Performance Analysis of Wireless Multiplayer Games on Terraplay Systems

XU CHEN

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2005

IMIT/LCN 2005-22

Performance Analysis of Wireless Multiplayer Games on Terraplay System

Xu Chen

Stockholm, Sweden

10 October 2005

Examiner: Professor Gerald Q. Maguire Jr

Industrial Supervisors: Magnus Jändel, CTO, Terraplay System AB

Abstract

This thesis project was sponsored by Terraplay System AB. Terraplay System is a wireless online multiplayer game service provider in Europe. The purpose of this project was to provide a convenient and free latency test tool named Terraplay Test Toolkit (TTK) for game developers who utilize the Terraplay platform. Since wireless multiplayer games are latency sensitive, it's necessary for game developers to estimate the real-time game latency before their development. TTK is based on Java Mobile Edition (J2me) technology, it can performs a real-time Round Trip Time (RTT) latency test. This project explored the application of the mobile development. Mobile development is a brand new field and becoming more and more popular. The design process and source code can be used as reference for mobile developers. Based on the Terraplay Test Toolkit, a series of experiments were conducted to test the ability of TTK. These experiments focused on end-to-end latency effects of wireless multiplayer games under different situations such as different packet size, different sending rates, high speed movement, and so on. Although TTK is not a professional test tool, it still reflects the latency variance under different conditions correctly. From these experiments, it was found that for the Terraplay enabled wireless multiplayer games, large average packet size (300 bytes) is NOT a factor with regard to the latency on the Terraplay System; the packet rate can affect the game latency. In a high speed-moving environment, game latency doesn't obvious increase. Using HTTP can cause three times higher latency than simply using TCP. These conclusions are based on experimental results. These conclusions should guide wireless multiplayer game developers and game players.

Abstract in Swedish

Denna avhandling är sponsrad av Terraplay System AB. Terraplay System är en trådlös multiplayer-spelservice i Sverige. Syftet med detta projekt var att ge ett lättillgängligt och gratis testverktyg för fördröjning kallat Terraplay Test Toolkit (TTK) till spelutvecklare som använder Terraplays plattform. Eftersom trådlösa multispel är känsliga för fördröjning, är det nödvändigt för spelutvecklare att innan de börjar göra en uppskattning av denna. TTK är baserat på Java Mobile Edition-teknologi (J2me), och kan utföra ett fördröjningstest i realtid, kallat Round Trip Time (RTT). Detta projekt undersökte tillämpningen av den mobila utvecklingen. Mobil utveckling är ett helt nytt fält och blir mer och mer populärt. Designprocessen och källkoden kan användas som referenser för mobila utvecklare. Baserat på Terraplay Test Toolkit gjordes en serie experiment för att testa TTK:s förmåga. Dessa experiment fokuserade på fördröjda effekter av trådlösa multiplayerspel under olika situationer så som olika paketstorlek, olika leveranshastigheter, höghastighetsrörelser och så vidare. Även fast TTK inte är ett professionellt testverktyg, och inte kan användas för exakta fördröjningsanalyser, återger det ändå fördröjningens variationer under olika villkor. Dessa experiment visade att trådlösa multiplayerspel anpassade till Terraplay, med genomsnittlig paketstorlek (300 bite) INTE är en faktor med avseende på fördröjningen till Terraplay System; paketets hastighet kan påverka paketets fördröjning. I en miljö med hög rörelsehastighet, ökar inte påtagligt spelets fördröjning. Användandet av HTTP kan orsaka en tre gånger så lång fördröjning, jämfört med att bara använda TCP. Slutsatserna är baserade på experimentresultaten. Dessa slutsatser kan hjälpa utvecklare av trådlösa multiplayerspel

Acknowledgements

This report is a result of my thesis project at Terraplay System AB. I would like to express my sincere gratitude to people at Terraplay and KTH:

I want to thank the CTO of Terrapaly, Magnus Jändel for not only giving me the opportunity to do this work for Terraplay, but also for his guidance during the whole project.

The thesis project would not been successfully without the help of my examiner, Professor Gerald Q. Maguire Jr. I would like to thank him for all his support, his quick feedback, and his inestimable comments.

During the project, my parents encourage me a lot, especially when I encountered problems. Their encouragement gave me the confidence to overcome these challenges. I would like to thank them for their deep love and support.

Finally, I would like to thank Dr. Johan Montelius for his kind suggestions regarding my mobile development; friends, Wu Di and Gan Wen Jia helped me with various test scenarios, as well as technical discussions.

Content

List of Figures

Chapter 1

Introduction

1.1 Background

Along with the 3G boom, all kinds of value added applications are appearing. For most telecom service providers, wireless multiplayer games are among the most important services. Multi-player mobile games means more than two players play together in the same game session using their own mobile phones or PDAs. Providing a multiplayer gaming service opens up potential for generating a higher Average Contribution Per User (ACPU), some of the revenue comes through an increase in the use of services other than the game itself. As a commercial service, wireless multiplayer games can increase ACPU from the following sources [1]:

- Data traffic and air-time used in the games for person-to-person communication
- Charging for multiplayer games, as consumers are accustomed to paying recurring fees for on-line activities such as subscriptions and download fees
- Increase in MMS and SMS traffic when people communicate with opponents before or after the game
- Voice traffic when gamers call each other before or after the game.
- Increase in number in GPRS and 3G subscriptions and activations (since they are needed in order to play multiplayer games). Multiplayer games are exactly the type of services that create effective viral and word-of-mouth marketing schemes.

3G communication technologies provide wide bandwidth and high-speed data transmission over wireless network. 3G terminals including mobile phones and PDAs are also become more and more powerful. These terminals not only have a color LCD screen, but also a powerful CPU and large memories. Compared with a PC, 3G phones are more popular. In some countries, the number of Mobile phones already exceeds the population. According to the prediction of In-Stat/MDR, in China, the 3G subscribers will be 118 million in 2008.[2] Frost & Sullivan research center claims that the mobile game industry in Europe has created about US$800 million revenue in the year 2002. They project it will be US$7 billion in 2006 [3]. All of these sources predict that wireless multiplayer games will be a huge market in the future. Today, a lot of companies are investing in this field and wireless multiplayer games are already one of the hottest discussion topics in many forums.

This bright future has brought prosperity to the multiplayer game community. The game community is the ecology system in the mobile game world. It includes game players

(players are the customers of mobile games), game developers, and telecom service providers (for example 3, Vodafone). Terraplay offers network solutions aimed at providing high quality, and commercial real-time gaming service, which targets both existing and future mobile and fixed networks. These systems intend to enable new types of multi-player games and gaming experiences by focusing on network technology. These systems intend to enable the operator to launch and run high performance commercial on-line gaming services both resource and cost effectively. They try to enable a wide range of business models based on the systems' features.[4] By integrating Terraplay's technologies, a telecom service provider can support wireless multiplayer game services in its networks.

1.2 Problem Statement

Since the service quality of wireless networks varies a lot both with both location and environment. It's important for mobile game developers to estimate the performance at the concept stage of their wireless multiplayer games, which will run on the Terrapaly platform. Factors that may affect the performance of mobile games include network latency, bandwidth, signal strength, and so on. Among them, game latency directly affects the performance of the wireless multiplayer games. The developers need to estimate the latency for their game before developing their games or they may need to change the game functions. Evaluating the effect of different factors on the performance of wireless multiplayer games on Terraplay systems that are utilizing 3G networks is the problem that I have focused on. The research results provide a useful reference to mobile game developers as they design a game from concept to implementation.

1.3 Scope of the Thesis

In order to analyze the performance of wireless multiplayer games under different conditions, I needed a test tool to emulate these multiplayer games on mobile phones and to exam the latency. In order to emulate these multiplayer games, I needed to test all kinds of multiplayer games to discover the major game metrics. These game metrics describe the common traffic characteristics that reflect the differences between different game genera. Additionally, because the test tool needs to communicate with the Terraplay System just like any of the multiplayer games, the functions and mechanisms of Terraplay System where also studied.

Due to the complex causes of latency in mobile networks, this thesis project did not focus on analyzing the reasons for this latency. Rather, it exams how latency variations can affect the performance of multiplayer games. Thus it was necessary to do experiments to observe this performance variation.

In summary, the scope and objectives of my thesis project were:

- To test multiplayer games that belong to different game genera,
- Identify relevant metrics of wireless multiplayer games,
- Developing a test toolkit for multiplayer games on Terraplay System,
- Measure game performance according to the selected metrics and conditions, and
- Reach some conclusions concerning the effects on performance

Chapter 2

Wireless Multiplayer Games

2.1 Concept of Wireless Multiplayer Game

A wireless multiplayer game involves over two players in the same game session each using their own wireless device such as PDA or Mobile phone. Players communicate and control their game through wireless networks. The game implementation can be embedded into mobile phones by manufactures or downloaded by players.

As a new wireless application, multiplayer games have important differences from traditional mobile games and PC based multiplayer games. Traditional mobile games only permit one player in each game session. They have no multiplayer functions. Additionally these mobile games don't need to connect to mobile networks, as there is no communication with other players. Compared with PC multiplayer games, wireless multiplayer games have the advantage of portability, but are more strongly effected by the network conditions.

2.2 Implementation Technology

Different technologies can be used for wireless multiplayer games. A number of different technologies, listed below, are used for games on mobile phones.

2.2.1 Embedded Games

Some games are programmed to run natively on the phone itself, installed on the phone at the factory, and shipped with it. Snake, available on many Nokia phones for more than four years, is the most famous example. However, consumers cannot install new embedded games themselves. These kinds of games are becoming less prevalent now.

2.2.2 SMS Games

Short Message Service (SMS) is used to deliver short text messages from one phone to another user. SMS games are played by sending a message to a phone number that corresponds to the game provider’s server, which receives the messages, performs some processing, and returns a message to the player with the results. SMS is not a particularly good technology for games, because it depends on text entry by the user, and thus is, in essence, a command-line environment. It is also expensive for a game of any significant duration. Although the deployment of Multimedia Message Service (MMS) technology makes message-based games more appealing, because the game content can include

more media materials such as better music, pictures and even video, this still not a great game environment.

2.2.3 J2ME Application

Java 2 Micro Edition (J2ME) is a form of the Java language that is optimized for small devices such as mobile phones and PDAs. Tens of millions of Java-enabled phones are already in consumers' hands. J2ME is limited in comparison to desktop Java, but it vastly improves the ability of mobile phones to support games compared with SMS. It allows far better control over the user interface than either SMS or WAP, and can connect over the network to a remote server. Because of its capabilities and the widespread and increasing deployment of Java-enabled phones, it is a natural platform for mobile game development today. J2ME is not the only interpreted language deployed on phones, but it is an industry standard backed by many manufacturers and therefore offers a large and growing installed base. J2ME fulfills the JTWI specification.

The Java Technology for the Wireless Industry (JTWI) specification, JSR 185, defines the industry-standard platform for the next generation of Java technology-enabled mobile phones. JTWI is defined through the Java Community Process (JCP) by an expert group of leading mobile device manufacturers, wireless carriers, and software vendors. JTWI specifies the technologies that must be included in all JTWI-compliant devices: Connection Limited Device Configuration 1.0 (CLDC 1.0), Mobile Information Device Profile 2.0 (MIDP 2.0), and Wireless Messaging API1.1 (WMA 1.1), as well as CLDC 1.1 and Mobile Media API (MMAPI) where applicable. CLDC is a "lowest common denominator" standard that includes only the minimal Java platform features and APIs for a wide range of consumer devices. The value of CLDC is to cover most types of mobile phones and PDAs. CLDC is a foundation of JTWI. MIDP is the profile based on CLDC. MIDP provides extended class libraries to support much richer functions such as User Interface (UI), and the record management system (RMS). WMA1.1 includes a message API. This messaging API is based on the Generic Connection Framework (GCF), which is defined in the CLDC 1.0 specification. The Mobile Media API (MMAPI) provides a powerful, flexible, and simple interface to multimedia capabilities. It exposes an interface for playing and recording audio and video data. The specification of JTWI raises the bar of functionality for high-volume devices, while minimizing API fragmentation and broadening the substantial base of applications that have already been developed for mobile phones.[5] The structure of JTWI is shown in the Figure 2-1. JTWI assures the application programs have the biggest cross-platform ability. The definition of "standard-size application" clearly defines the size of the application, the minimum available memory for each game application. By strictly abide by the requirements of JTWI, the game applications should work on most mobile phones without any changes.

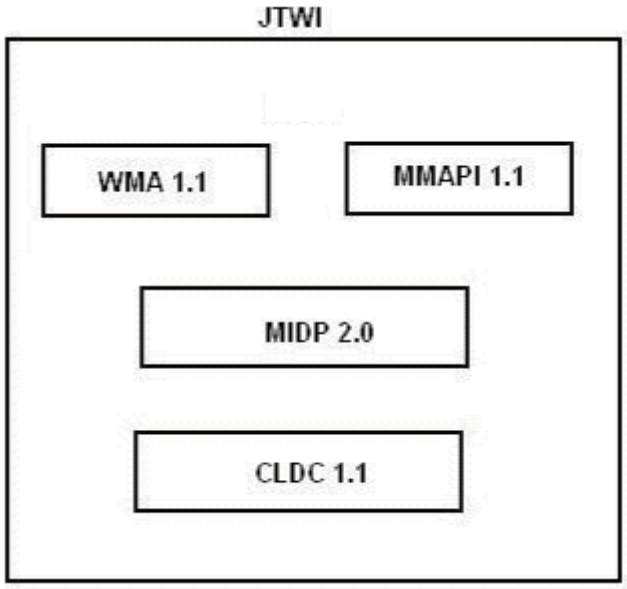


Figure 2-1: JTWI Structure

2.2.4 C++ Applications

Mobile games can also be developed in C++, a language that compiles to native machine code. Compiled languages in general offer better control over the UI because it can directly access the phone's hardware, and offer greater speed for the same processing power when compared to an interpreted language. Development in C++ enables rich, high-performance games. Currently, there are some C++ mobile game applications. One of the most successful C++ mobile game platforms is Mophun. Mophun™, a product of Synergenix Interactive, is a gaming accelerator designed to bridge the gap between traditional console based and mobile games. It was designed to harness the hardware resources of a mobile device and create a wonderful gaming experience regardless of the specific hardware.[1] Compared with the J2ME platform, Mophun can provide more impressive game effects, especially 3D effects. In my game tests, I selected Mophun games for analysis. Although Mophun was developed in C++, Terraplay use the same mechanism to enable multiplayer functions for J2ME multiplayer wireless games.

