replications, the total number of experiments is $32 \times 6 = 192$.

The obtained results show that, LR_{c2s} never exceeds 1% in my experiment settings (refer to Appendix A). In the rest of this thesis, I only focus on performance metrics LR_{s2c} and RTT.

I calculate the amount of variation explained by each factor and each factor inter-

Table 3.1: Factors in the Four 2^4 Factorial Experimental Designs

Factor	**Design 1**	**Design 2**	**Design 3**	**Design 4**
Distance	Within 10 feet /75 feet	All in 10 feet /Spread	All in 10 feet	Within 10 feet /75 feet
WEP	Open/128 bit shared key	Open/128 bit shared key	Open/128 bit shared key	Open/128 bit shared key
FTP	No/One FTP	No/One FTP	No/One FTP	No/One FTP
Video	No/22Mbps	No/22Mbps	No/22Mbps	No/22Mbps
NC	1	8	1/8	4 and 1

action using the obtained values of performance metrics for each design. Tables 3.2, 3.3, 3.4, and 3.5 list the factors and the factor interaction which have an importance ratio larger than 1% in each of the designs. The full detailed experimental results and the explanations of the variations of each design can be found in Appendix A.

In Design 1, *Video* is the most important factor in terms of LR_{s2c}. *FTP* and the interaction of *Video* and *FTP* have lower but still significant impact. *FTP* is the most important factor in terms of *RTT*. *FTP*+*Video* and *Video* have comparatively much lower impact on *RTT*. All other factors or factor interactions have very small impact in explaining the amount of variation in results. It can be concluded that when the number of game client is one, the important factors are *Video*, *FTP*, and *FTP* + *Video*.

Table 3.2: Results of Design 1

Performance Metrics	*FTP*	*Video*	*FTP* + *Video*
LR_{s2c}	27.75%	45.00%	24.53%
RTT	82.87%	2.23%	11.25%

In Design 2, *FTP* is the most important factor for both performance metrics.

Video is another very important factor in terms of LR_{s2c}. Same as Design 1, all the factors other than *FTP*, *Video*, and the interaction of *Video* and *FTP* explain less than 1% of variation in results, and are thus negligible. It can be concluded that, when the number of game client is eight, the important factors are *Video* and *FTP*.

Table 3.3: Results of Design 2

Performance Metrics	*FTP*	*Video*	*FTP + Video*
LR_{s2c}	65.55%	30.95%	2.96%
RTT	86.21%	5.16%	7.50%

For Design 3, Table 3.4 shows that, *FTP* is the most important factor on both performance metrics. *Video* and *NC* are also important in terms of LR_{s2c}. There are also some comparatively lower impact of *NC + FTP* and *FTP + Video* on LR_{s2c} and impact of *Video* and *FTP + Video* on *RTT*. All the other factors are not important.

Table 3.4: Results of Design 3

Performance Metrics	*FTP*	*Video*	*FTP + Video*	*NC*	*NC + FTP*
LR_{s2c}	33.51%	24.73%	4.62%	24.89%	9.70%
RTT	86.51%	2.23%	10.30%		

In Design 4, Table 3.5 shows that, *FTP* dominates the average *RTT*. *Video*, *FTP + Video*, and *Distance* have some impact, but are comparatively much lower. All the other factors are not important in this design.

Table 3.5: Results of Design 4

Performance Metrics	*FTP*	*Video*	*FTP + Video*	*Distance*
RTT	87.34%	5.05%	4.19%	1.55%

The above results allow me to remove two factors: *Distance* and WEP. In Design 1 and Design 2, the amount of variation explained by both types of distance factor definitions, and any factor interaction that contains the distance factor, is always less than 1%. Although in Design 4, *Distance* has an impact slightly larger than 1%, it is still very small. This means that the distance factor is not important for this experimental environment setting. From Tables 3.2, 3.3, 3.4, and 3.5, we can also conclude that WEP is not important, since the importance ratio of the authentication factor and any factor interaction that contains the authentication factor also never exceeds 1%. Thus, I conclude that the factors WEP and *Distance* do not have significant impact on the game traffic performance in my target environment.

3.7 The Experiment Settings

After completing the four 2^k factorial experimental designs, I identify three major factors: FTP, NC, and $Video$. For FTP, I decide to stay at two levels: with and without FTP session. For NC and $Video$, as it is not feasible to perform experimentation for every possible values, I select a number of levels for each. NC has levels at 1, 2, 4, 6, and 8. $Video$ has levels at 0, 4, 8, 12, 16, 20, 22, 24, and 26Mbps. $Video$ has finer levels when larger than 20Mbps. This is because RTT and LR_{s2c} have sudden increase when $Video$ exceeds 20Mbps, and I would like to have a clear view on the increasing trend of the performance metrics in this range. Note that in these levels, the highest video load level is 26Mbps. This is higher than the 22Mbps used in the 2^4 designs above. The higher levels are selected to illustrate what would happen when the wireless channel is approaching saturation. On the other hand, in the 2^4

designs, it was expected that the video load level is kept at 22Mbps or lower in order to avoid starvation of other traffic on a shared 802.11g network. The conclusions from 2^4 designs above would remain valid if 26Mbps is used instead.

In this part of experimentation, the unimportant factors, WEP and *Distance*, are fixed at WEP=yes and *Distance*=1.

To summarize, the levels for each factor are listed below:

Factor-I: NC has 5 levels: 1, 2, 4, 6, and 8.

Factor-II: *Video* has 9 levels: no video streams, with 4, 8, 12, 16, 20, 22, 24, and 26Mbps.

Factor-III: FTP still has two levels: no FTP and one FTP session.

Factor-IV: WEP is fixed; encryption with a 128 bit shared key is always used.

Factor-V: *Distance* is also fixed at level one – where all the machines are within 10 feet from the AP.

Each setting still has 6 replications. This part of experimentation yields a total of $5 \times 9 \times 1 \times 1 \times 6 = 270$ experiments.

Chapter 4

Experimental Results

In this chapter, I report on the observed results from my experiments, and present the major findings of this study. The results show that the major factors impact packet loss ratio and latency differently. To clarify the factor's impact on game traffic performance, I separate the loss ratio analysis and the latency analysis, and present them in two sections respectively.

4.1 Impact on Packet Loss Ratio

In this section, I describe how the factors impact LR_{s2c}, the packet loss ratio for the GS to GC game traffic. The 2^k experimental design shows that FTP download and UDP video streaming not only individually affect the performance, but their interaction also affects the performance. To better describe their impact on performance, I report the experimental results in two subsections: "without" and "with" FTP. "Without" indicates that other than the game traffic, all other background

traffic, if there is any, is from the UDP-based video streaming; there is no FTP background traffic. "With" represents the case where background traffic includes both FTP traffic and UDP-based video streaming traffic. Last, I describe the impact of NC, the number of game clients on performance.

4.1.1 Without FTP

Figure 4.1 plots the trend of the game traffic loss ratio along the increase of the $Video$ for various values of NC.

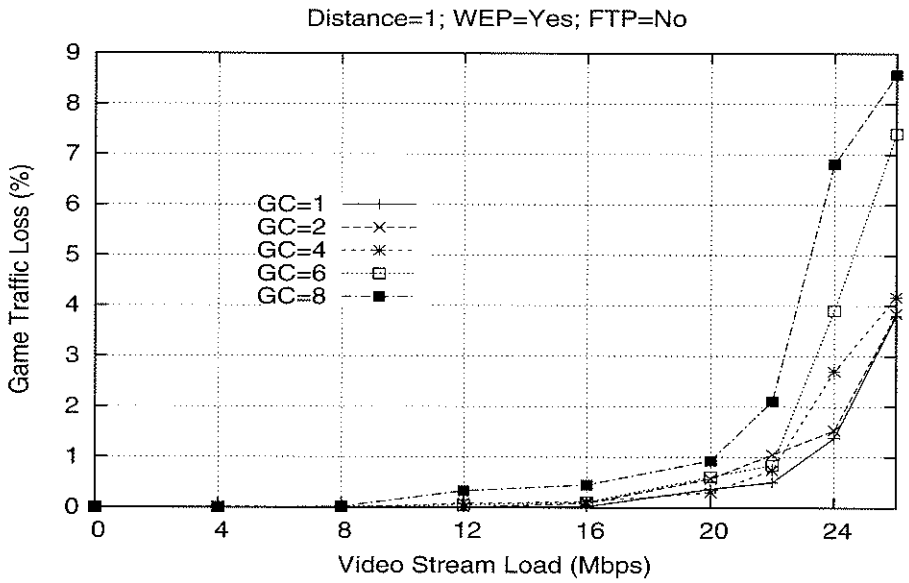

Figure 4.1: Loss Ratio vs. Amount of Video Stream

The background traffic only contains video streams. It can be observed that when $Video \leq$ 8Mbps, there is no loss on the network; between 8 and 16Mbps, the loss is lower than 1%; when $Video \geq$ 20Mbps, the loss ratio has a sharp increase. This

implies $Video$ may have an important influence on FPS game loss ratio even without FTP traffic. From this figure, we can also see that, when $Video \geq$ 8Mbps, it becomes more and more clear that a higher NC leads to a higher loss ratio. This implies that NC has an impact on loss. The impact of NC is explained in Section 4.1.3.

The reason why a higher video load leads to a higher loss is that a higher video load produces more wireless traffic. As described in Section 3.4.2, one video stream generates 25 frames per second. At 16Mbps video load, there are 16 video streams, which produce 400 frames per second. Accounting for video frame fragmentation, packet arrival rate is higher than the frame arrival rate. When packets arrive at the AP, because of the stochastic nature of traffic, a queue may form, when packets arrive at a higher bit-rate than the AP's transmission rate, the queue may quickly fill. When the buffer becomes full while more packets are arriving, the packets that arrive later will be dropped. This results in packet loss. Video streaming applications use the UDP protocol, in which the sender cannot see the congestion in the network. Thus the sender continues sending packets regardless of the network load and capability. When the number of video streams and/or video streams' amount become larger, more packets will be sent to the AP, resulting in increased loss ratio.

4.1.2 With FTP

This subsection studies the case when there is FTP background traffic while a game session is on-going. It shows that the inclusion of FTP significantly affects the game traffic loss ratio in a Wi-Fi network.