2.3 Limitation for Mobile Games

In terms of processing power and capabilities, the current generation of phones is similar to the second generation of arcade machines, early 1990s home computers, and early handheld game machines. They also have, by comparison with PCs, limited input and display capabilities: e.g. small screens and keypads optimized for phone dialing.

2.3.1 Limited Game Size

Most of the current mobile games are smaller than 250KB. The size limitation of mobile games comes from many aspects: limited memory, small screen size, and limited CPU processing ability. Most Java-enabled phones have a limited amount of memory that is available for MIDlets (Sun's name for J2ME applications). There's always a limitation on

[1] http://www.mophun.com

the size permitted for a MIDlet, since mobile devices only have limited memory. The actual limit varies in different handsets and (sometimes) the carrier's policies. The small screen of mobile limits the visual effects in mobile games and introduces design limitations in the game scenario. And the CPU of the mobile phone is also not good enough for processing very high quality 3D games. All of these factors decide the size of current multiplayer games will be limited.

2.3.2 Limiting Battery Consumption

Battery capacity in mobile stations has always been a scarce resource and therefore has had a fundamental effect on the design of both mobile stations and cellular networks. On the mobile station side, the limited battery capacity has set many constraints on the physical implementation of the mobile station, for example, screen size and available processing power. On the network side, the support for battery life-time saving is implemented by a specific mobile station (MS) state transfer mechanism. Actual implementation of the MS state transfer mechanism is dependent on the network type (WCDMA/GSM). WCDMA networks use Radio Resource Control (RRC) mechanism to decrease battery consumption. In WCDMA networks, the MS is in Idle, CELL_FACH (Forward access channel), CELL_DCH (Dedicated channel), or CELL_PCH (Paging channel) state. In the idle state, the MS doesn't have any RRC connection, whereas in all other states, the RRC connection exists. The power consumption in CELL_PCH is significantly lower than in other RRC states, being only about 1 percent of the power consumed in the CELL_DCH state. Correspondingly, in the CELL_FACH state, power consumed is only about 50 percent of that in the CELL_DCH state. Generally with games the power consumption in the Dedicated Channel (DCH) state is about the same level as a normal speech call if the bit rate used is at the same order of magnitude.[6] However, even though the MS state transfer mechanism allows the mobile station usage time to be extended, the change from one state to another always causes extra delay, which, depending on the game type, may have an effect on the gaming experience.

2.4 Architecture of Mobile Multiplayer Games

The system architecture determines how mobile multiplayer games are implemented. There are mainly two architectures. One architecture is peer-to-peer and the other is server-driven. The main difference between these two architectures is the existence of a proprietary game server. Each architecture had its own advantages and disadvantages.

2.4.1 Peer-to-Peer Architecture

Use of a Peer-to-peer architecture means all of the players are directly connected with each other without the coordination of any game server. All of the clients store and update the game state according to the game logic in each client. Figure 2-2 shows the peer-to-peer structure.

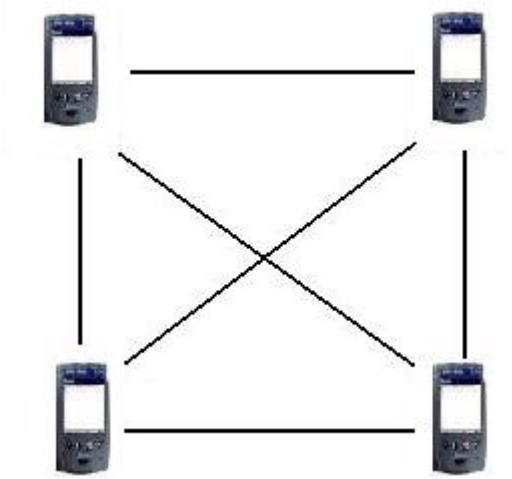


Figure 2-2: Peer-to-Peer Structure

The absence of a separate game server makes this architecture cheaper for the service provider. Since each client in this architecture has the same status, one broken client will not result in the collapse of the whole game session. This makes the architecture more stable. However, It does have drawbacks: For one thing, if the game has any hidden information (areas of the map a player is not entitled to see, for example), it is fairly easy to "hack the client" to reveal this hidden information. Second, in a peer-to-peer game, the amount of data that needs to be exchanged increases exponentially with the number of players. As an example, if the number of players increases from 4 to 5, each player must now be sending information out to 4 others, rather than 3 others. So, we have increased network traffic from 12 (4 players each sending to 3 others) to 20 (5 players each sending to 4 others). On a dial-up connection, a player may quickly start to run up against bandwidth limits. Peer-to-peer games used to be limited to 8 or fewer players, although clever data compression schemes have allowed some to support up to 32 simultaneous players.[9]

2.4.1 Server-driven Architecture

A Server-driven architecture has a game server, to which all clients connect. Most current multiplayer games are based on a client/server architecture. In a server-driven architecture, one entity acts as a game server that stores the game states and distributes the changes in game state to all clients according to the game logic. The game logic usually resides partly in the server and partly in the clients. The goal is to minimize the traffic in the network by performing as much processing as possible in the server and clients separately, and only updating the game state when it is needed. Therefore, a separate game server is easier to control and it also leads to lower latency as compared to peer-to-peer architecture. The Server-driven's architecture is shown in Figure 2-3. The architecture of a multiplayer game has significant effects on the game performance: Terraplay Move is a wireless multiplayer game server. It has already been successfully deployed in Europe. Vodafone and 3 use this product to provide multiplayer game service for their customers.

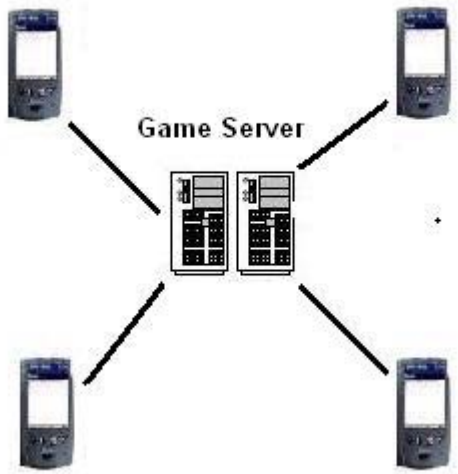


Figure 2-3: Server-driven Architecture

2.5 Communication Technology

Wireless multiplayer games need to communicate with Terraplay Game Server-Terraplay Move. Terraplay Move includes Move Gateways (MGW), Move Lobby. They are responsible for packet distribution and client coordination. Terraplay's Move Software Developer Kit (Move SDK)[7] supports J2ME technology and lets developers to create exciting multiplayer games. For Java enabled mobile phones, MIDP1.0 supports HTTP, while MIDP 2.0 supports HTTP, TCP, and UDP. Mobile networks have different properties from the fixed networks over which most Internet traffic runs: they have relatively low bandwidth and relatively high latency. In order to improve the communication performance, these communication technologies often have been optimized for the wireless environment.

2.5.1 HTTP

HTTP is one of the most common communication technologies for mobile devices. All new 3G mobile phones support this protocol. When a phone performs normal Internet IP access over its data connection, a wireless IP router is used to forward packets to the Internet. This device routes IP packets from the wireless bearer to the wired bearer.
Figure 2-4 shows the structure of an HTTP connection over such an underlying network.

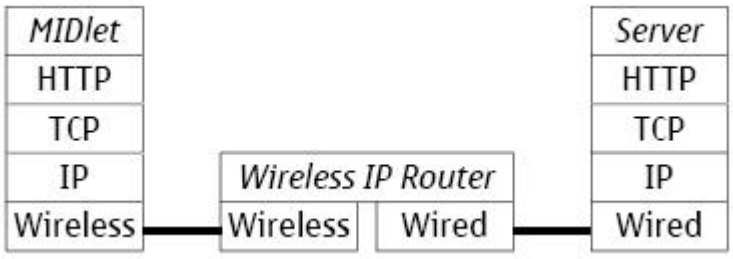


Figure 2-4: Basic HTTP connection

WAP 2.x introduces the concept of a wireless-profiled HTTP and wireless-profiled TCP, which introduced wireless-related optimizations to these protocols, but requires a WAP Gateway (WAP Proxy) to convert between the wireless forms of these protocols and the

normal Internet forms. Current all new 3G mobile phones support WAP. The WAP Gateway can increase the communication performance. Figure 2-5 shows the structure of such an HTTP over WAP Gateway.

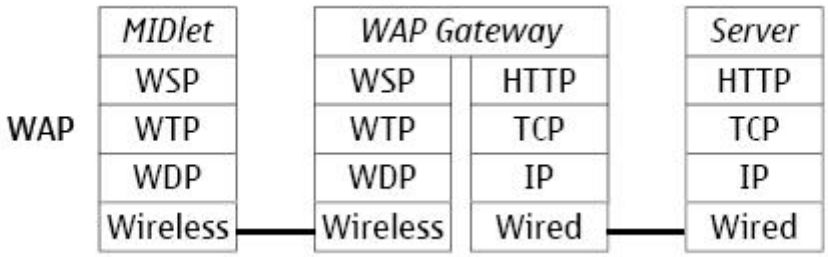


Figure 2-5 HTTP over WAP

2.5.2 TCP and UDP

TCP is a reliable connection-oriented transmission protocol. Data is always sent and received by the application in the correct order. A connection is established between the two communicating machines and maintained for the duration of the communication. A "socket" refers to one end point of a TCP connection. A "server socket" on a TCP server accepts new connection requests and creates a new socket for subsequent use by each requestor. TCP's reliability often greatly simplifies program design; however, when it is used over an unreliable network it can perform significantly worse than when using the User Datagram Protocol. If a packet is lost, TCP tries to resend it and will not deliver subsequent packets until the lost packet has been successfully resent. If you don't need this reliability via sending, UDP is more suitable. MIDP 1.0 does not support TCP, but MIDP 2.0 specifies TCP support, unfortunately it is only optional for manufacturers to include it. However, I think most new 3G phones support TCP now. Figure 2-6 shows the basic structure of TCP.

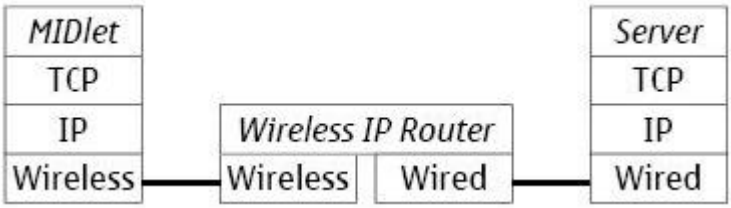


Figure 2-6: Basic structure of TCP

Wireless profiled TCP can be used to provide an optimized connection for the wireless networks, but it must support split and end to end modes of operation [8]. Figure 2-7 shows the structure of TCP over WAP.

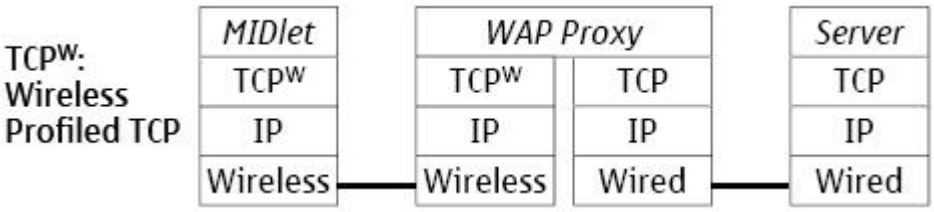


Figure 2-7: Structure of wireless profiled TCP

User Datagram Protocol (UDP) is an unreliable, packet-oriented protocol, which means data packets sent may be received once, more than once, never, or even in the wrong order and there is no connection between the communicating machines, each packet is sent independently. UDP is the simplest transport protocol of all: because UDP does not address reliability, it does not need to be profiled differently for wireless networks. UDP adds very little overhead to the communications and is generally the most efficient transport technology. Additionally, many applications don't need TCP's reliability; for example, when the data is time sensitive, as for video streaming, it's often better to simply lose a packet than to hold up subsequent packets while retrying a now-out-of-date packet. MIDP 1.0 does not include support for UDP. MIDP 2.0 specifies UDP support, but makes it optional for manufacturers to include it. The structure of UDP is showed in Figure 2-8.

Figure 2-8: Structure of UDP over wireless network

2.6 Types of Multiplayer Games

Games can be divided into types according to the game content. In most mobile games, the game content can be affected by the technology requirements. Nokia classifies mobile games in its document "Introduction to mobile game development"[9]. According to Nokia's documents, four main game types can cover all of the mobile games. They are Multiplayer Solo-Play games, Turn-Based Games, Act Whenever Games, and Slow Updated Games. Terraplay describes some game genera, such as racing game, first person shooting game (FPS), card games, and so on. These game genera can be regarded as branch types, which can be classified into the four main game types.

2.6.1 Multiplayer Solo-Play Games

These are games that feature rounds in which players each play a single-player game; their scores are compared at the end of the rounds. Generally the game does not utilize the network, so its speed is not restricted by network latency and fast, arcade-style games are possible. In these games, latency is only an issue at the end of the round, and even then it doesn't affect the results of the game. HTTP is sufficient for sending the score in this kind of game. However, the server has no way to contact the player when all of the

other players have completed their game – instead, clients must periodically "poll" the server with HTTP requests. In MIDP 2.0, this problem can be avoided by using TCP. In TCP, the server can send a message to clients any time, provided that the connection is open.

2.6.2 Turn-Based Games

Turn-based games proceed in discrete "turns," rather than continuously. Card games and chess games are included in this game type. Generally, this kind of game can afford long latency, but HTTP is not a good choice for them. HTTP has a major disadvantage, because there's no way for the game server to notify the MIDP client that it's the player's turn. Instead the client must poll, periodically asking the game server. While the client is waiting, some kind of "busy" indicator should be displayed, and the player should have a way to stop the game. With MIDP 2.0, TCP is probably the best choice. The server can tell players immediately when it's their turn to move. It's important that they can be reliably notified when it's their turn, so UDP is probably not suitable unless a positive acknowledgement and retry is added.

2.6.3 "Act Whenever" Games

These are games that run continuously over a longer period of time. During each game session, the players keep remain in control. Racing games and FPS games all belong to this game type. If there is no real-time action in the game, then HTTP may be a suitable networking technology. If the game involves even slow, real-time action, HTTP's latency will probably be too great. With MIDP 2.0, we should consider UDP, TCP, or a combination of them.

2.6.4 Slow Update Games

These are essentially a special case of "Act Whenever" games. The player can act at any time, and the game keeps running all the time. However, the player is participating continuously, even when not connected. When the player connects, he or she can see the state of the game and update the behavior rules for his/her agents in the game, but the agents act continuously in the game following those behavior rules. For this sort of game, latency is typically not an issue because the user's agent acts in real time, rather the user himself. Therefore, HTTP is usually sufficient. In this case, all the agents could be executing in the server.

Chapter 3

Gaming Performance

The success of multi-player mobile games is dependant on the gaming experience of players. Gaming performance is the result of different factors that can affect the gaming experience. The main factors are the physical attributes of networks, including mobile terminal processing ability, the properties of the air interface provided by wireless networks; and the game design. The basic mechanisms provided by Terraplay gateway and server are packet forwarding and client synchronization. Game developers can affect game performance by utilizing some specific design mechanisms. In this chapter I will simply introduce the effects of using a wireless network.

3.1 Latency

The main effect of using wireless networks for multiplayer games comes from two aspects: latency in cellular network and limited bandwidth. Among these, latency is the most important factor. Latency can determine the fluency of game reaction. In a wireless multiplayer game, fluency is the most important feature when evaluating game performance. In a multiplayer game, if the average network latency is too great, each player won't receive data from server in time. This will make the player feel it's difficult to manipulate objects in the game, since player can't see the results of their commands quickly enough.

In wireless networks, the main end-to-end latency usually comes from the air interface delay. This air interface includes both physical and link layers. It performs data transmission, frame transmission, handover, resource allocation, and so on. Generally, when the bit rate of the air interface is increased, this latency can be reduced significiantly. But this is not always true, as GPRS uses interleaving which slows transmissions independent of the signaling rate in a burst.

One of delay components is the time to set up and tear down a connection from the mobile station to the fixed network. When connection is set up, the network needs to reserve resources in the radio access network, in the core network, and in the transport network. These allocations obviously take time and are always present, independent of the air interface bit rate. For example, in GPRS networks, it takes 800 ms to handover and to allocate resources, after that, it takes another 200 ms to release these resources. In addition, to maximize the utilization of radio resources, it is a common practice to close the connections as soon as they are not in active use. This fast teardown of the radio connection may result in repeated channel setup and teardown times during each session, particularly in the case of GRPS/EGPRS connections in early networks. To avoid this

repeated channel setup and teardown during each session, there are features that will also keep the channel up for a short time after there are no packets waiting to be sent in the buffers. This will improve the distribution of round-trip times, particularly for connections with burst transmissions patterns.

Packet loss is another important factor that causes latency. In a wireless network, packet loss not only can cause latency, but can also cause disconnection from the game session. Packet losses can happen both in the initial handshaking phase and during the gaming phase. For a TCP connection, occasional high packet-loss may make the client disconnect from the game session. In generally, packet-loss will increase the average latency during the gaming period.

3.2 Bandwidth

The available bandwidth is dependant on the underlying wireless technologies. For multiplayer games on the Terraplay System, a client receives packets from all other clients, and this client also sends its own data to all of the other clients through the Terraplay gateway. Although multiplayer games can produce more traffic than SMS, the traffic is still limited compared with the maximum capacity of 3G and GPRS. The following table lists the maximum and average bit rates of different systems.

System		**GPRS** (per TSL=577 microsecond)	**EGPRS** (per TSL=577 microsecond)	**WCDMA** (384 kpbs)
Downlink bit rate [kbps]	**Max**	**20**	**59**	**384**
	Average	**10**	**32**	**200**
Uplink bit rate [kbps]	**Max**	**20**	**59**	**384**
	Average	**10**	**32**	**200**

Table 3-1: Indicative throughput values in mobile networks [10]

According to my game tests, the output of the multiplayer games less than 20 Kbps. Compared with the capability of 3GPP and EGPRS networks, bandwidth is not a very important issue for mobile games now. With the development of wireless technology, 3G networks can provide far greater bandwidth for data communication. Besides, the number of users of mobile multiplayer games in one game session is very small, so the traffic during the gaming period can be accepted. However, although the bandwidth is a factor in the performance of multiplayer games, the effect of bandwidth is not the most important factor now. The effects of the wireless network on the performance of multiplayer games are mainly due to its latency.

There are other factors that can affect multiplayer wireless games, such as access-point handling, signal strength, battery consumption, and so on. But the effects of these factors on multi-players games are **indirectly** factors.

Chapter 4

Game Analysis on The Terraplay System

In this chapter, I will analyze the factors affecting multiplayer games from the viewpoint of game developers. In the previous chapter I introduced the factors due to wireless networks. However, in multiplayer games, game design is also a factor that **may** affect the game performance. By analyzing some existing games on the Terraplay system, I attempt to discover the traffic model of multiplayer games on Terraplay system. The parameters in the traffic model form a game metric. The difference between game types will be reflected by the changes in these game metrics.

Identifying relevant metrics of wireless multiplayer games on Terraplay platforms was one of objectives in this thesis project. These mobile game metrics also form the foundation of my design for a Test Toolkit. In this chapter, I will describe these metrics based upon testing several different multiplayer game types. Terraplay's mobile games mainly belong to two types: Action Whenever Games and Turn-based Games.

4.1 Test Environment

The test environments are based on Terraplay's Move SDK 2.1[4]. Move SDK is an emulator for the Terraplay Move System. Terraplay's MOVE system provides a high quality, commercial mobile multi-player gaming platform for network operators. Terraplay Move is responsible for forwarding each packet that was produced by a mobile client to all of the other clients that are in the same game session. In these tests, I use two laptops as mobile game clients. The multiplayer mobile games run on J2ME Wireless Toolkit 2.2[5]. Other mobile emulators such as SonyEricsson SDK 2.2[11] or Nokia J2ME Developer's Suit 2.1 [12] could also be used. These SDKs can emulate all kinds of mobile phones on computers. I used ethereal to capture and filter all of the packets produced by these mobile clients. Ethereal was started first and capture data as game clients began to communicate with the game server. Ethereal recorded the traffic at the server side. The computer running the Move SDK is the game server. The game server's IP address is 212.247.178.228. Because Ethereal was running on the server side, the IP addresses of the two mobile clients running behind a NAT are the same, as the router's IP: 212.247.185.51, but the port numbers are different. In these tests, packets between two clients and the game server (including a lobby server and a gateway) were captured by Ethereal both on the Lobby Server port (5000) and Gateway port (3435). Figure 4-1 shows the setup of this test environment.

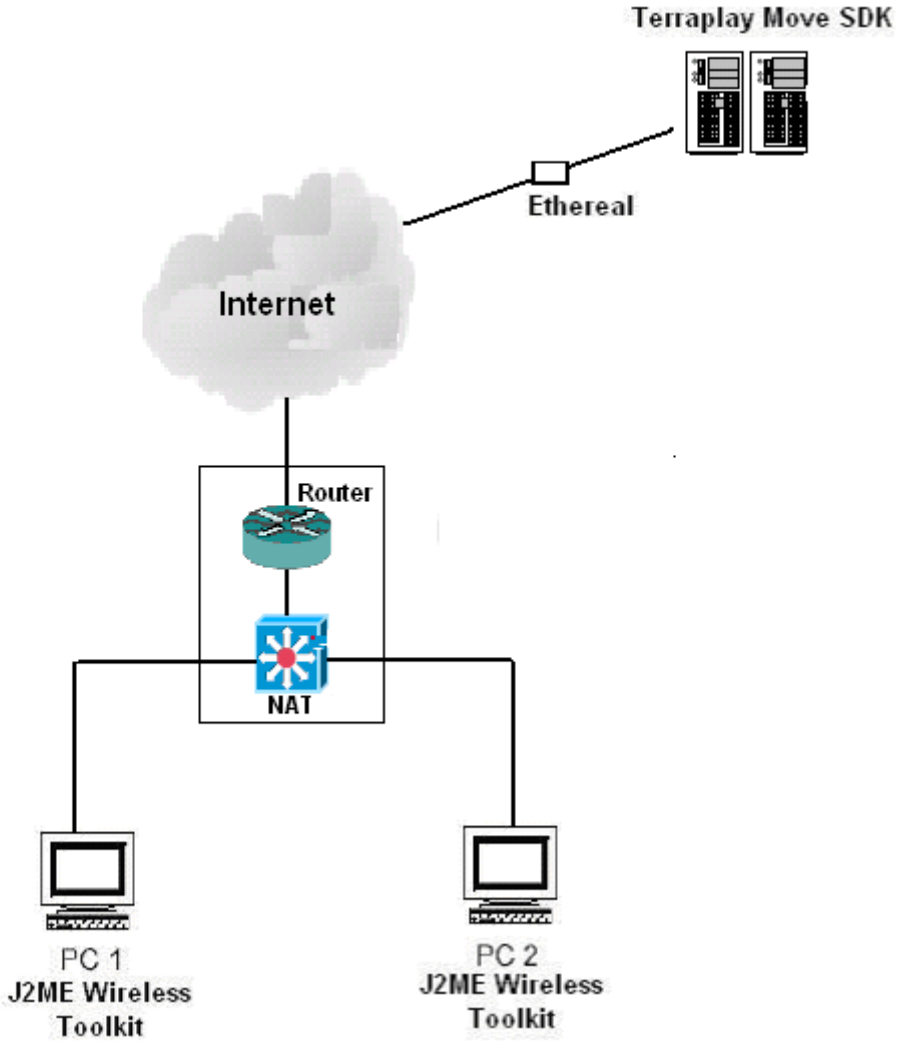


Figure 4-1: Setup of Game Test Environment

4.2 Test of Multiplayer Games

The tests were organized according to game types. Since Terraplay mainly provides Action Whenever games and Turn-based games, the games tested were Netbaby Grand Prix, Mole war, No Refuge, Lock'n'Load, Gumball Rally, and 5 In a Row. By analyzing the test data, I observed that the game traffic on the Terraplay System has the following characteristics that are important for creating the game metrics.

4.2.1 Diversity of Packets

The Terraplay System supports the multiplayer functions on the mobiles by sending game data as Terraplay messages.[13] During the gaming period, there are many kinds of messages such as: gwConnectionRequest, gwHeartBeatRequest, gwSendStreamObject-Notification, and so on. These messages implement the functions of ensuring that a player is still connected, transmitting game data, and so on. Messages details are in [14]. Although there are a variety of packets, only the message gwSendStreamObjectNotification contains actual game data that reflects the change in game states. Hence, this is the message we will focus on.

4.2.2 Single Mode of Multi-Player Games

All multi-player games on Terraplay platforms have the same game model. Terraplay's MOVE provides a lobby server where players can create game sessions, join sessions, and post high scores. Players joining a game discover each other, after joining a session. The game-play consists of one or several rounds. Before starting a round, players meet in a game room to decide upon properties of the next round (for example track or level). The rounds should be launched from the game room. Each round is a discrete entity within the game. The game room should be used to synchronize the start of the round. New players can join the game between rounds. A round consists of a series of updates. During an update all players can make a move. This move should subsequently be transmitted to all other players. A move could be a player requesting new cards in poker, hitting a golf ball, or updating the position of a racing car. Chess players alternate between making a move in each update cycle. The local game state should be updated based on all the relevant moves. For each update, all clients that have joined the same game session will get the same set of moves as all the other clients. The update cycle can be fast, as it is generally limited only by the latencies of the wireless network. The fact that all the games are based on the same game model makes it possible to evaluate different game by using a single. This is the foundation for defining game metrics.