Figures 4.2 and 4.3 plot the game traffic loss ratio "with" FTP for $NC = 1$ and

$NC = 8$, respectively. To ease comparison, in each graph, the results when "without" FTP are also included. It can be observed that in both graphs, the inclusion of FTP traffic greatly lowers the loss ratio performance of game traffic. For $NC = 1$, the largest difference between "with" and "without" FTP traffic approaches 3%. For $NC = 8$, this difference is close to 5%. The difference increases the fastest when video stream size increases from 0 to 4Mbps. Then this difference remains the same until the video stream size reaches 20Mbps. When $Video > 20$Mbps, both loss rates increase greatly. Also, the loss for "without" FTP increases faster than that for "with" FTP. In Figure 4.2, when the video stream size reaches 26Mbps, the two situations have very similar loss ratios. When $Video = 0$, both cases have their loss close to 0%.

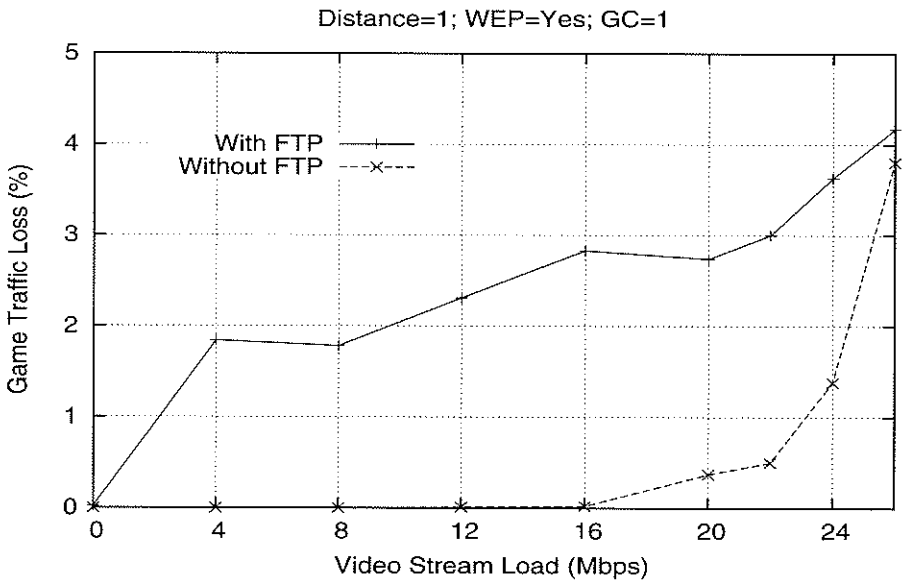

Figure 4.2: FTP Impact on Loss Ratio for 1 GC

The performance degradation when there is FTP traffic can be explained by the

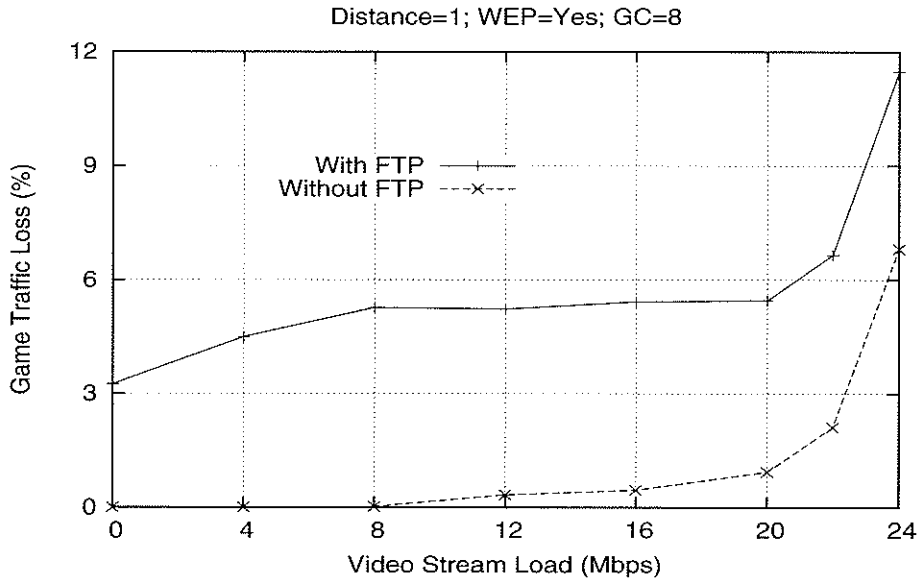

Figure 4.3: FTP Impact on Loss Ratio for 8 GC

nature of FTP applications. FTP traffic is delivered using the TCP transmission protocol at the transport layer. TCP protocol has a congestion control mechanism for supporting reliable data transmission. Every TCP segment sent has a sequence number attached, and requires acknowledgement (ack) from the receiver to the sender [33]. The receiver acknowledges the arrival of every segment if all the previous segments have been received correctly. If a segment arrives while some previous ones have not, the receiver sends a duplicated ack acknowledging the last segment that it received properly. In this way, the sender is able to determine the possible loss, and then reduce the sending speed.

The sender controls the sending speed by maintaining a congestion window, which decides the total amount of unacknowledged data allowed at any time. This conges-

tion window is initially set at a very small value, i.e. 1 or 2 Maximum Segment Size (MSS) in bytes, which is the size of the largest segment that can be transferred over the network. Then the congestion window increases using a slow start method, which doubles the congestion window size each time a non-duplicated ack is received [39, 1]. If the sender (i) reaches a timeout while waiting for the ack of a packet, or (ii) receives three duplicated acks, the sender re-transmits the segment following the last segment that has been acknowledged. In case (i), the congestion window is decreased to the initial size. In case (ii), the congestion window shrinks to half. In both cases, the sender sets up a slow start threshold (*ssthresh*) at the half value of the congestion window size before the shrink happens. The congestion window is in slow start process when it is smaller than the *ssthresh*, and increases by one MSS after every RTT, when it is larger than the *ssthresh*. In this way, the FTP application is able to respond to the current network load, thus be able to explore and make use of the maximum available bandwidth in the traffic path.

When a queue (at a router or an AP) inside the network is overflowed, loss events occur, and TCP congestion control is triggered. This implies that TCP may tend to keep the queue length to be within the buffer size limit. At the same time, because TCP endeavors to make use of the available bandwidth as much as possible, it tends to keep the bottleneck queue relatively full. In my test-bed network, the major queue in the network is the one at the AP along the server to client direction. When the video load is low to moderate and there is no FTP traffic, the queue is relatively short, yielding good RTT performance. With FTP traffic, however, the queue is relatively full with the FTP traffic. If a first-come-first-served (FCFS) scheduling is used at this

queue, the game traffic will encounter a much longer queue, when compared to the case of no FTP traffic, resulting in worse performance. When the video traffic load is high enough (e.g., above 24Mbps), even without FTP traffic, the wireless channel is approaching saturation and the queue starts to overflow. In this case, it is observed that the performance is equally bad regardless of there is FTP traffic or not. These observations indicate that QoS strategies may be needed in order to well support game traffic when there is FTP background traffic in the network.

In addition, from Figure 4.2, we can see that, for both levels of FTP, the higher the video traffic load, the higher the loss ratio. Without FTP traffic, the loss ratio has a sharp increase between the video load levels of 24 and 26Mbps; with FTP traffic, this degradation of loss ratio is more gradual. This is because of the "damping" effect of TCP traffic, which affects the dynamics of the queue length. Specifically, when there is only UDP traffic, there would be a sharp increase in queue length as the channel approaches saturation. But with FTP traffic, the queue tends to stay relatively full all the time and such a sharp increase in queue length at high video traffic load would not occur.

Figure 4.4 displays the loss ratio results against the amount of $Video$ for every level of NC when with FTP background traffic. It can be observed that the trend in loss ratio results for all NC values is the same as that in Figures 4.2 and 4.3. When there is no video traffic, the loss ratio is comparatively low – close to 0% for $NC \leq 6$. When $Video$ increases from 0 to 4Mbps, the loss rate increases substantially. From $Video = 4$Mbps to $Video = 20$Mbps, the loss increases in a very slow manner. When $Video > 20$Mbps, the loss ratio increases exponentially. Same as explained

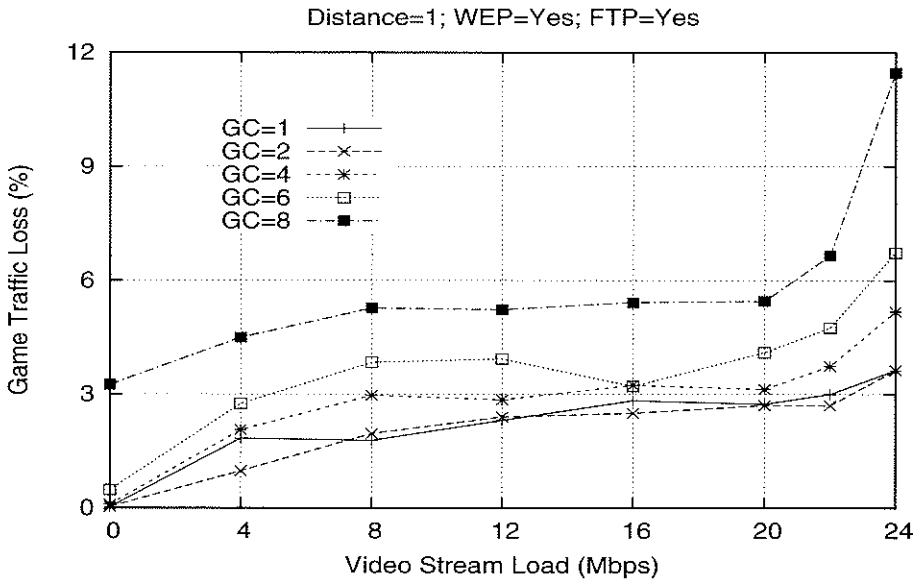

Figure 4.4: Loss Ratio vs. FTP+UDP

above, when *Video* is increased from 0 to 4Mbps, the loss starts to happen since the AP queue is already quite full. While *Video* increases from 4Mbps to 20Mbps, the TCP traffic backs off so the wireless channel capacity may still be in pace for data and the increase in loss can still be in a very small range. However, when *Video* > 20Mbps, the AP cannot keep up with the ever increasing load even when there is only UDP traffic; at this point, for FTP traffic, the TCP congestion window stays at its minimum size. Then, the loss increases exponentially just like when without TCP.

4.1.3 The Impact of the Number of Game Clients

I describe the effect of NC, both for the sole impact of *Video*, and for the combination of *Video* and FTP. The results for LR_{s2c} when without and with FTP are

shown in Figures 4.5 and 4.6 respectively.

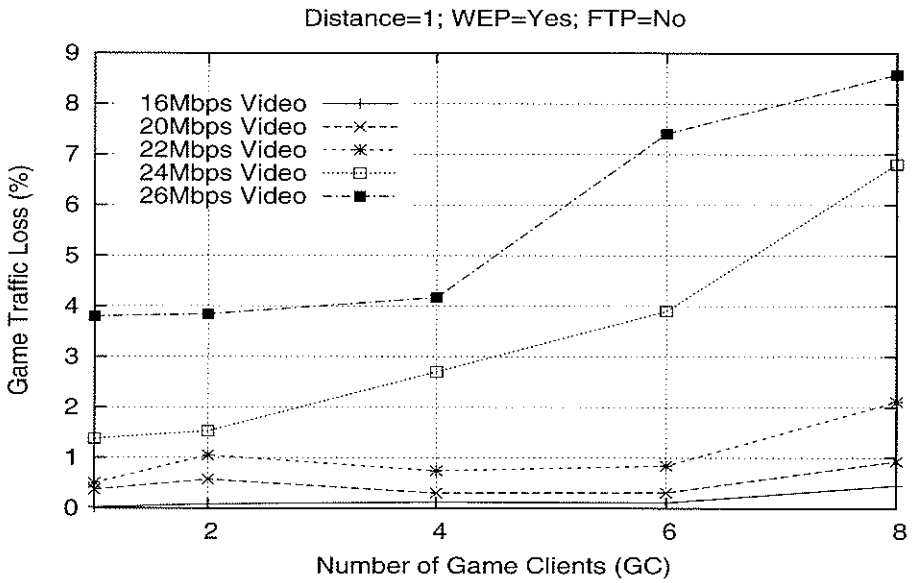

Figure 4.5: Loss Ratio Time vs. Number of GC (UDP only)

In both figures, NC is varied from 1 to 8. Consider the scenario when there is no FTP traffic. It can be observed that NC has a large impact on the loss ratio, especially at heavy video traffic load. This phenomenon may be explained as follows. In a game session, at regular time intervals, the server sends state update messages back-to-back to all GCs. These updates form a burst – a train of packets. The larger the NC, the longer the burst. When the video traffic load is heavy, the queue at the AP is almost full, a longer burst would result in higher loss of game packets. Consider next the presence of FTP traffic. The number of GCs has a large impact on the loss ratio performance of game traffic. The more GCs, the higher the loss ratio. This can be explained by the relationship between the loss ratio and the state update burst

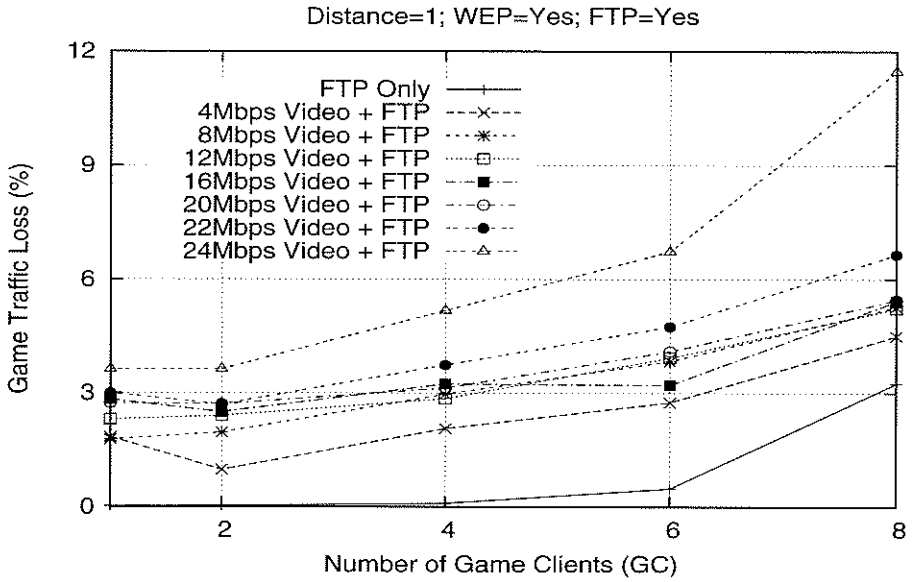

Figure 4.6: Loss Ratio Time vs. Number of GC (with FTP)

size, as described above for the case without FTP traffic. Particularly, with FTP traffic, the buffer at AP is kept relatively full. When NC is increased, a longer burst of game packets arrives at the queue, resulting in a higher probability of having a buffer overflow.

I conclude that the number of GCs largely affects the loss ratio performance experienced by game traffic. When there is no FTP traffic, this holds only when the video traffic load is high. With FTP traffic, this holds for all video load levels.

4.2 Impact on Round-Trip Delay

Under my experimental settings, the average game traffic RTT observed does not exceed 25 ms. There have been studies on the delay requirement for network delivery

for preserving interactivity in interactive applications. The RTT requirements that these studies focus on are primarily in a range between 100 and 200 ms [22]. In my experiments, all the game clients reside within a local area network, which have much better performance than on a wide area network. Considering that most game players play multi-player online games over the Internet, we should take into account that when the local game players are connected to remote players via the AP, if the delay in the access LAN already reaches 25 ms, the delay quota left for the wide area network delivery for a good game play has to be significantly reduced. Ideally, it would be desirable to have negligible delay at the access LAN. For this reason, analyzing the factors' impact on latency is also very important.

4.2.1 Without FTP

Figure 4.7 displays the average RTT versus the amount of *Video* when only UDP video streams exist in background. It shows that, the average RTT increases slowly when $Video < 20$Mbps, and has a much faster increase after that. This result is consistent with the results shown in Figure 4.1. The RTT is a sum of two one-way delays, where a one-way delay contains processing delay, transmission delay, propagation delay, and the queuing delay at the AP buffer. In my test-bed, the processing delay, transmission delay and propagation delay can be considered fixed. The only varying delay which affects the RTT is the queuing delay at the AP buffer. When *Video* is low, on average, the AP is always able to transmit every packet before the next one comes. Then there is no queuing delay, and RTT stays at the lowest level: less than 2 ms. When the load of video increases, the queuing delay becomes

larger. Consequently, the RTT becomes higher as well.

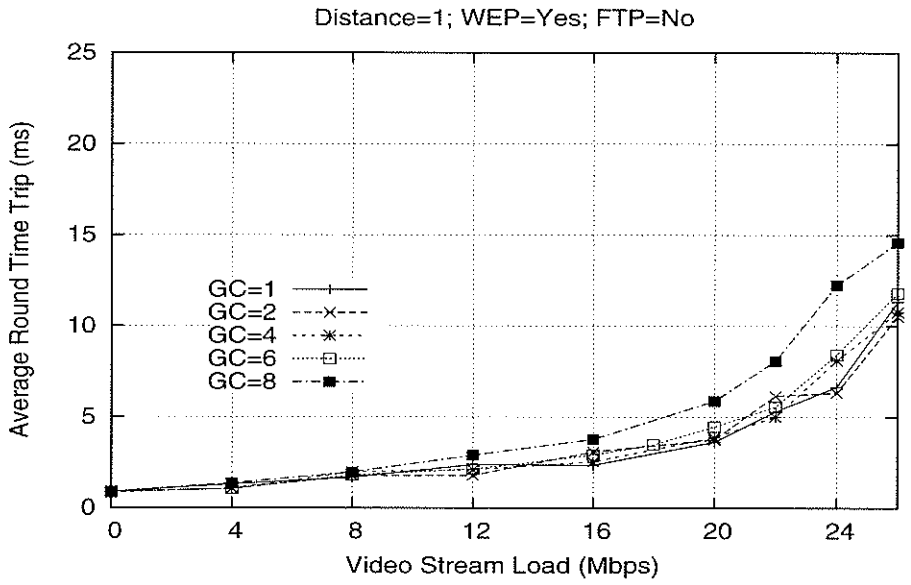

Figure 4.7: Round-Trip Time vs. Amount of UDP

4.2.2 With FTP

Figures 4.8 and 4.9 plot the average RTT "with" and "without" FTP for $NC = 1$ and $NC = 8$, respectively. It can be observed from these two graphs that, similar to loss ratio results, the inclusion of FTP traffic greatly lowers the RTT performance as well. For $NC = 1$, with FTP traffic, when the amount of video traffic varies, RTT ranges from 15 to 25 ms; without FTP traffic, RTT is below 10 ms for most video load levels. When $NC = 8$, with FTP has much longer delay than without. The latency remains in the range of 17 to 22 ms while the video load is varied.

Same as explained in Section 4.1.2, without FTP, when the video load is low

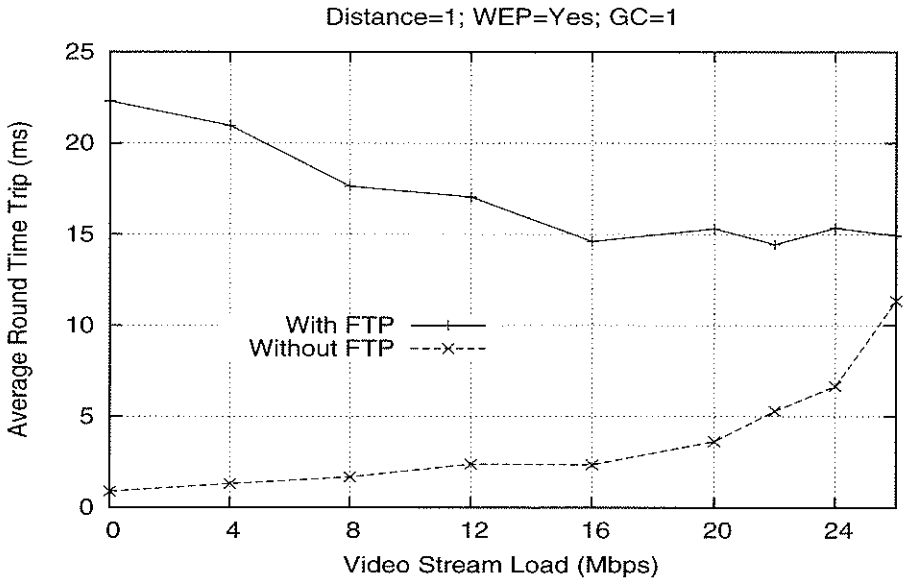

Figure 4.8: FTP Impact on Round-Trip Time for 1 GC

to moderate, the queue is relatively short, thus, the RTT is low. With FTP in the background, the AP queue stays relatively full all the time. Thus the game packets always join the AP queue at the end, and wait a fairly long time before being transmitted. Therefore, the RTT is kept high.

Figure 4.8 also displays an interesting phenomenon. Without FTP traffic, the RTT performance deteriorates as the video traffic load is increased because the average queue length grows as the load is increased. Contrarily, with FTP traffic, the RTT performance actually becomes better as the *Video* increases. This is probably because as the video load becomes higher, with TCP congestion control, the FTP traffic backs off more significantly, leading to better RTT performance.