4.2.3 Common Game Metrics

Based on the game model and analysis of game tests, I found "Avg. Packet Size" and "Avg. Packets/s" are two basic parameters. In the Terraplay system, the only packet that contains game content is the gwSendStreamObjectNotification. From my tests, I observed that the parameters of "Avg. Packet Size" and "Avg. Packets/Sec" accurately reflect the traffic characters of different genera on the Terrraplay System. For example, for most "Action Whenever Games" the "Avg. packets/sec" parameters is always higher, since all clients in the game session need to update their states frequently. But, most Turn-based games don't need very high "Avg. Packets/Sec", but they sometimes generate packets larger than the "Avg. Packet Size", especially for some card games. So from the observers' point of view, these two parameters "Avg. Packet Size" and "Avg. Packet/Sec" were selected as the traffic metrics of wireless multiplayer games on Terraplay's System. Examining the test data, I noticed there are two situations concerning these metric values. One situation is when the metric values are fixed during the whole gaming period. In my tests, racing games like Netbaby Grand Prix and Gumball Rall belong to this situation. Another situation is that the "Avg. Packet Size" and "Avg Packet/Sec" are not fixed and varies among clients. This situation often happened in some FPS, action games or strategy games. This is caused by the complexity of game content. These games need more content such as shooting with different weapens, adding blood, or weapons. All of these happen randomly. These random events cause variance in clients' taffic. To adress this situation, I select the largest values of the "Avg. Packet Size" and "Avg. Packet/Sec", since the largest one causes the greatest game latency. By analyzing the data from these game tests, I can set up a Traffic Model for each game (that I tested). This Traffic Model is that traffic description of those particular multiplayer mobile games.

4.3 Game Traffic Model

The Game Traffic Model sumarizes the results from game tests. The game tests reveal the traffic characteristic of different games; this is often hided behid the game content.

4.3.1 Action Whenever Games

Action whenever is the most popular game type. It includes a lot of different game genera such as racing games, first person shooting (FPS) games, and so on. These games generally produce much more traffic during game sessions. Additionally, they are more sensitive to latency than other types of games.

4.3.1.1 Racing Games

According to Terraplay a “Racing game is any game that involves competing in races through a surrogate playing piece or vehicle, either getting it from one point to another or completing a number of circuits in the shortest time.”[15] Racing games are very popular on traditional game platforms (PC). On new game platforms: such as mobile phones, racing games are also showing strong signs of success. Currently, multiplayer wireless racing is the biggest game genra. In my tests, Netbaby Grand Prix (a java game) and Gumball Rally (a Mophun game) were two examples classic racing games. Racing games generally are highly sensitive to game latency. Although the two games use different technologies, they have the same traffic charactersitics on Terraplay’s MOVE platform. The Avg. packet rate of Netbaby is 2.4 packets/s and Gumball Rally is 0.65 packets/s. Figure 4-2 and Figure 4-3 show that these two games follow the same basic traffic models. The main difference is that Netbaby sends moves at four times the rate of Gumball Rally.

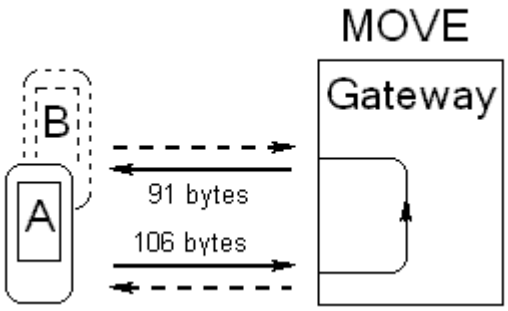


Figure 4-2: Traffic Model of Netbaby per move

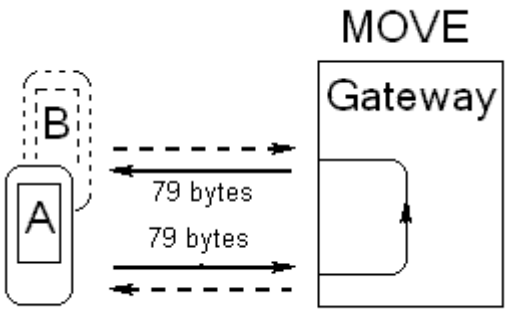


Figure 4-3: Traffic Model of Gumball Rally per move

4.3.1.2 Action and First Person Shooting (FPS) games

Action games typically feature violent physical force, especially shooting, as their main interactive feature. A first-person shooter (FPS) is a game where the player's on-screen view of the game world simulates that of the character, and there is a high percentage of combat involved. First-person shooter games can be considered a sub-genre of Action games.Compared with other kinds of action games, FPS games have a higher requirement to quickly react to a player's command. Lock'n'Load and Mole War are two classic Action and FPS games. As noted earlier, action games involve more complicated content, and their random events cause high variance in clients' traffic. This variance should be considered when analyzing game metrics. The avg packets rate of Mole War is 2.5 packets/sec, and Lock'n'Load is about 3 packets/sec. Figure 4-4 and Figure 4-5 again show that the basic traffic model of these two action games. Here, we noticed that rate of moves is very similar.

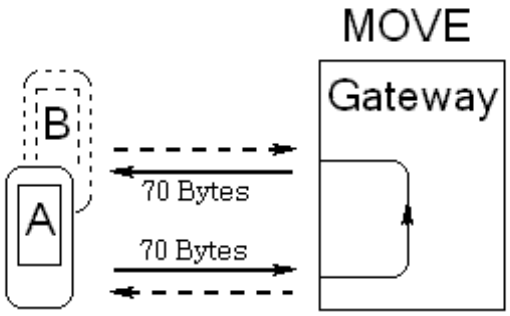


Figure 4-4: Traffic Models of Mole War Per Move

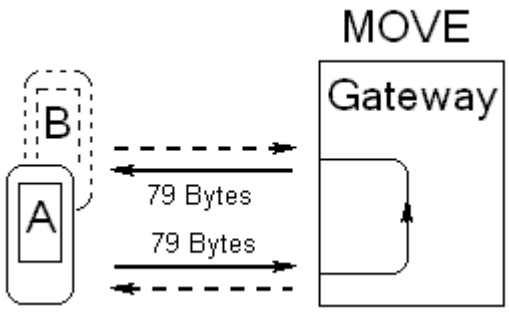


Figure 4-5: Traffic Model of Lock'n'Load per move

4.3.1.3 Strategy Game

Strategy games are games in which the players' decision-making skills have great significance in determining the outcome. No Refuge is a successful strategy game. The Avg. Packets rate of No Refuge is only 0.6 packets/sec. Figure 4-6 shows the per move traffic model of No Refuge.

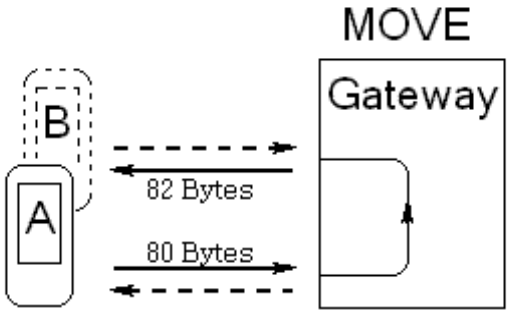


Figure 4-6: Traffic Models of No Refuge per move

4.3.2 Turn Based Game

Turn based games are also called "turn-based strategy games". A player of a turn-based game is allowed a period of analysis before committing to a game action. Most card games and chess games belong to this types of game. "5 in a row" is a popular Turn based game on the Terraplay System. The Avg. Packets rate is 0.055 packets/sec. Figure 4-7 illustrates the per move traffic of "5 in a row".

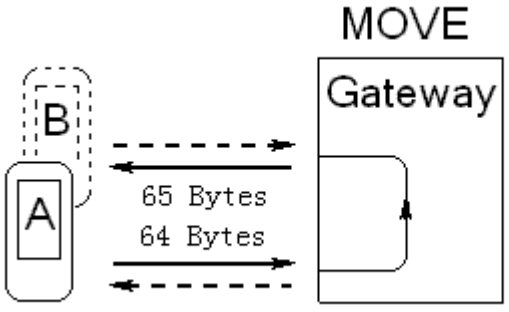


Figure 4-7: Traffic Models of "5 in a row" per move

4.4 Summary of Game Tests

From these tests, it's clear that the game metrics Avg. Packets rate and Avg. Packet Size have a close relation with the game types. This is summarized in Table 4-1. On the other hand, by modifying the values of these game metrics, game developers also have the possibility to affect the game performance. How these factors affect the game performance are important issues on today's Terraplay Systems.

Game	**Avg. Packet Size (bytes)** player making move/ Other player(s)	**Avg. Packet Rate (per sec)**	**Avg. Bit rate (kbps) Per Move**
Netbaby	106 / 91	2.4	0.288
Gumball Rally	79 / 79	0.65	0.1027
Mole War	70 / 70	2.5	0.35
Lock'n'Load	79/79	3	0.474
No Refuge	80/82	0.6	0.0972
5 In a Row	64/65	0.055	0.007095

Table 4-1: Summary of Test Games

Chapter 5

Implementation of Terraplay Test Toolkit

The Terraplay Test Toolkit (TTK) is an emulator developed as part of this thesis project. In order to test game performance in a wireless network，I needed a toolkit to emulate multiplayer games and record the round trip latency. This Test Toolkit was developed exactly for this purpose. It can measure the latency in a wireless network. As a game test toolkit, TTK should be accurate and convenient. Accuracy means TTK should use the same mechanisms as any multiplayer games that will utilize the Terraplay system. Convenience means TTK should be easy to operate and have all the necessary functions such as calculation, storage, etc. The following section summarizes this Test Toolkit. More details about developing and using are included as Appendix A and Appendix B.

5.1 Architecture and Infrastructure

The architecture of the Terraplay Test Toolkit (TTK) emulates a two-client environment. TTK runs these two clients on the same mobile phone. The physical route of packet transmission is round trip. There are two reasons for running the two clients on the same mobile phone. The first reason is that TTK needs to calculate the latency according to the mobile's local clock. The other reason is the convenience, since all the tests can be performed using a single phone. The mobile phone communicates with the Terraplay Lobby Server and Terraplay Gateway via a wireless network. In order to supports some functions such as the remote database, a web sever is also necessary. Figure 5-1 shows the architecture of the Terraplay Test Toolkit.

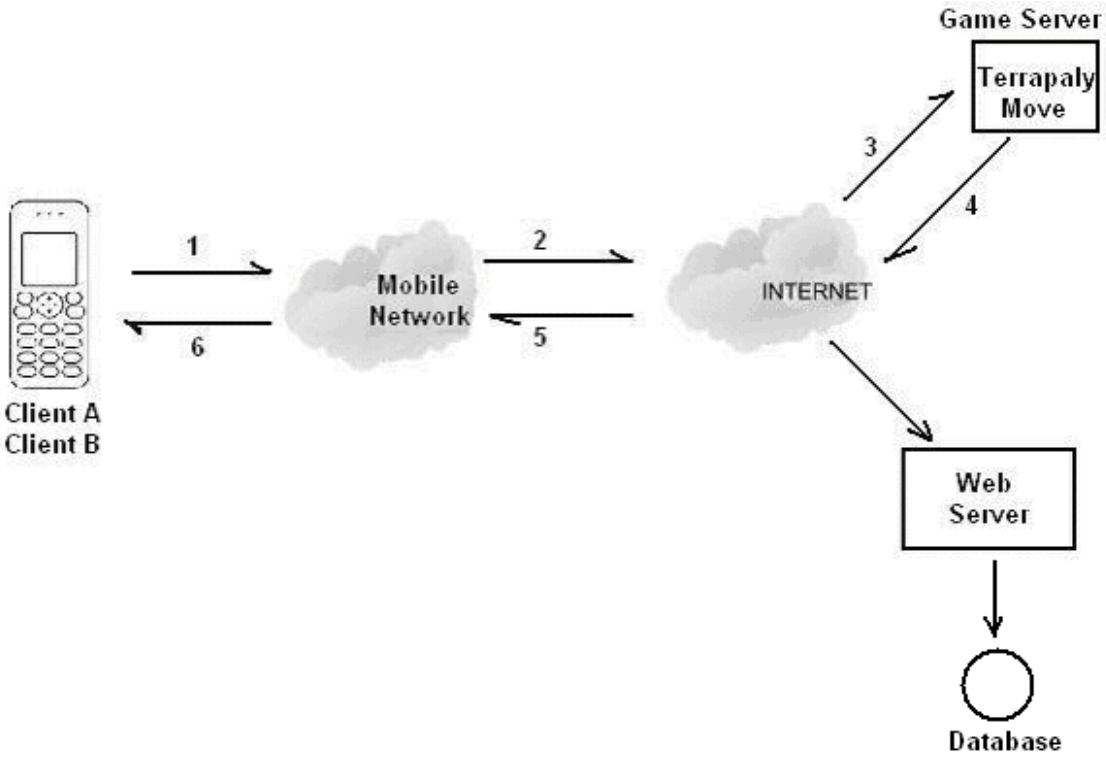


Figure 5-1: Architecture of Terraplay Test Toolkit

In this architecture, client A initiates the game session; while Client B joins the existing game session. When a packet is sent from client A, the “sent time” is recorded in a timestamp. When Terraplay Move forwards this move to the client B, the “received time” is also recorded. The latency that TTK needs to measure is the end-to-end latency, because the relevant latency in the mobile multiplayer games is also an end-to-end latency. So the latency of each move is calculated as: Latency= Received time-Sent time. This end-to-end latency is mainly made up of the latencies in mobile network, Internet, and the latency of the Terraplay MOVE system itself. The latency of Terraplay MOVE is less than 30ms. [1]

5.1.1 Game Server-Terraplay MOVE

As a commercial mobile multi-player gaming platform, Terraplay MOVE tries to optimize network resource utilization and game performance, as the main driver for network operators and Game Service Providers to profit by providing superior quality.

MOVE’s architecture does not interfere with the game logic, thus the network server is generic to all games. All game logic is in the game itself, and game synchronization can be based on a game client/server or peer-to-peer (P2P) model. Developers need not be bogged down with networking hassles. MOVE is intended to give them creative freedom by providing networking APIs to create games at a much lower cost and in a resource effective way.

MOVE delivers a package for mobile multi-player gaming service with supplementary services such as game matchmaking, TCP/IP in-game communication, high scores presentation, service message capabilities, session isolation, and monitoring capabilities for accurate gaming statistics.

The key features that MOVE architecture has are: [4]

- ✓ Content independent
- ✓ Generic platform
- ✓ Independent of operator's network
- ✓ Support a wide range of business models
- ✓ Cross platform (mobile vs. PC vs. console)
- ✓ Scalable
- ✓ Lobby functionality
- ✓ Intelligent multicasting

The MOVE architecture is designed to be composed of the Terraplay System (TPS), which is the heart of the system and is designed to handle large numbers of players on any client platform including both PCs and PlayStation2. The MOVE architecture extends the TPS with the MOVE Gateway (MGW). MOVE makes use of the Terraplay Wireless Protocol (TWP), which is a binary protocol that aims to help the game developer to utilize the in-game and lobby network resources of Terraplay's MOVE server. Since these resources are multi-media stream, binary protocol is necessary. The MOVE architecture also consists of a MOVE Lobby Server (LBS) that provides basic functionality for mobile terminals that intend to connect to a game session.

Figure 5-2 illustrates MOVE's functionality for clients that try to connect and request gaming services. The presence of the MOVE Gateway (MGW) provides access from MOVE to the Game Access Server (GAS) server in TPS where the data transfer takes place, while the MOVE lobby takes the responsibility to gather all the clients prior to a game or tournament session. The Terraplay Games Access Server (GAS) routes and prioritizes games data between players (peer-to-peer) or players and the games content server (client-server). It is designed to ensure that the critical game data is exchanged between players regardless of the network connection method they use. This data communications model is however particularly relevant to the wireless environment where bandwidth is limited and can be subject to significant temporary constraints, such as network congestion. O&M Console and Terraplay Management System (TMS) provide session control functionality. They allow efficient O&M with centralized alarm handling. Besides, they allocate session resources in the network and provide game statistics.

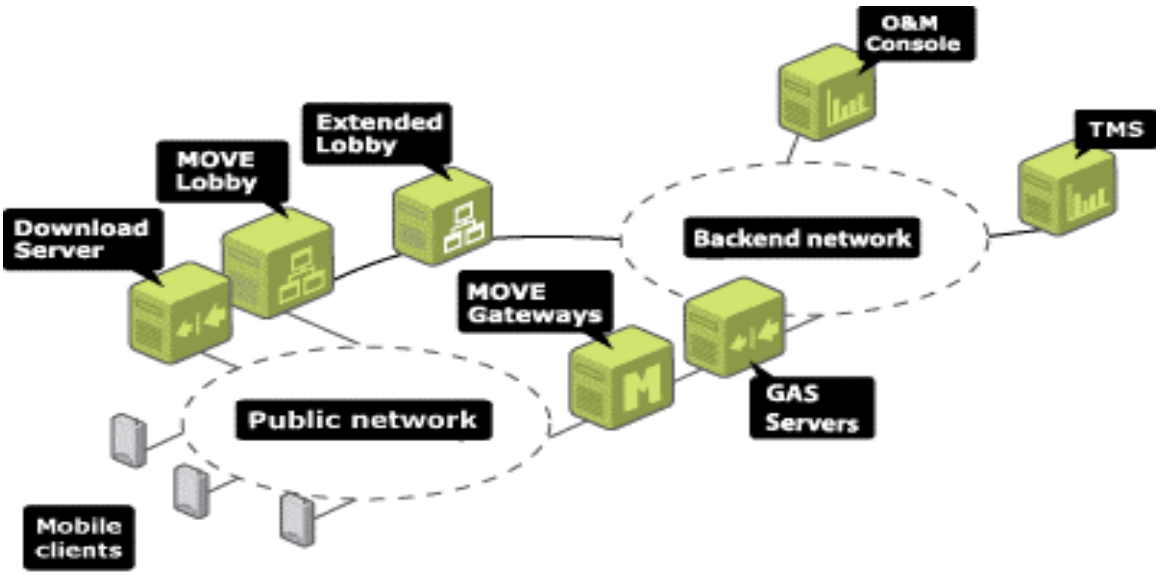


Figure 5-2: Terraplay MOVE Gaming services overview [15]

5.1.2 Web Server

A servlet running in the web server provides supplementary enterprise functions. The location of this web server has been introduced in Figure 5-1. The mobile phone can communicate with this web server, and sends all the latency values to this web server after testing. The servlet connects to an Access database by using Java Database Connectivity (JDBC) [16] technology, all the latency values that it receives will be stored in this database. The Web server that I used is Tomcat 5.0[17].

5.2 Functions

The functions of the Test Toolkit are to measure the end-to-end network latency in order to estimate the gaming performance in a wireless network environment, when utilizing the Terraplay MOVE. This performance can be evaluated under different conditions and for different purposes.

5.2.1 Uploading Scores

Uploading a score is used to test the performance of HTTP and TCP connections. The Test Toolkit uploads a number to the Terraplay Lobby server by choosing one of these two protocols. After finishing the score uploading, the latency of the current wireless network environment is calculated on the mobile. By using this test, we can compare the performance of HTTP and TCP over different networks and in different conditions.

5.2.2 Emulate

Emulate is used to emulate multiplayer games and test the comprehensive performance under various conditions (such as with a specific Avg. packet size, sending interval of packets, and so on). It communicates with the Terraplay gateway and records the RTT latency.

5.3 Summary of Terraplay Test Toolkit

5.3.1 Accuracy of Terraplay Test Toolkit

As a test toolkit, accuracy is fundamental. Accuracy here means the TTK should work in the same way as any multiplayer games on the Terraplay platform. In order to achieve this aim, the Terraplay Test Toolkit strictly follows the Terraplay game model. This means TTK does the same things as all the other multiplayer games during the gaming period. During the gaming period, the working states of TTK includes: Idle state, initiate state, lobby state, game room state, active state, and finish state. There are no additional states and actions in TTK, and the consequences of the actions are also the same as those games. The inner mechanism is shown in Appendix A. Besides this mechanism, Terraplay Test Toolkit uses the two game metrics, namely Avg. Packet Size and Avg. Packets rate as the parameters to define the traffic. From my previous analysis, these two metrics have a close relation with the game genera. By changing the values of these two parameters, game developers can measure how latency varies according to the change of game concepts. These measurements will be the basis for performance analysis.

5.3.2 Convenience of Terraplay Test Toolkit

As a test toolkit, TTK should be able to work on most mobile phones. The Terrapaly Test Toolkit strictly follow by the JWTI standard, that means this TTK can be installed and will work on all of the GPRS and 3G mobile phones without any modifications. TTK can measure the latency of each packet, and automatically calculates the average latency during the test period. It also shows the largest and smallest latency values when we stop the test. Additionally, TTK has a complete set of storage functions. TTK sets up a Record Management System (RMS) on the mobile phone. This RMS is a special permanent database system on the mobile phone. It can store the general information of each test. However, the functionality of RMS is very limited since RMS is restricted by the limited storage of mobile phone. Hence, I can’t store all the test data on a mobile phone. In order to analyze the latency variance accurately, I set up a web server on a computer, the TTK can communicate with that computer directly via a mobile phone, and thus we can store the latency values of each packet into a database system such Access, MySQL, and so on. Figure 5-1 shows this structure. It’s convenient for post-processing by using tools like such as Matlab or Excel. More details about how to use TTK are included in Appendix B.

Chapter 6

Performance Experiments

In this chapter, I will use the Terraplay Test Toolkit to do experiments under different conditions. These experiments will show how game performance is affected by different factors. After analysis of this data I will draw some conclusions.

6.1 Experimental Objectives

The objective of these experiments is to accumulate data for evaluating the performance of multiplayer game concepts over WCDMA networks. These records can be used as a reference for game developers when they design their games. The experiment examines the following points:

Performance analysis with varying Avg. packet size
Performance analysis with varying Avg. Packets rate (numbers/Sec)
Performance difference between TCP and HTTP connections

6.2 Experiment Environment

All the experiments use the Vodafone WCDMA network in Stockholm. The terminal used in my experiments was a SonyEricsson1010 [18]. Vodafone is one of the 3G service providers in Sweden. The experiments were divided into four groups according to the experiment's objectives. Since game players often use games as a kind of relaxing activity, this means that multiplayer games will be used in different environments. This environment probably has important effects on the performance of multiplayer games. I need to access the performance by doing my experiments under different conditions.

In all of my experiments, the procedure for my test was same. I defined the TTK parameters. Then the configured TTK communicated with a Terraplay Move server through Vodafone's WCDMA network. Following communication with the Terraplay Move server, TTK reports the test results and sends all the test data to a database that is connected to a web server. Figure 6-1 shows the structure of my experiments.

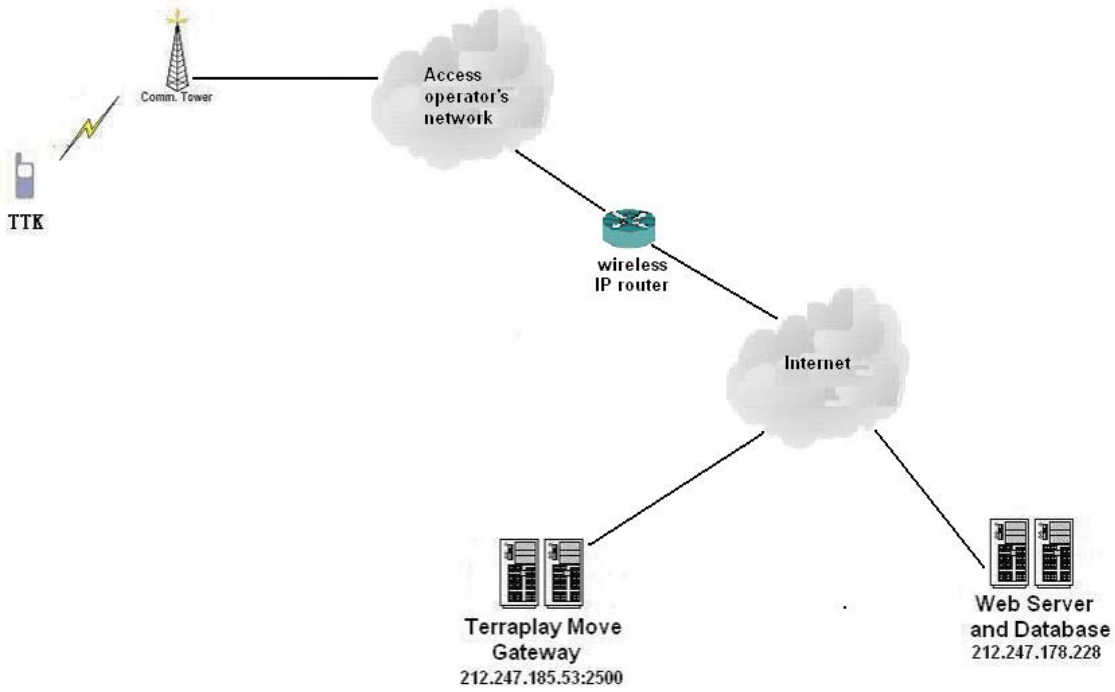


Figure 6-1: TTK Experiment Environment

In my experiments, I used the TCP protocol to transmit packets. TCP is a reliable transmission protocol. Figure 6.2 shows the structure of the TCP protocol over wireless networks.

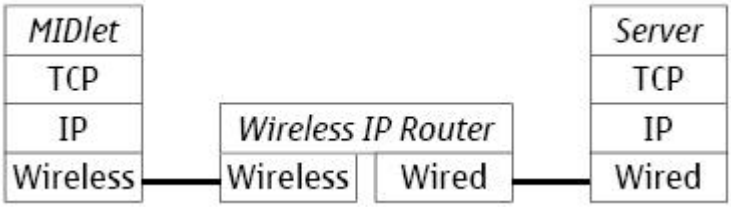


Figure 6-2: TCP over Wireless networks

The Mobile phone sets up a connection with a game server. In my experiment, the MIDlet on the mobile phone is the Terraplay Test Tookit (TTK). The game server that TTK will communicate is the Terraplay Move Server.

6.3 Analysis of These Performance Experiments

The analysis was based on the objectives of each group of experiments. From the analysis of each group of experiments, I will examine those factors affecting the performance of wireless multiplayer games. These experiments provided the basis for my conclusions.

6.3.1 Performance analysis while varying the Avg. packet size

In my previous game analysis, I found that the Avg. packet size was one of the important game metrics. On the Terraplay platform, game data are transmitted among players by

encapsulating them in a gwSendStreamObejectNotification packet. The packet size determines the capacity of packets. The structure of a gwSendStreamObejectNotification packet is made up of two parts: one part is the message header. The other part is the data part. The length of message header is fixed at 76 bytes. The length of the data part is defined by the user. For example, if the data part of the packet is at 24 bytes, then the whole packet size is 100 bytes. Figure 6.3 shows the structure of gwSendStreamObejectNotification message in my tests.