The average RTT versus the amount of video traffic when with FTP is plotted in

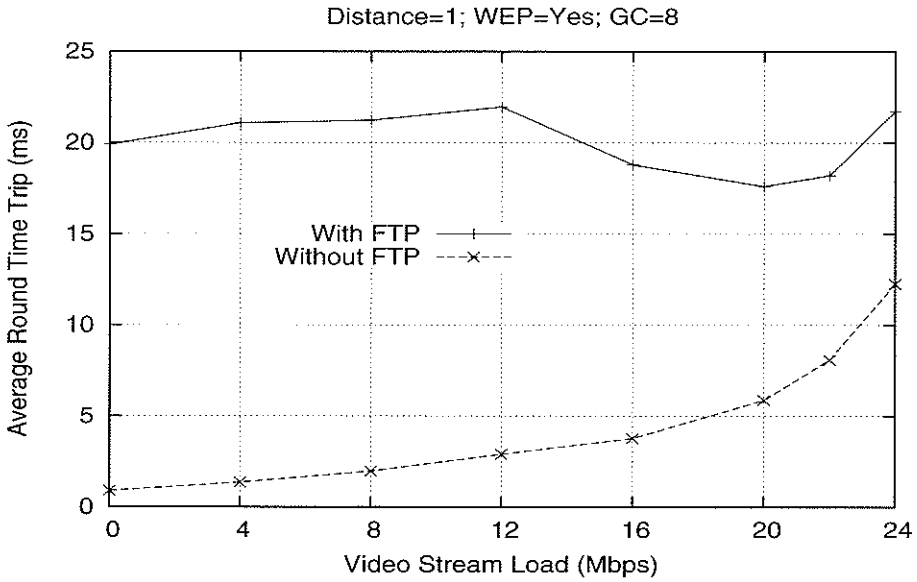

Figure 4.9: FTP Impact on Round-Trip Time for 8 GC

Figure 4.10. In addition to the corresponding results for $NC = 1$ and $NC = 8$ in Figures 4.8 and 4.9, NC values of 2, 4, and 6 are included. Results in this figure are consistent with both Figures 4.8 and 4.9. With FTP traffic in the background, the impact of $Video$ is not very important. However, when the $Video$ increases from 0 to 20Mbps, RTT has a small reduction. After that, the average RTT starts to increase again.

4.2.3 The Impact of the Number of Game Clients

Figure 4.11 displays the impact of NC on average RTT when without FTP while the video stream size is varied. In this case, the number of game clients has a large impact on performance only when at heavy load and when the number of GC increases

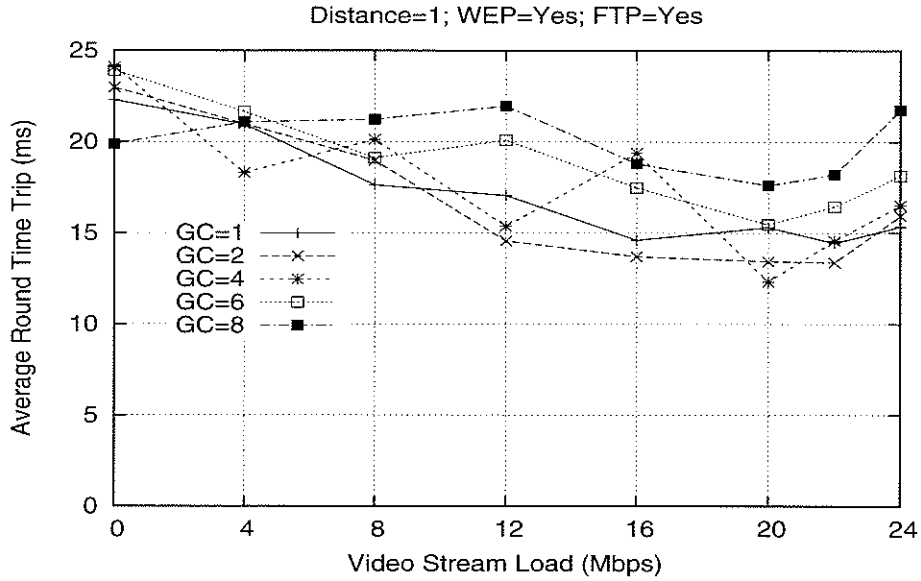

Figure 4.10: Round-Trip Time vs. FTP+UDP

to 8. This is because the traffic generated by one GC is very low, the traffic generated by adding a few more GCs is still not high. Thus the load added to the AP buffer is also very small. Therefore, the increase of NC should not increase the queuing delay. However, when more wireless clients join the Wi-Fi network, there is less chance for a wireless device to obtain the media for transmission. Thus, the waiting time before service increases. For this reason, the number of wireless game clients increases the transmission delay. As a consequence, the RTT increases.

Figure 4.12 shows the NC's impact on average RTT when with FTP as the level of *Video* is varied. In this case, the RTT of game traffic is kept at a high level regardless of the number of GCs. This is probably because when with FTP traffic, the queue is kept at an "almost full" level by the TCP congestion control. With FCFS channel

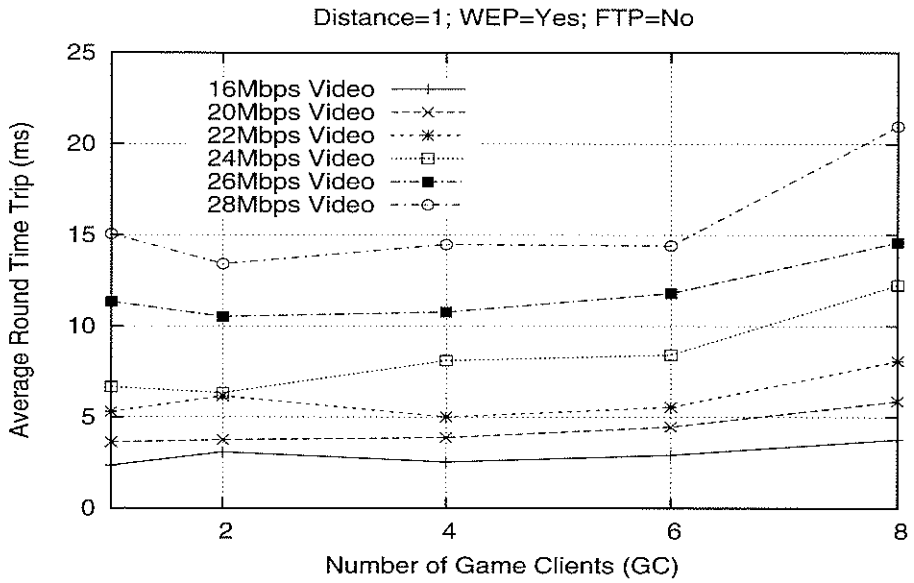

Figure 4.11: Round-Trip Time vs. Number of GC (UDP only)

scheduling, queuing delay encountered by the game traffic is about the same, which corresponds to the time it takes to empty (or serve) an almost full queue.

4.3 Summary of Observations and Findings

I summarize the observations and findings from these experiments in this section. I observe that:

- When $Video \leq$ 20Mbps, both performance metrics RTT and LR_{s2c} increases gradually. When $Video \geq$ 20Mbps, the two performance metrics increase exponentially.

- FTP traffic has the greatest impact on both performance metrics. RTT when

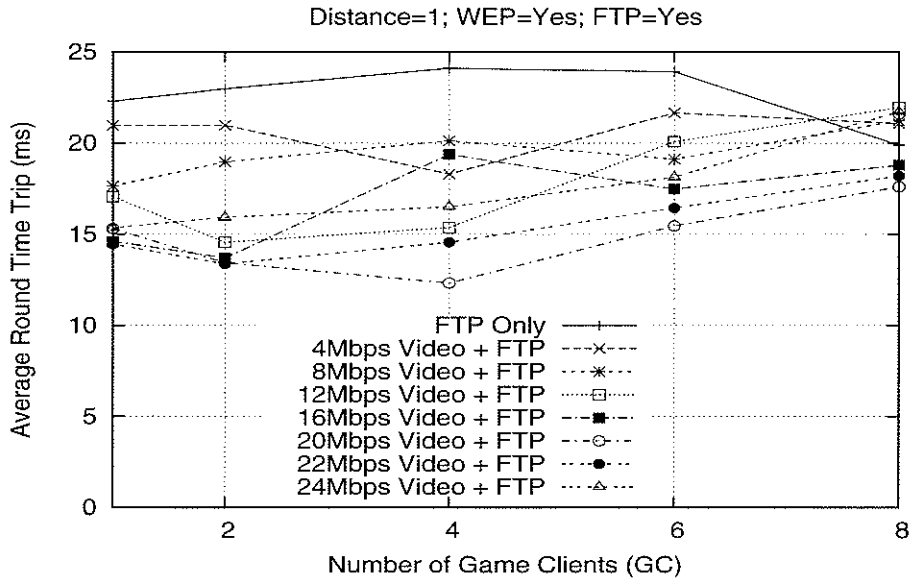

Figure 4.12: Round-Trip Time vs. Number of GC (with FTP)

with FTP is much higher than when without FTP. The results also show that when with FTP and when $Video$ is less than 20Mbps, the increase of $Video$ leads to smaller RTT, probably because that TCP congestion control is over-reacting. In terms of LR_{s2c}, when with FTP but without $Video$, loss is close to 0%. The inclusion of video streams results in a fast increase of LR_{s2c}. However, when $Video \geq$ 20Mbps, loss with FTP increases comparatively much slower than without FTP. Then, the loss ratios will ultimately become very close.

- Comparing to FTP and $Video$, NC has smaller impact on game performance. Nevertheless, we still can see that the increase of NC results in higher LR_{s2c} and RTT. This phenomenon becomes more clear when the background traffic is heavy (either with FTP, or when $Video \geq$ 20Mbps).

From these experimentation results, I conclude that video streaming traffic, FTP traffic, and the number of GC have a substantial impact on game performance. Among the three major factors, FTP traffic has the greatest impact on both performance metrics.

In this chapter, the results are all displayed in figures with line-points. Their error bars are plotted in Appendix B. From the error bar figures shown in Appendix B, we can see that the variations of the performance metrics are less than one third of the performance metrics values. With the error bars, the performance metrics still follow the trends which have been explained in this chapter, thus those variations do not affect my conclusions.

It should be noted that in the above, I have focused on the average RTT performance, this is because current game performance requirement studies are focused on the average round-trip delay. However, in my experiments, although the average RTT does not exceed 25 ms, the maximum RTT observed in my experiments can be as high as 200 ms. We currently do not know how much this maximum RTT may impact game performance. This may be studies in the future.