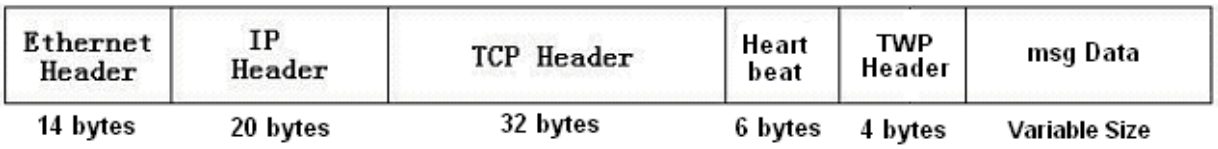


Figure 6-3: Structure of gwSendStreamObjectNotification

The Ethernet Header has 14 bytes, the IP header has 20 bytes, the TCP header has 32 bytes, the heartbeat has 6 bytes. The Terraplay Wireless Protocol Header (TWP Header)[14] part includes 1 byte of TWNP, 1 byte of Message Code, and 2 bytes of RequestID. The msg Data part is the data part.

The packet size in my experiments varies from 100 bytes to 300 bytes. Currently the packet size of most multiplayer games is less than 150 bytes. So I choose 300 bytes as the biggest packet size to cover all the current and near future games for analyzing. By changing the size of a packet, I can see if the packet size affects the game performance. I varied the packet size from 100 bytes to 300 bytes, while the parameter named “Sending Interval” was fixed to 400ms between packets. The test results are based on averaging 10 tests. This average latency and standard deviation (STDEV) of the each group is also shown in Table 6-1. The standard deviation is a statistic that tells you how tightly all the various examples are clustered around the mean in a set of data. All of the tests used a TCP connection.

	100 bytes	150 bytes	200 bytes	250 bytes	300 bytes
1	593	611	623	615	581
2	599	622	582	610	574
3	560	601	597	621	643
4	600	587	642	600	656
5	597	631	630	606	621
6	619	597	569	593	617
7	620	614	587	592	596
8	632	581	581	637	606
9	598	632	592	613	593
10	597	627	618	598	571
Average Latency	602	610	602	609	606
STDEV (ms)	20	18	24	14	29

Table 6-1: Individual Tests and Average Latency with different Packet Sizes (with 400 ms between packets)

Figure 6-4 Shows the STDEV distributions of latency as the packet size increase from 100 to 300 bytes.

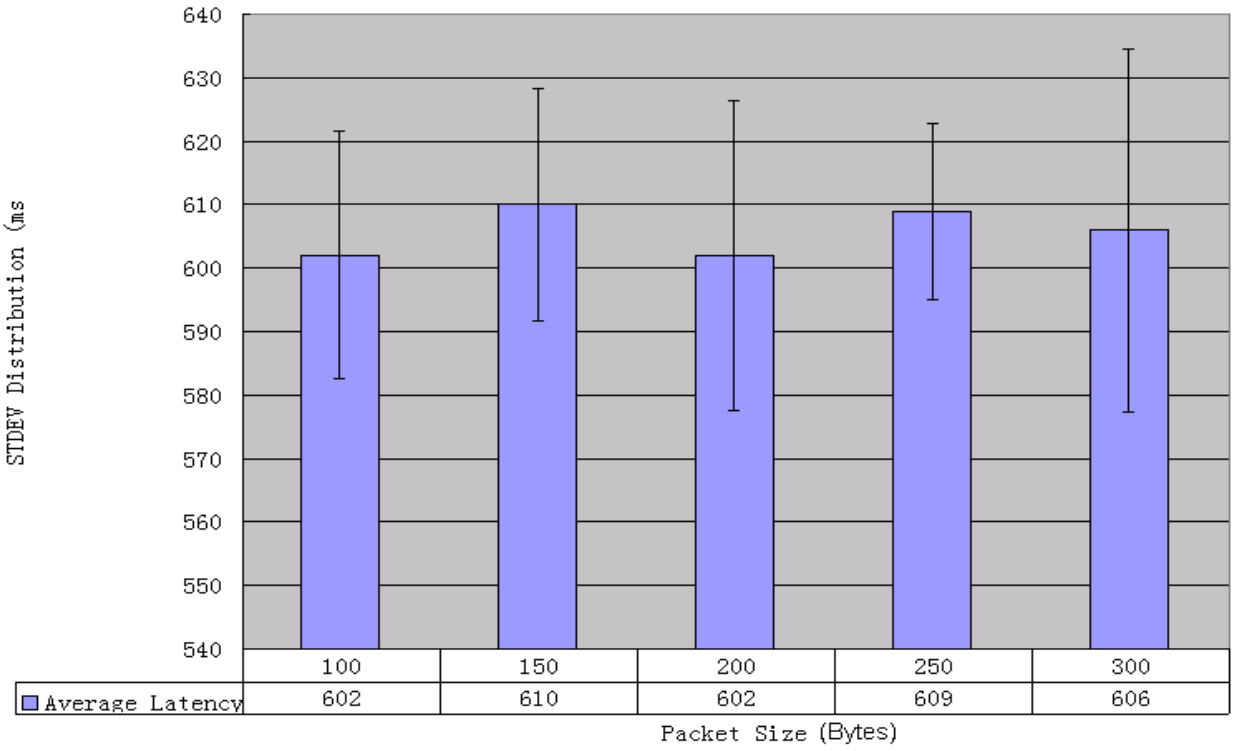


Figure 6-4 STDEV Distribution of Latency When Packet Size Changes

Figure 6-5 shows the RTT latency as the packet size increases from 100 to 300 bytes.

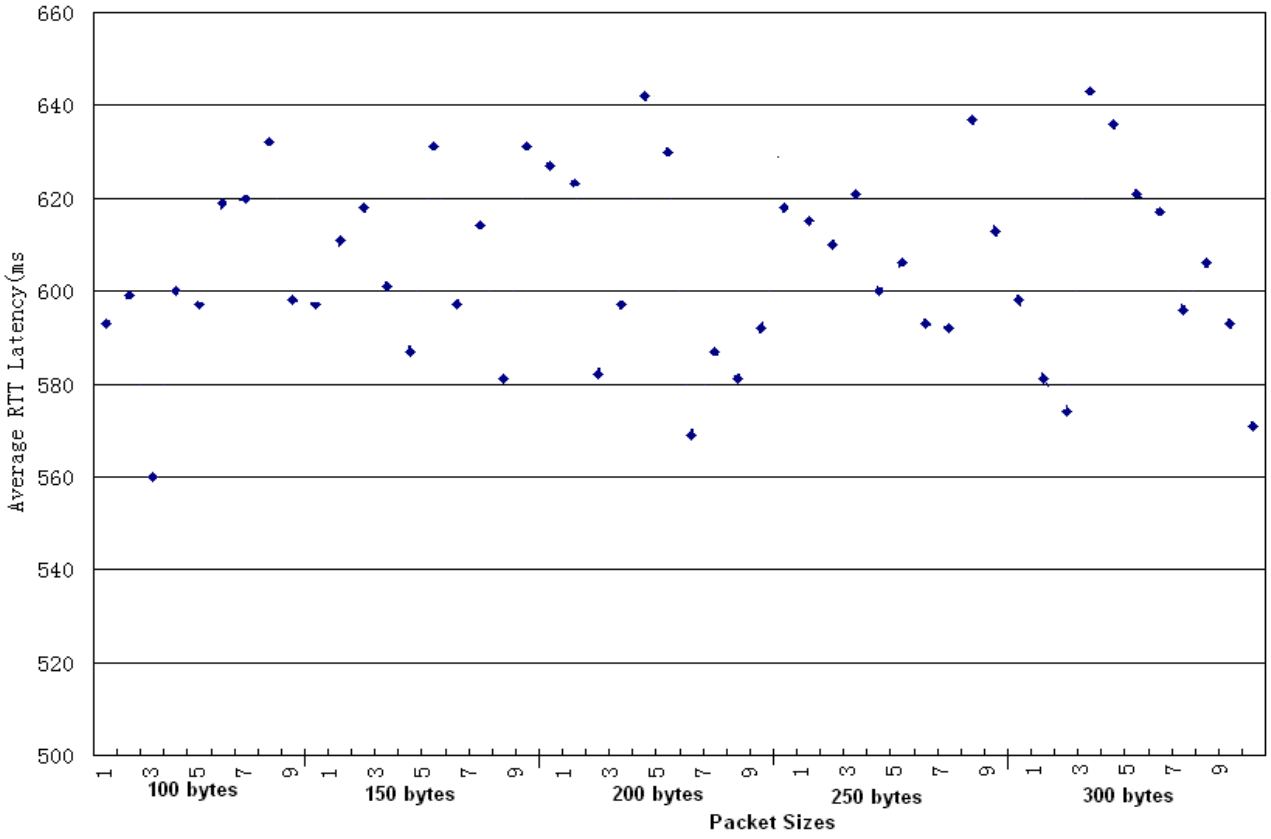


Figure 6-5: RTT Latency When Packet Size Changes

From the Figure 6-4 and 6-5, one can see that the latency values are nearly stable (560ms~660ms) independent of the packets size. Figure 6-6 shows the latency distribution when the packet size is 100 bytes based on 3000 100-byte packets. Figure 6-7 shows the latency distribution when the packet size is 300 bytes. Table 6.2 listed the summary of the latency distribution.

		< 400	400~450	450~500	500~550	550~600	600~650
100 bytes	Packets	104	331	320	359	513	808
	proportion	0.0347	0.1104	0.1067	0.1197	0.1711	0.2694
300 bytes	Packets	38	243	356	409	420	750
	proportion	0.012	0.0785	0.115	0.132	0.135	0.2422

		650~700	700~750	750~800	800~900	900~1000	> 1000
100 bytes	packets	353	78	21	45	21	44
	proportion	0.1177	0.026	0.007	0.015	0.007	0.01467
300 bytes	packets	561	174	17	39	31	88
	proportion	0.1812	0.0562	0.0055	0.0126	0.01	0.0284

Table 6-2: Summary of Latency distribution for two Packet Sizes

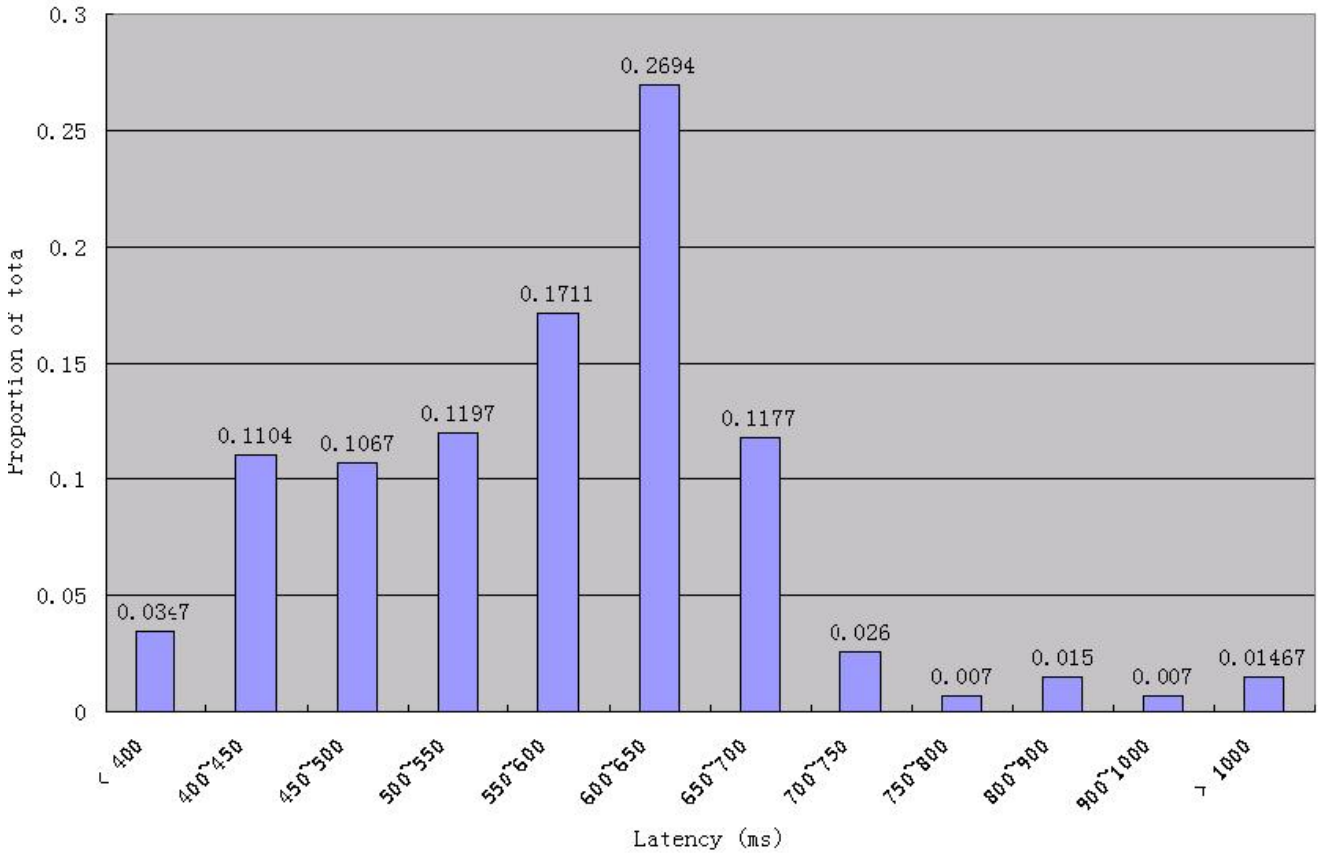


Figure 6-6: Proportion of Latency when packet size is 100 bytes

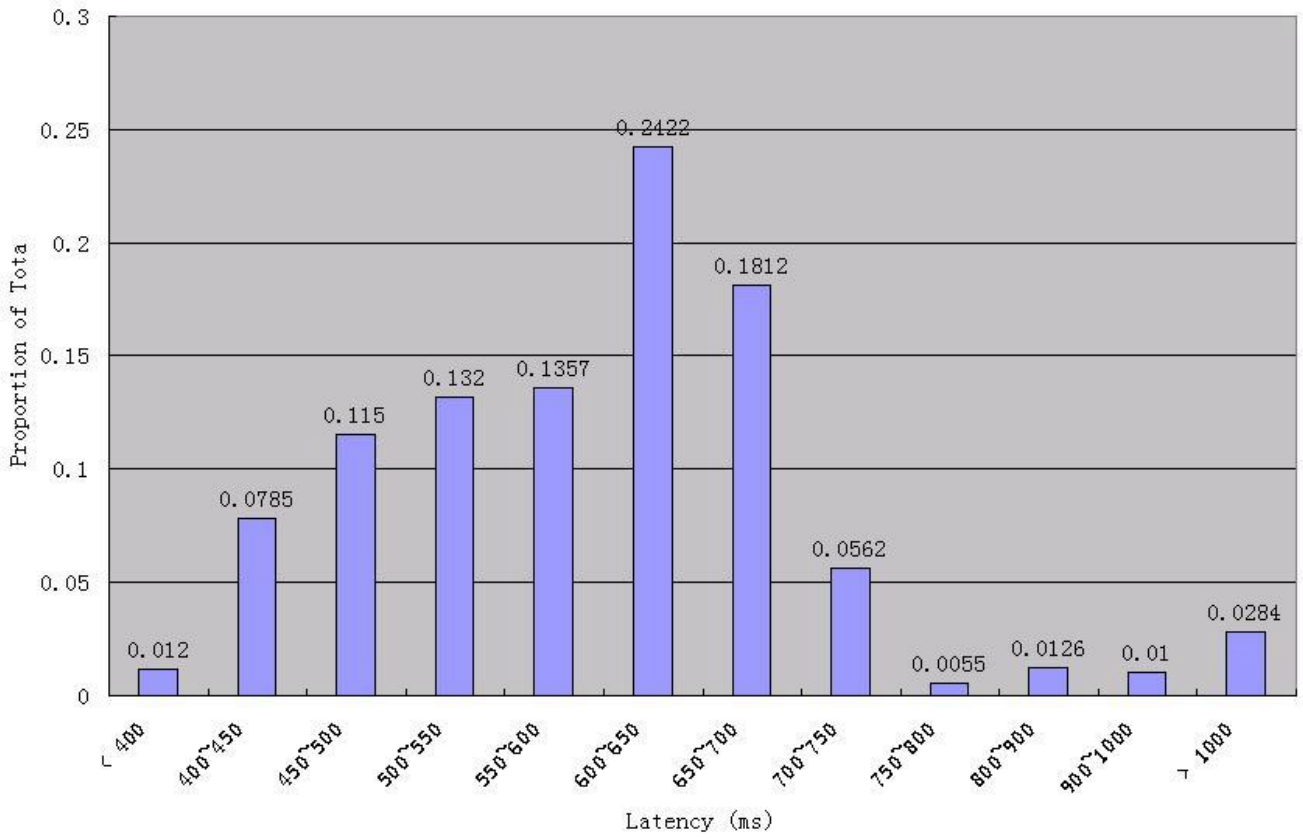


Figure 6-7: Proportion of Latency when packet size is 300 bytes

From Figure 6-6 and 6-7, we can see that the latency distributions are basically the same. Combining Figures 6-4, 6-5, 6-6, and 6-7 I observed two facts of Terraplay System. One is the “Average RTT Latency” is almost constantly, around 600 ms, another is that the latency distribution is nearly independent of packet size. These facts indicate that within this range the packet size has no obvious effect on the game performance. The reason is that the bandwidth of WCDMA is sufficient for gaming traffic. When the “Avg. Packet Size” is 100 bytes, the uplink bandwidth is about 250 bytes/second. When the “Avg. Packet Size” is 300 bytes, the uplink bandwidth is 750 bytes/second. Comparing with the capability of WCDMA, there is more than sufficient bandwidth, since for each of our wireless multiplayer games, the average packet size is smaller than 300 bytes, thus the average packet size was not an important factor in the Average RTT latency of wireless multiplayer games.

I also noticed that at different times and in different locations the average RTT latency values may differ. However, the results are basically the same.

6.3.2 Performance analysis with varying with “Sending Interval”

The packet rate (“Avg. number of Packets/Sec”) is another game metric. The “Sending Interval” (which is reciprocal of the rate) is the TTK parameter to test this performance. There are three reasons why I use "sending interval" instead of "Packet rate" as the parameter. The first reason is that the mobile phone calculates time according to its local clock, Local clock is in terms of milliseconds. “Sending Interval” directly uses millisecond as the input, while “packet rate” need to be converted. For example, a packet every

700ms is about 1.42857 packets/ second, every 800ms about 1.25 packets/second. The second reason is that J2ME is a kind of JAVA technology; Integer type data uses much less memory than Float or Double data types. For a mobile, memory is a kind of very limited resource. So it's very important to save memory. The third reason is that using float or double type data will cause accuracy lose, which will hurt the accuracy of calculating the game latency. When I was doing these tests, the "Avg. Packet Size" was fixed at the 100 bytes, while the "Sending Interval" was changed from 100 ms to 1000 ms. The test procedure was the same as what I tested the effect of variations in "packet size". The purpose of these tests was to see how the latency varies as the "sending interval" changed. The values of "Sending Interval" in the experiments were divided into 10 groups. In each group, there were 10 tests. Each test is based upon roughly 300 packets. According to my experience, the number of 300 packets is closed to number of packet in a game session. Table 6.4 shows all of the test results.

Sending Interval	100ms	200ms	300ms	400ms	500ms	600ms	700ms	800ms	900ms	1000ms
1	734	627	582	594	634	614	668	613	636	793
2	722	580	535	589	631	630	611	603	634	806
3	725	625	564	597	578	627	660	623	629	768
4	729	559	589	636	661	666	636	614	612	783
5	723	560	555	555	680	586	637	671	623	818
6	739	571	580	595	661	573	612	642	618	734
7	731	583	559	628	670	565	606	611	610	826
8	741	585	577	635	649	601	645	606	594	836
9	746	629	538	592	673	613	632	622	661	758
10	730	580	555	608	631	640	611	604	595	761
Average	732	590	563	603	647	612	632	621	621	788
STDEV	8	27	18	25	30	31	22	21	20	33

Table 6-3: Latency (in ms) for different sending intervals

Following these tests, I can found that the Average RTT latency varies when the "sending interval" is changed. Figure 6-8, Figure 6-9 and Figure 6-10 show the latency variance when the sending interval is 100ms, 300ms and 1000ms.