In addition, some further results on the 2^k factorial experimental design and the background traffic throughput results when with FTP are reported in the Appendix A and C, respectively. These results can be used as reference for future research.

Chapter 5

Conclusions and Future Work

In this thesis, using an experimental approach, I have evaluated the impact of a number of selected factors on the game traffic performance experienced by game traffic on an IEEE 802.11g network. The factors experimented include the distance between a game client and the wireless AP, the enabling of data encryption, the inclusion of FTP and video streaming background traffic, and the number of game clients. The performance metrics used in this study are game packets RTT and game packets loss ratio. I have found that FTP and video streaming traffic significantly affect the performance. The number of game clients is important when the aggregated background traffic load is high. In particular, it has a higher impact on loss ratio than on RTT. Comparatively, distance and WEP have minimal effect. Based on these observations, in the next section, I suggest possible QoS strategies at the application and MAC layers respectively. These strategies may be used to improve the support of a Wi-Fi network to a MOG.

5.1 QoS Strategies in a Wi-Fi Gaming Environment

It was suggested in the last chapter that QoS strategies may be needed to better support games in a Wi-Fi environment that may be shared by FTP and video background traffic. Assuming no change is to be made at the transport layer, i.e., with TCP and UDP, in this section, I discuss strategies at the application layer and inside the network respectively.

At the application layer, a possible strategy is to throttle both FTP and video background traffic when game sessions are ongoing. This may imply certain restrictions on the applications that other wireless clients sharing the same Wi-Fi network with the game clients can run. From my study, I have observed that both FTP and video traffic significantly affect the game traffic performance, especially when the aggregated load level is high; therefore, throttling their traffic may leave ample bandwidth on a Wi-Fi network to game traffic and better support the game sessions.

Inside a Wi-Fi network, QoS strategies refer to the queue management and channel scheduling algorithms inside the AP at the MAC layer. A possible strategy is to employ priority based scheduling in which a higher priority is given to the game traffic. In terms of implementation, a separate queue may be maintained for the game traffic, while the other types of traffic share another queue. Whenever the channel becomes idle, the channel scheduler serves the traffic in the game traffic queue, if it is non-empty; otherwise, the other queue is attended. Within each of these two queues, FCFS scheduling may suffice. Prioritizing game traffic may pose a problem when

the total bandwidth consumed by the game traffic is high. It may lead to the game traffic starving the other traffic. However, FPS games have been identified to have low bit-rate. Thus, this strategy can be viable with moderate game traffic load.

Comparing the above two strategies, the one at the application layer does not require any modification at the AP. The downside is that it may result in a low network utilization. The strategy at the MAC layer, on the other hand, requires modification at the AP, but may yield higher network utilization. Further investigation of these two strategies can be part of the future work.

5.2 Future Work

This research was limited by the available resources in terms of the number of playing devices and the physical space. In future work, this study can be extended by using more wireless game clients and more background traffic clients. Also, larger distance may be experimented, e.g. 100 feet, 200 feet, etc.

In Chapter 4, it is noticed that, the co-existence of UDP and TCP traffic leads to lower game traffic latency when the video load is increased. The interaction between UDP and TCP, and its effect on application performance can be studied in more detail in future researches.

Appendix A

2^k Factorial Experimental Design Full Results

This appendix reports the detailed results obtained from each of the 2^k factorial experimental designs. These results include the average RTT, LR_{s2c}, and LR_{c2s} from each experimentation setting, and the calculation of the amount of variation in results that are explained by each individual factor or factor interaction. The results of each design is in one separate section.

A.1 Results of Design 1

Table A.1 shows the raw results of the experiments in Design 1. Table A.2 shows the amount of variation in results that is explained by each individual factor or factor interaction.

Table A.1: Results of Design 1

Distance	*WEP*	*FTP*	*Video*(**Mbps**)	LR_{s2c}	LR_{c2s}	*RTT*
Within 10 feet	Open	no	no	0.00%	0.00%	0.717
			22	0.36%	0.00%	4.000
		yes	no	0.06%	0.00%	22.675
			22	2.63%	0.00%	13.821
	128 bit WEP	no	no	0.00%	0.00%	0.698
			22	0.50%	0.00%	5.296
		yes	no	0.05%	0.00%	30.207
			22	3.00%	0.00%	14.451
75 feet	Open	no	no	0.00%	0.00%	0.879
			22	0.39%	0.00%	4.399
		yes	no	0.02%	0.00%	22.291
			22	2.59%	0.00%	16.733
	128 bit WEP	no	no	0.00%	0.00%	0.811
			22	0.48%	0.00%	4.203
		yes	no	0.18%	0.00%	26.913
			22	3.58%	0.00%	18.579

Table A.2: Amount of Variation Explained by Each Factor in Design 1

Factor	LR_{s2c}	*RTT*
Distance	0.11%	0.03%
WEP	0.78%	0.97%
FTP	**27.75%**	**82.87%**
Video	**45.00%**	**2.23%**
Distance + *WEP*	0.14%	0.04%
Distance + *FTP*	0.10%	0.06%
Distance + *Video*	0.05%	0.38%
WEP + *FTP*	0.42%	0.73%
WEP + *Video*	0.53%	0.29%
FTP + *Video*	**24.53%**	**11.25%**
Distance + *WEP* + *FTP*	0.18%	0.00%
Distance + *WEP* + *Video*	0.04%	0.03%
Distance + *FTP* + *Video*	0.05%	0.54%
WEP + *FTP* + *Video*	0.25%	0.47%
Distance + *WEP* + *FTP* + *Video*	0.06%	0.12%

A.2 Results of Design 2

Tables A.3 and A.4 show respectively the raw experimental results and the amount of variation in results that is explained by each individual factor or factor interaction of Design 2.

Table A.3: Results of Design 2

Distance	*WEP*	*FTP*	*Video*(Mbps)	LR_{s2c}	LR_{c2s}	*RTT*
All within 10 feet	open	no	no	0.00%	0.00%	0.722
			22	2.15%	0.01%	7.498
		yes	no	3.27%	0.00%	20.136
			22	7.23%	0.04%	18.214
	128 bit WEP	no	no	0.00%	0.00%	0.887
			22	2.11%	0.00%	8.083
		yes	no	3.25%	0.00%	19.890
			22	6.65%	0.04%	18.210
Spread	open	no	no	0.00%	0.00%	0.886
			22	1.41%	0.00%	10.139
		yes	no	3.19%	0.01%	20.393
			22	6.94%	0.08%	21.131
	128 bit WEP	no	no	0.00%	0.00%	0.885
			22	2.12%	0.00%	9.892
		yes	no	3.22%	0.00%	20.315
			22	6.87%	0.02%	20.166

Table A.4: Amount of Variation Explained by Each Factor in Design 2

Factor	LR_{s2c}	RTT
$Distance$	0.05%	0.62%
WEP	0.00%	0.00%
FTP	**65.55%**	**86.21%**
$Video$	**30.95%**	**5.16%**
$Distance + WEP$	0.10%	0.02%
$Distance + FTP$	0.02%	0.01%
$Distance + Video$	0.03%	0.43%
$WEP + FTP$	0.11%	0.02%
$WEP + Video$	0.00%	0.00%
$FTP + Video$	**2.96%**	**7.50%**
$Distance + WEP + FTP$	0.00%	0.00%
$Distance + WEP + Video$	0.09%	0.02%
$Distance + FTP + Video$	0.04%	0.00%
$WEP + FTP + Video$	0.11%	0.00%
$Distance + WEP + FTP + Video$	0.01%	0.00%

A.3 Results of Design 3

Tables A.5 and A.6 show respectively for Design 3 the experimental results and the amount of variation in results that is explained by each individual factor or factor interaction.

Table A.5: Results of Design 3

WEP	*NC*	*FTP*	*Video*(**Mbps**)	LR_{s2c}	LR_{c2s}	*RTT*
Open	1	no	no	0.00%	0.00%	0.717
			22	0.36%	0.00%	4.000
		yes	no	0.06%	0.00%	22.675
			22	2.63%	0.00%	13.821
	8	no	no	0.00%	0.00%	0.722
			22	2.15%	0.01%	7.498
		yes	no	3.27%	0.00%	20.136
			22	7.23%	0.04%	18.214
128 bit WEP	1	no	no	0.00%	0.00%	0.879
			22	0.50%	0.00%	5.296
		yes	no	0.02%	0.00%	22.291
			22	3.00%	0.00%	14.451
	8	no	no	0.00%	0.00%	0.887
			22	2.11%	0.00%	8.083
		yes	no	3.25%	0.00%	19.890
			22	6.65%	0.04%	18.210

Table A.6: Amount of Variation Explained by Each Factor in Design 3

Factor	LR_{s2c}	*RTT*
WEP	0.00%	0.03%
NC	**24.89%**	0.53%
FTP	**33.51%**	**86.51%**
$Video$	**24.73%**	0.01%
$WEP + NC$	0.09%	0.01%
$WEP + FTP$	0.01%	0.03%
$WEP + Video$	0.00%	0.05%
$NC + FTP$	**9.70%**	0.06%
$NC + Video$	**2.06%**	**2.19%**
$FTP + Video$	**4.62%**	**10.30%**
$WEP + NC + FTP$	0.04%	0.00%
$WEP + NC + Video$	0.10%	0.01%
$WEP + FTP + Video$	0.00%	0.00%
$NC + FTP + Video$	0.19%	0.27%
$WEP + NC + FTP + Video$	0.05%	0.00%

A.4 Results of Design 4

Tables A.7 and A.8 show the experimental results and the amount of variation in results that is explained by each individual factor or factor interaction in Design 4. Although Table A.8 shows that *Distance* has very small impact on game performance, from Table A.7, we can see that when distance is increased, *RTT* becomes higher. This is consistent with the nature of wireless protocols.

Table A.7: Results of Design 4

Distance	*WEP*	*FTP*	*Video*(Mbps)	*RTT*
Within 10 feet	Open	no	no	0.876
			22	6.873
		yes	no	20.409
			22	19.411
	128 bit WEP	no	no	0.876
			22	8.976
		yes	no	19.935
			22	18.428
75 feet	Open	no	no	0.930
			22	7.841
		yes	no	22.452
			22	25.234
	128 bit WEP	no	no	0.930
			22	10.602
		yes	no	22.216
			22	23.360

Table A.8: Amount of Variation Explained by Each Factor in Design 4

Factor	*RTT*
Distance	**1.55%**
WEP	0.01%
FTP	**87.34%**
Video	**5.05%**
Distance + *WEP*	0.00%
Distance + *FTP*	0.75%
Distance + *Video*	0.39%
WEP + *FTP*	0.35%
WEP + *Video*	0.04%
FTP + *Video*	**4.19%**
Distance + *WEP* + *FTP*	0.01%
Distance + *WEP* + *Video*	0.00%
Distance + *FTP* + *Video*	0.08%
WEP + *FTP* + *Video*	0.24%
Distance + *WEP* + *FTP* + *Video*	0.02%

Appendix B

Result Figures With Error Bars

This appendix displays the error bar figures for the figures shown in Chapter 4. The figures shown in this appendix has the same sequence as those in Chapter 4. To make the figures clearly show the error bars of the performance metrics for different settings, Figure B.4, B.10, B.10, and B.1 only display the results when GC equals to 1, 4 and 8; Figure B.6 and B.12 plots only the results when FTP=Yes and $Video$ is 0, 16, 20, and 24Mbps; Figure B.5 and B.11 shows the results when FTP=No and $Video$ is 16, 20, and 24Mbps.

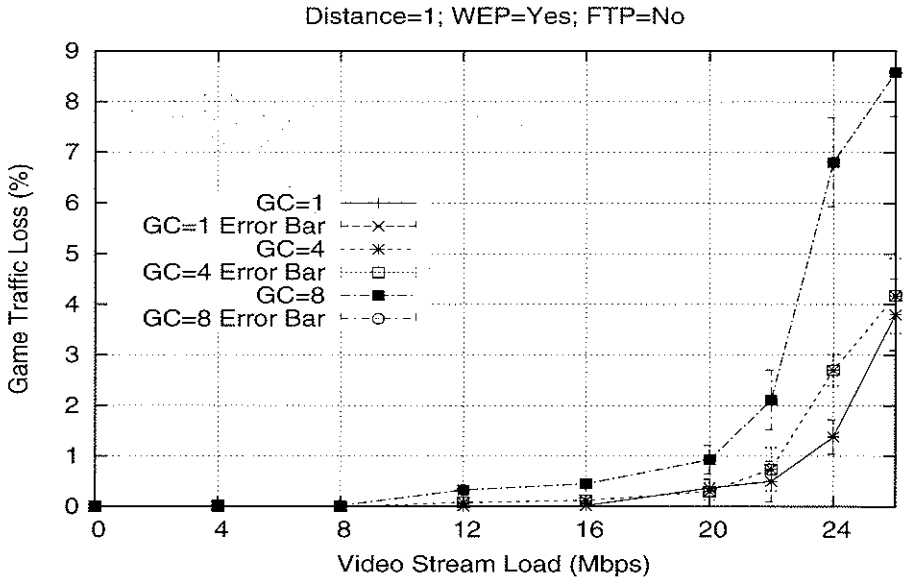

Figure B.1: Loss Ratio vs. Amount of Video Stream with Error Bars

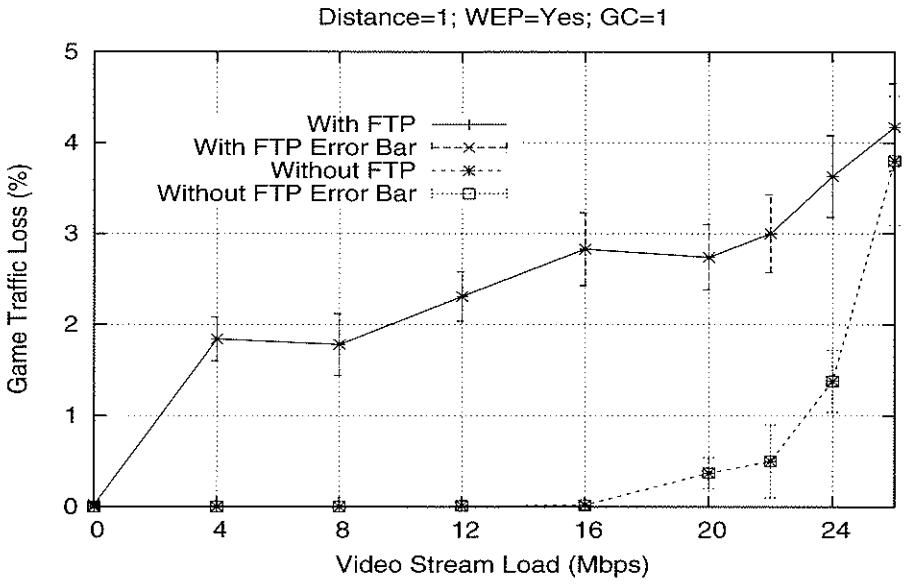

Figure B.2: FTP Impact on Loss Ratio for 1 GC with Error Bars

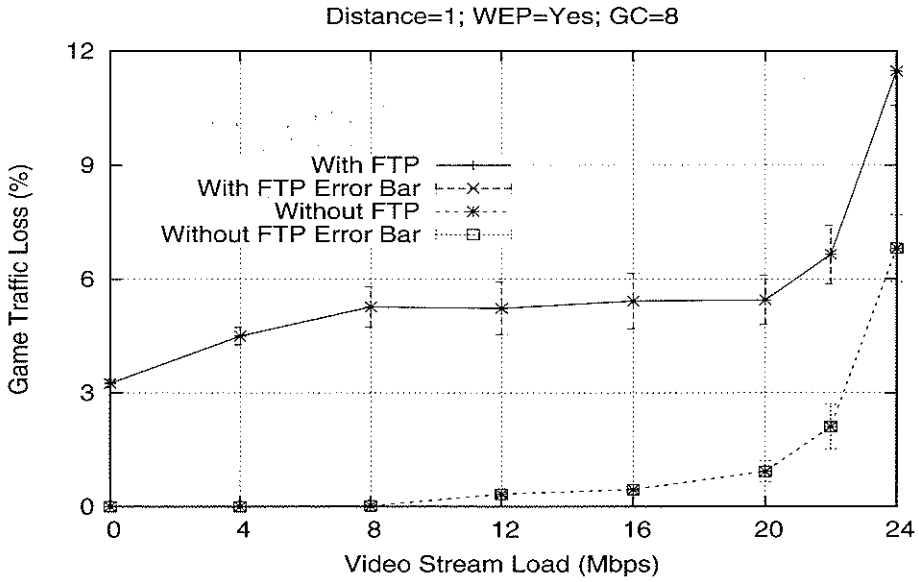

Figure B.3: FTP Impact on Loss Ratio for 8 GC with Error Bars

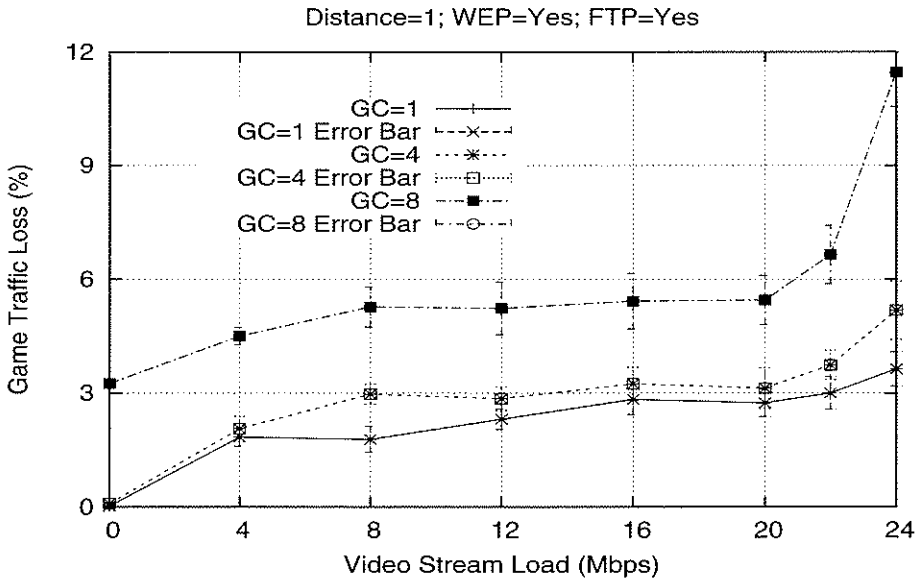

Figure B.4: Loss Ratio vs. FTP+UDP with Error Bars

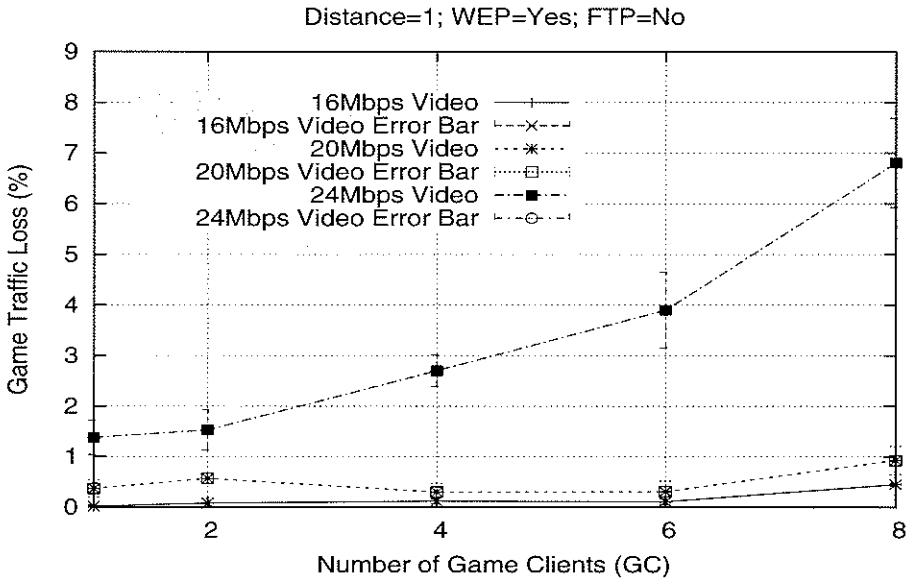

Figure B.5: Loss Ratio Time vs. Number of GC (UDP only) with Error Bars

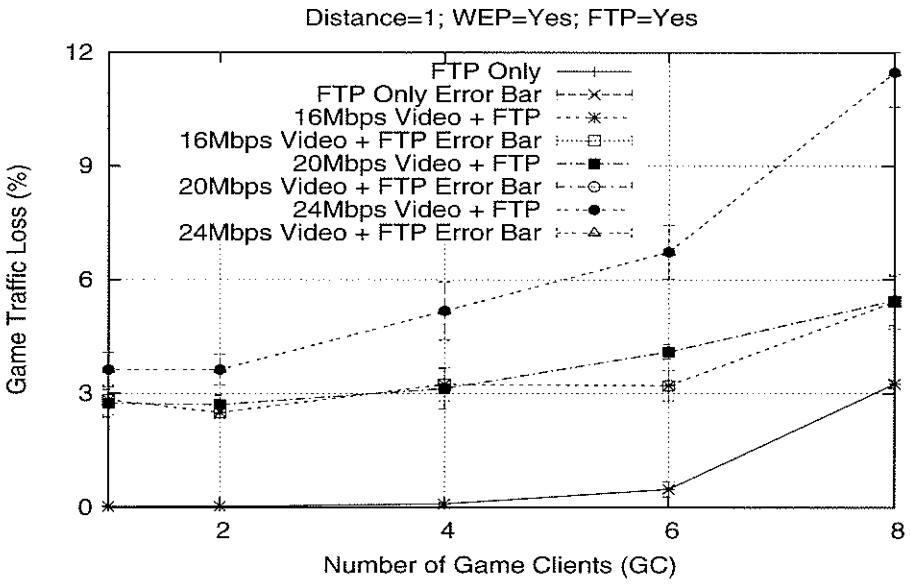

Figure B.6: Loss Ratio Time vs. Number of GC (with FTP) with Error Bars

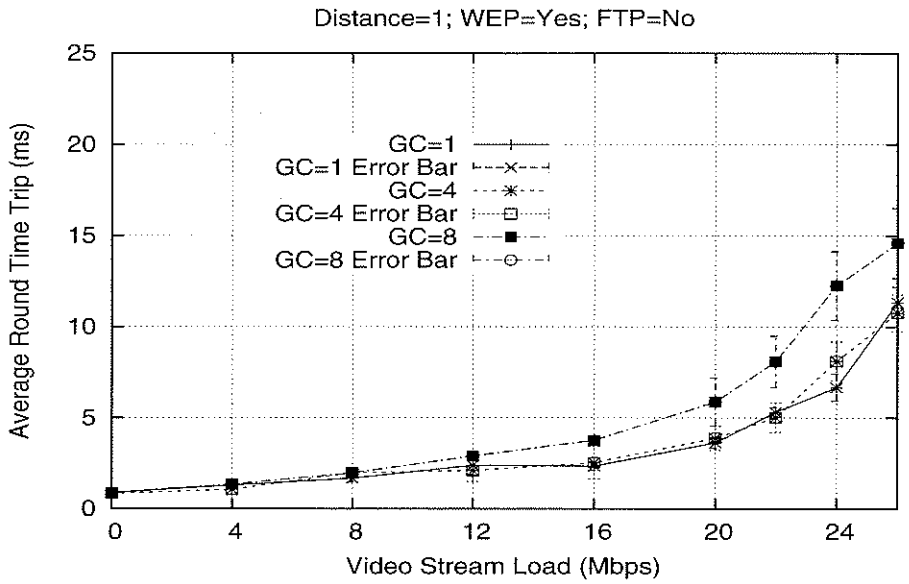

Figure B.7: Round-Trip Time vs. Amount of UDP with Error Bars

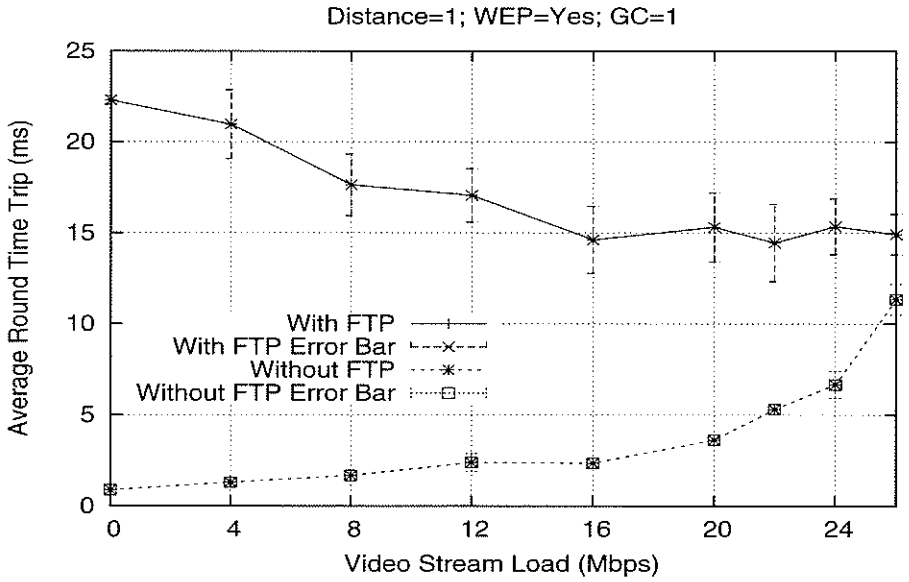

Figure B.8: FTP Impact on Round-Trip Time for 1 GC with Error Bars

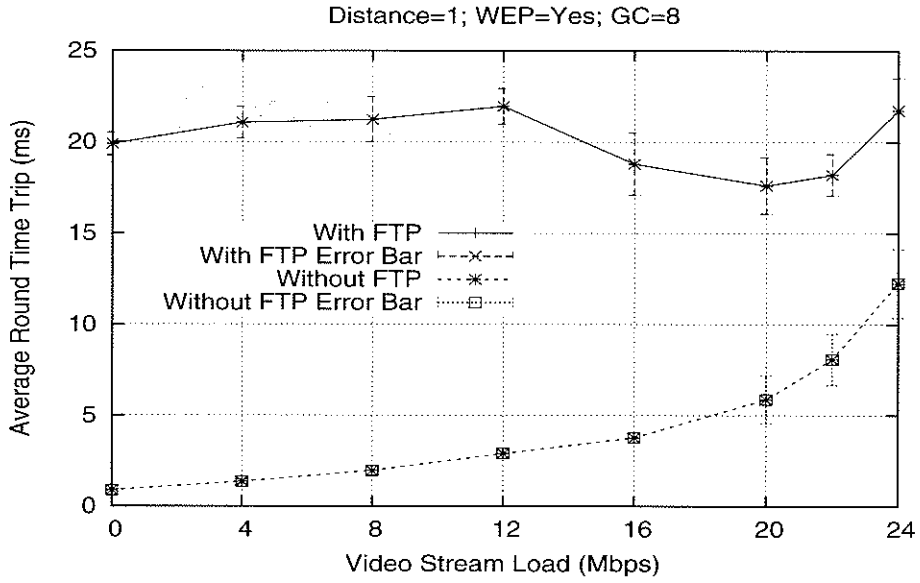

Figure B.9: FTP Impact on Round-Trip Time for 8 GC with Error Bars

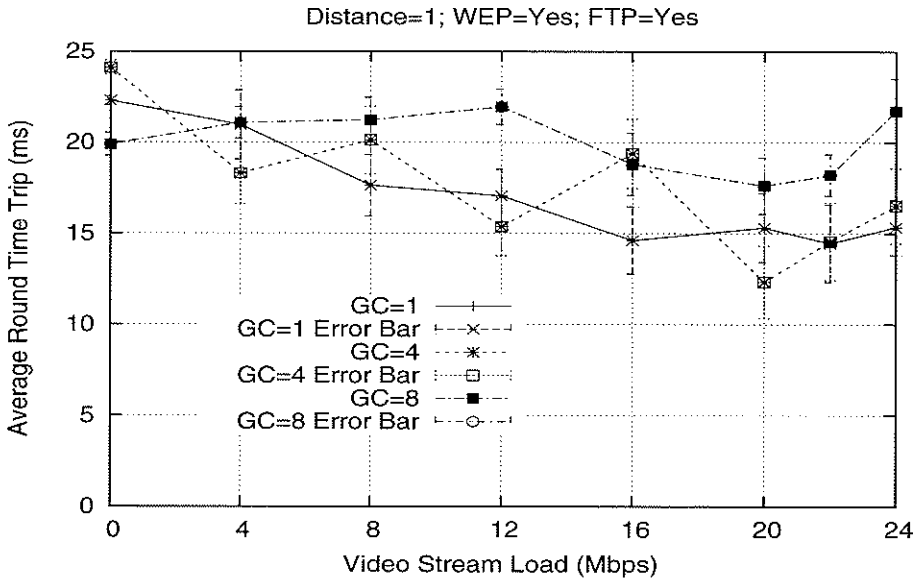

Figure B.10: Round-Trip Time vs. FTP+UDP with Error Bars

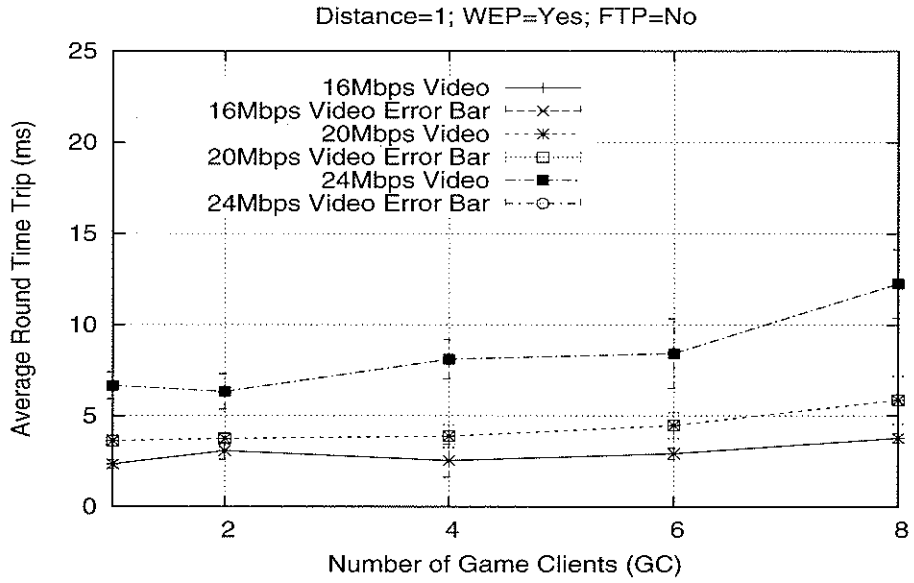

Figure B.11: Round-Trip Time vs. Number of GC (UDP only) with Error Bars

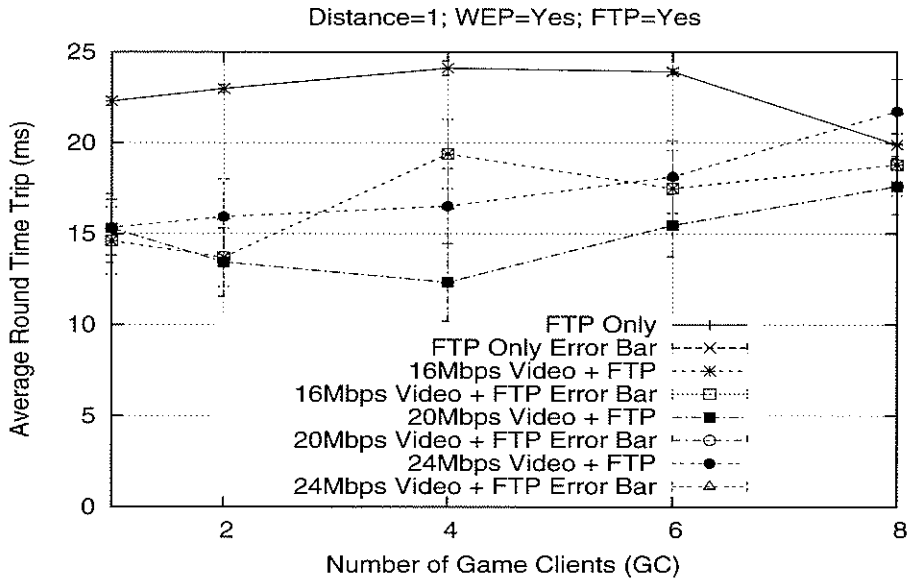

Figure B.12: Round-Trip Time vs. Number of GC (with FTP) with Error Bars

Appendix C

Background Traffic Throughput When With FTP

This appendix reports the background traffic throughput when a FTP session is included in the background. The throughput of FTP and video streaming are reported separately in two sections. In the third section, the overall throughput is provided.

C.1 FTP Download Throughput

From Figure C.1, it can be observed that FTP throughput decreases noticeably when the UDP (Video) traffic is increased.

Table C.1: FTP Throughput (Mbps)

Video (Mbps)	1 GC	2 GCs	4 GCs	6 GCs	8 GCs
0	27.20	27.20	25.60	24.80	24.00
4	18.40	17.60	17.60	16.00	13.71
8	13.26	14.78	13.78	12.93	13.75
12	13.92	12.75	11.83	9.52	10.93
16	9.66	7.16	8.67	5.76	6.81
20	3.15	10.45	10.63	8.59	7.64
22	7.20	8.58	7.96	6.40	7.54
24	7.82	7.48	7.05	6.78	5.74

C.2 UDP Video Stream Loss Ratio

Table C.2 shows that the UDP (Video) traffic loss rate increases when the load increases.

Table C.2: Video Stream Loss Rate (%)

Video (Mbps)	1 GC	2 GCs	4 GCs	6 GCs	8 GCs
4	4.51	2.20	6.90	3.10	5.69
8	8.39	5.40	3.50	4.40	6.37
12	7.37	6.68	7.80	7.40	6.50
16	7.42	7.88	6.03	6.39	6.64
20	7.79	9.26	10.30	8.07	9.74
22	9.34	7.34	9.20	10.22	10.93
24	9.44	8.35	9.90	10.27	14.82

C.3 Overall Background Traffic Size

Figure C.3 provides the overall background traffic throughput when both video stream and FTP background traffic exist. The calculation formula used is:

$$Throughput_{Overall} = (Load_{Video} \times \frac{100 - Loss_{Video}}{100}) + Throughput_{TCP}$$

Table C.3: Overall BT Throughput (Mbps)

Video (Mbps)	**1 GC**	**2 GCs**	**4 GCs**	**6 GCs**	**8 GCs**
0	27.20	27.20	25.60	24.80	24.00
4	22.22	21.51	21.32	19.88	17.49
8	20.58	22.35	21.50	20.58	21.24
12	25.03	23.95	22.90	20.63	22.15
16	24.48	21.90	23.71	20.74	21.74
20	21.59	28.60	28.57	26.97	25.69
22	27.15	28.96	27.94	26.15	27.14
24	29.55	29.48	28.67	28.32	26.18

Bibliography

[1] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control (rfc 2581). http://www.ietf.org/rfc/rfc2581.txt?number=2581, April 1999.

[2] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The effects of loss and latency on user performance in unreal tournament 2003. In *Proceeding of 2004 ACM SIGCOMM Workshop on NetGames (NetGames2004)*, August 2004.

[3] IEEE-SA Standards Board. *ANSI/IEEE Standard 802.11*, chapter 11. IEEE, 1999.

[4] M. S. Borella. Source models of network game traffic. *Computer Communications*, 23(4):403–410, February 2000.

[5] X. Cao, G. Bai, and C. Williamson. Media streaming performance in a portable wireless classroom network. In *Proceeding of IASTED European Workshop on Internet Multimedia Systems and Applications (EuroIMSA)*, February 2005.

[6] M. Claypool, D. LaPoint, and J. Winslow. Network analysis of counter-strike

and starcraft. In *Proceeding of the 22nd IEEE International Performance, Computing, and Communications Conference (IPCCC)*, April 2003.

[7] Atheros Communications. Methodology for testing wireless LAN performance with chariot. http://www.atheros.com/pt/whitepapers/Methodology_Testing_WLAN_Chariot.pdf.

[8] Microsoft Corporation. The age of kings. http://www.microsoft.com/games/age2/.

[9] Microsoft Corporation. Gamespy status. http://archive.gamespy.com/stats/.

[10] NCsoft Corporation. Lineage2. http://www.lineage2.com.

[11] Valve Corporation. Counter strike. http://www.counter-strike.net/.

[12] Blizzard Entertainment. Warcarft. http://www.battle.net//.

[13] J. Färber. Network game traffic modelling. In *Proceeding of the 1st ACM workshop on Network and System Support for games (NetGames2002)*, April 2002.

[14] W. Feng, F. Chang, W. Feng, and J. Walpole. Provisioning on-line games: A traffic analysis of a busy counter-strike server. In *Proceeding of SIGCOMM Internet Measurement Workshop*, November 2002.

[15] Frank H.P. Fitzek and Martin Reisslein. Mpeg-4 and h.263 video traces for network performance evaluation. http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html.

[16] J. Gretarsson, F. Li, M. Li, A. Samant, H. Wu, M. Claypool, and R. Kinicki. Performance analysis of the intertwined effects between network layers for 802.11g transmissions. In *Proceeding of ACM Wireless Multimedia Networking and Performance Modeling (WMuNep) 2005*, October 2005.

[17] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings Interantional Conference on Mobile Computing and Networking (MobiCom)*, September 2004.

[18] Tim Higgins. Enhanced 802.11g needtoknow. http://www.tomsnetworking.com/Sections-article67-page6.php.

[19] id Software Inc. http://www.idsoftware.com/.

[20] id Software Inc. Quake 3. http://www.idsoftware.com/games/quake/quake3-gold.

[21] Sony Online Entertainment Inc. Everquest2. http://everquest2.station.sony.com/#home.

[22] J. F. Kurose and K. W. Ross. *Computer networking: a top-down approach featuring the Internet.* Addison Wedley, third edition, 2005.

[23] J. Lakkakorpi, A. Heiner, and J. Ruutu. Measurement and characterization of internet gaming traffic. Research Seminar on Networking, Helsinki University of Technology, Networking Laboratory, Espoo, Finland, February 2002.

[24] T. Lang, G. Armitage, P. Branch, and H. Choo. A synthetic traffic model for

half-life. In *Proceedings of the Australian Network and Telecommunications Conference (ATNAC) 2003*, December 2003.

[25] T. Lang, P. Branch, and G. Armitage. A synthetic traffic model for Quake3. In *Proceeding of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2004 (ACE'04)*, June 2004.

[26] A. Law and W. Kelton. *Simulation Modeling and Analysis*, chapter 9. McGraw-Hill Inc., New York, second edition, 1992.

[27] Epic MegaGames Inc. http://www.epicgames.com.

[28] T. Nguyen and G. Armitage. Quantitative assessment of IP service quality in 802.11b and DOCSIS networks. In *Proceeding of the Australian Network and Telecommunications Conference (ATNAC) 2004*, December 2004.

[29] C. E. Palazzi, G. Pau, M. Roccetti, and M. Gerla. In-home online entertainment: Analyzing the impact of the wireless MAC-transport protocols interference. In *Proceedings of IEEE International Conference on Wireless Networks, Communications, and Mobile Computing (WIRELESSCOM 2005)*, June 2005.

[30] L. Pantel and L. Wolf. On the impact of delay on real-time multi-player games. In *Proceeding of NOSSDAV'02*, May 2002.

[31] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[32] L. L. Peterson and Bruce S. Davie. *Computer networks: a Systems Approach.* Morgan Kaufmann Publishers, third edition, 2003.

[33] J. Postel. Transmission control protocol, darpa internet program protocol specification (rfc 793). http://www.ietf.org/rfc/rfc0793.txt?number=793, September 1981.

[34] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, and N. Degrande. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In *Proceeding of 2004 ACM SIGCOMM Workshop on NetGames (NetGames2004)*, August 2004.

[35] Y. Seok. Online game traffic measurement & analysis. Multimedia and Mobile Communications Laboratory Seminar, Seoul National University, Korea, January 2003.

[36] SGI. Opengl. http://www.opengl.org/.

[37] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in warcraft 3. In *Proceeding of 2003 ACM SIGCOMM Workshop on NetGames (NetGames2003)*, May 2003.

[38] J. Smed, T. Kaukoranta, and H. Hakonen. A review on network and multi-player computer games. Technical Report TUCS Technical Report 454, Turku Centre for Computer Science, April 2002.

[39] W. Stevens. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms (rfc 2001). http://www.ietf.org/rfc/rfc2001.txt?number=2001, January 1997.

[40] J. Tomcik. Gaming models for evaluation criteria. Technical Report IEEE C802.20-04/86, IEEE, November 2004. http://grouper.ieee.org/groups/802/20/Contribs/C802.20-04-86.ppt.

[41] J. Weinmiller, H. Woesner, J. Ebert, and A. Wolisz. Analyzing the RTS/CTS mechanism in the DFWMAC media access protocol for wireless LAN's. In *Proceedings of Communications Systems of International Federation for Information Processing (IFIP TC6) Workshop Personal Wireless Communications (Wireless Local Access)*, April 1995.

[42] E. W. Weisstein. Extreme value distribution. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ExtremeValueDistribution.html.

[43] A. Wijesinha, Y. Song, M. Krishnan, V. Mathur, AhnJ., and V. Shyamasundar. Throughput measurement for UDP traffic in an IEEE 802.11g WLAN. In *Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and first ACIS International Workshop on Self-Assembling Wireless Networks*, May 2005.

[44] W. Willinger, M. Taqqu, R. Sherman, , and D. Wilson. Selfsimilarity through high-variability: Statistical analysis of ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, February 1997.

[45] K. Xu, M. Gerla, and S. Bae. How effective is the ieee 802.11 rts/cts handshake in ad hoc network. In *Proceedings of IEEE GlobeCom'02*, November 2002.