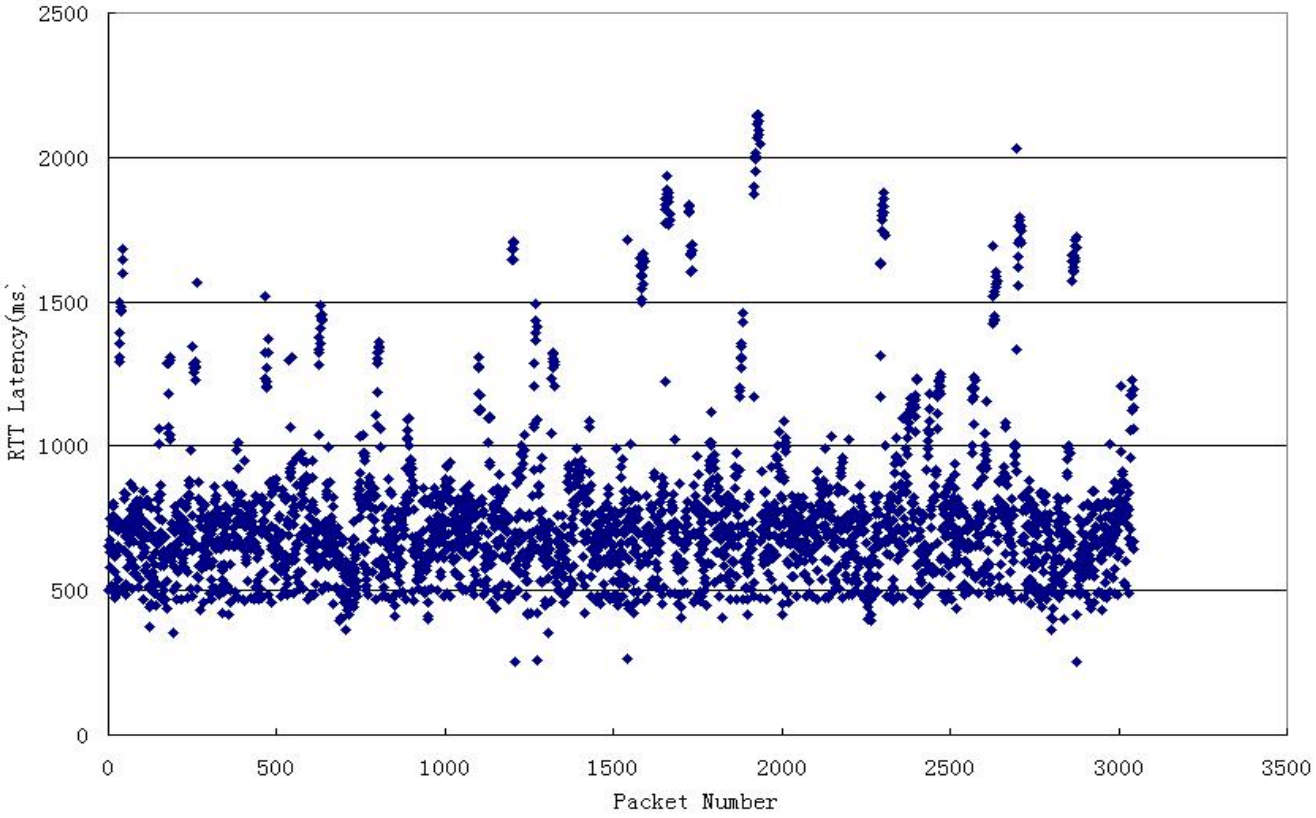


Figure 6-8: RTT Latency when the interval is 100ms

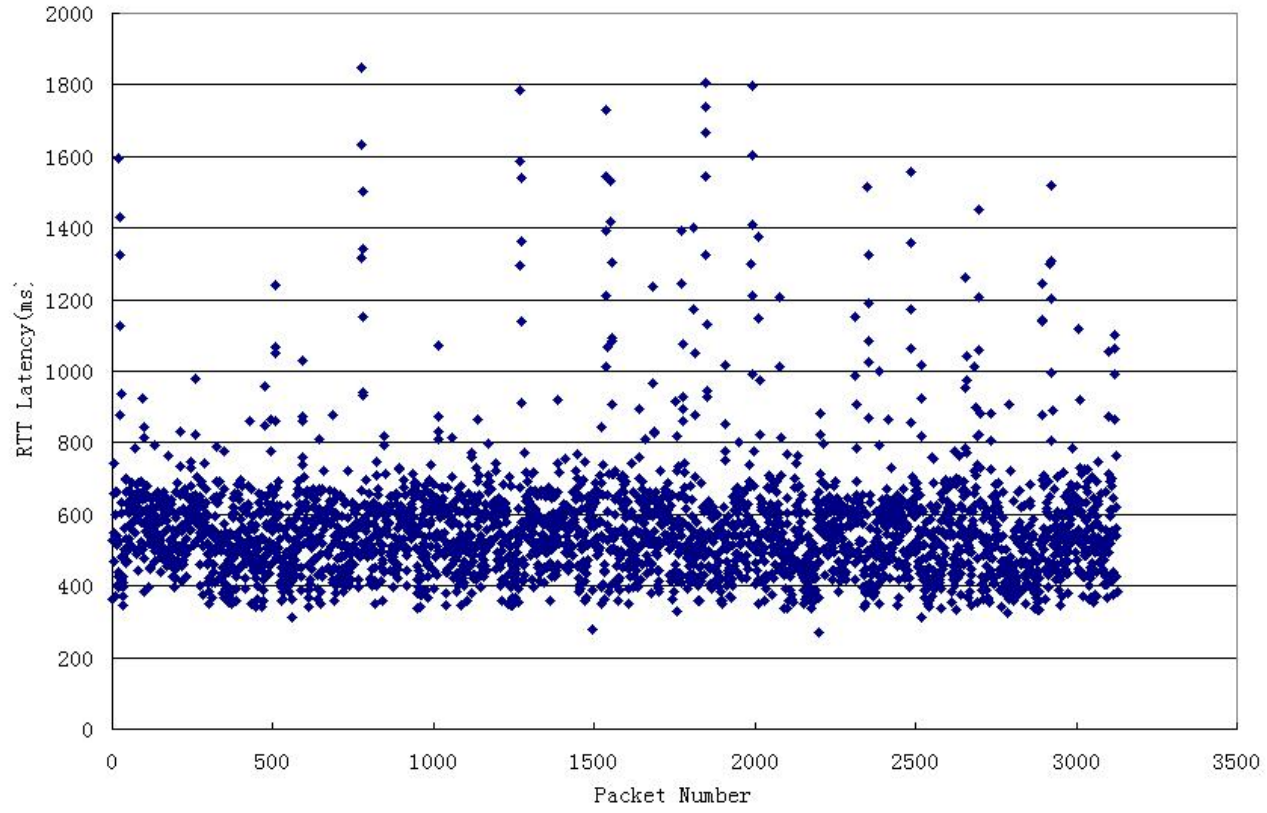


Figure 6-9: RTT Latency when the interval is 300ms

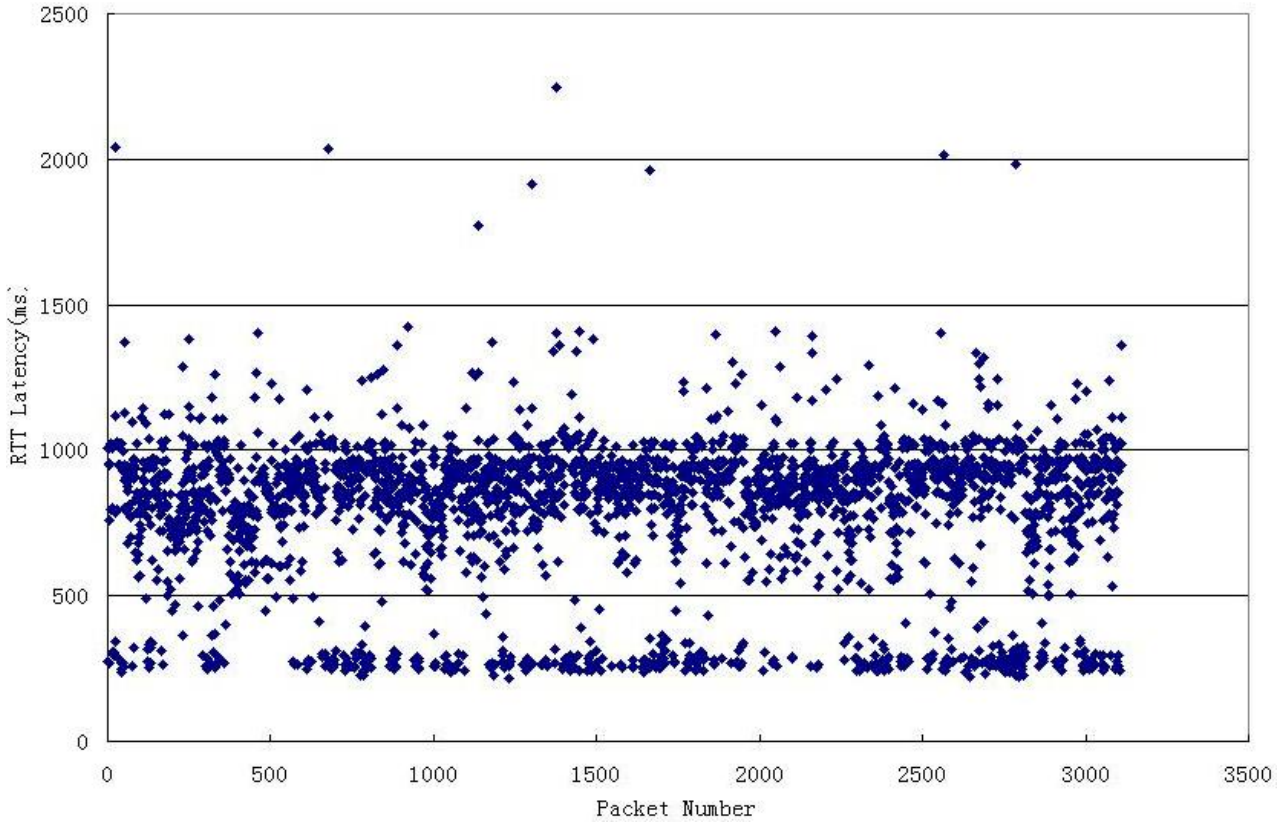


Figure 6-10: RTT Latency when the interval is 1000ms

Figure 6-11 an 6-12 shows the result of table 6.3. From this figure, I can see the trend of average latency variance as the “sending interval” increases from 100ms to 1000ms.

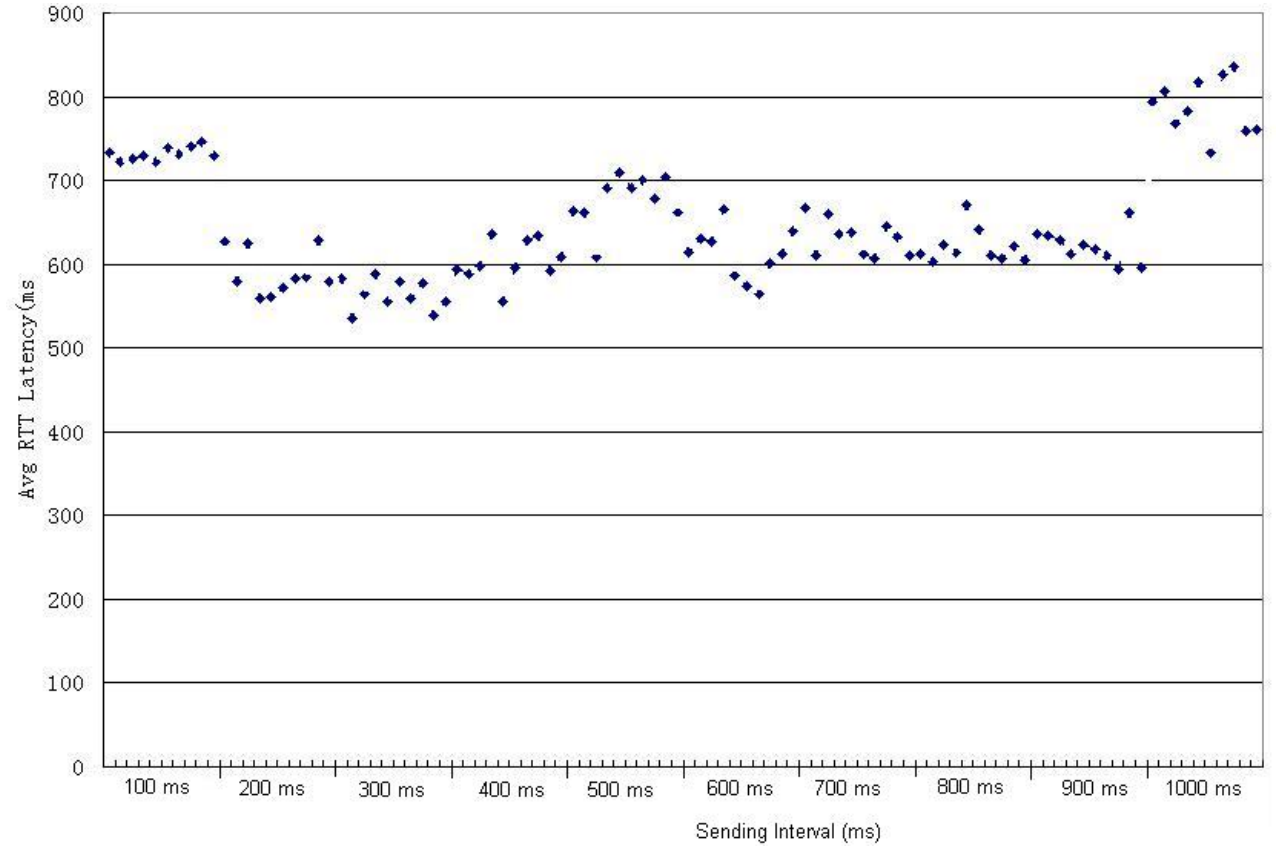


Figure 6-11: RTT Latency as the sending interval changes from 100 to 1000ms

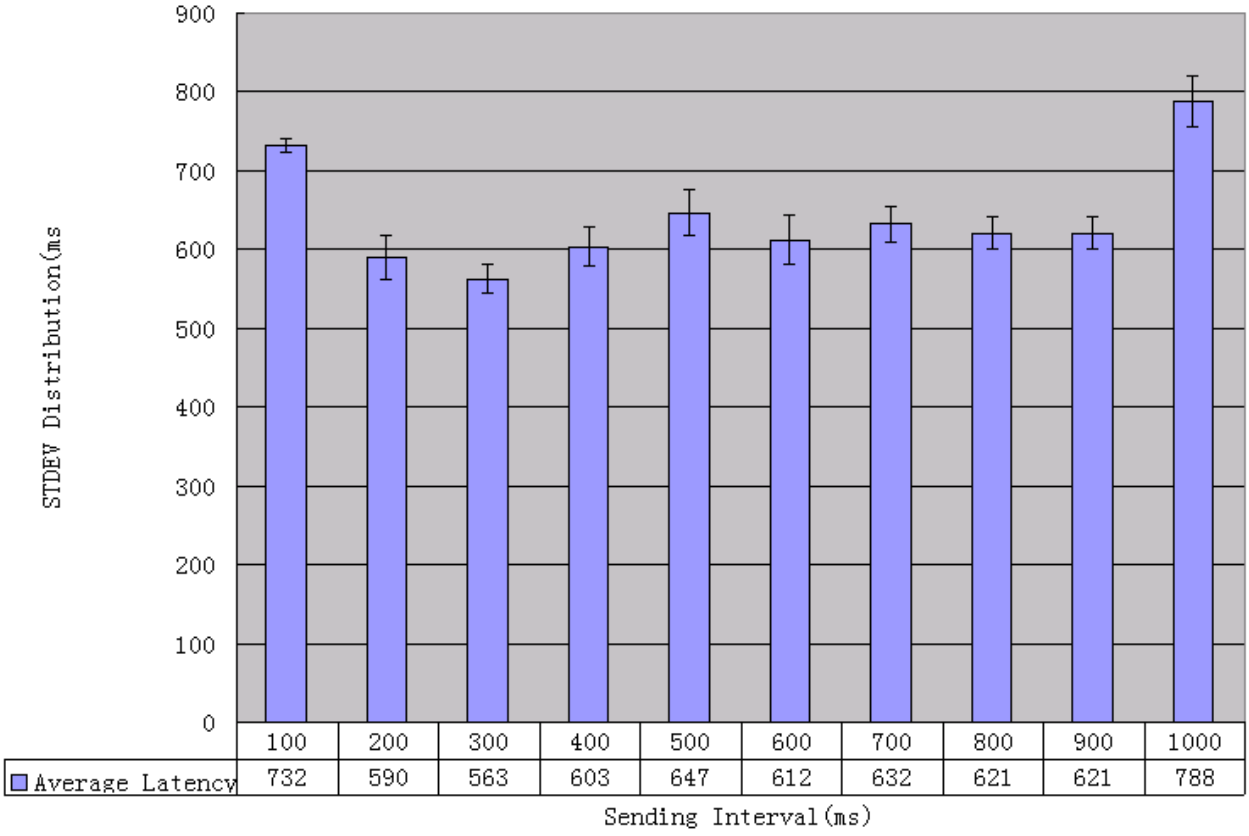


Figure 6-12: STDEV Distribution when the sending interval changes

Figure 6-11 clearly shows how the Average RTT Latency varies as the sending interval was increased from 100 ms to 1000ms. At different locations and different times, the values may be different since the network conditions are different. When the Sending Interval is 100 milliseconds, the average RTT Latency is much higher than the latency at a sending interval of 200ms. The reason for this phenomenon is that when the mobile sends packets to the base station, the base station needs time to allocate resources and buffers for the received data. Since the sending interval is too short, the base station can't process the packets in time, resulting in a higher average RTT latency. When the "Sending Interval" varies from 200 milliseconds to 500 milliseconds, the Average RTT Latency is the lowest.. From 500 milliseconds to 900 milliseconds, the Average RTT Latency is a little bigger than before but still remains stable. When the sending interval is 1000 milliseconds, the Average RTT Latency increases a lot at once. The increase of latency comes from the reason that mobile uses different radio channel to communicate with the base station. According to the mechanism of WCDMA, when the traffic rate is low, mobile station will use the Forward Access Channel (FACH) and Random Access Channel (RACH) instead of Dedicated Transport Channel (DCH). FACH and RACH are common channels. FACH is in the downlink and RACH is in the uplink. While all the users in a cell share the common channels, DCH is only reserved for a single user. Different channels have different bit rates. Table 6-4 shows the bit rates of these channels.

From this table, we can see that DCH has a greater bit rate than common channels. DCH is used for large amounts traffic, and FACH/ RACH is used for smaller amount of traffic. When the "sending interval" is 1000ms, radio station thinks the traffic is small and uses

FACH/ RACH to transmit packets. So the average RTT latency increases immediately.

Transport Channel Type							
Dedicated Channel				Common Channel			
DCH				RACH		FACH	
Uplink		Downlink		Bit Rate (kb/s)	TTI (ms)	Bit Rate (kb/s)	TTI (ms)
Bit Rate (kb/s)	TTI (ms)	Bit Rate (kb/s)	TTI (ms)	32	10	32	10
64	20	64	20				
64		128					
64		384	10				
384	10	2000					

TTI: Transmission Time Interval

Table 6-4: Channel bit rates [19]

6.3.3 Performance analysis when moving at high-speed

Game players might play when they are in the subway or bus. So it's necessary to test the performance when moving at high-speed. I did these tests in a subway in which the train ran at the average speed of 70km/hour[1]. The average packet size was 100 bytes and the sending interval was 400ms. Figure 6-13 shows the comparison between the static and moving situations in one game period about (1.5 minutes). This figure shows the variance while static and moving. From this figure, I found the average latency of the moving and static situations are almost same. The two average latency arc distributions are nearly the same variance. However the high-speed moving situation had the highest latency peak in my tests. There are many causes for such a latency peak such as handover among base stations and variance of signal strength. I did another group of experiments to test the latency distribution in the situations of static and high-speed moving. Figure 6-14 is the latency distribution in the static situation. The average latency of this test is 587ms. Figure 6-15 is the latency distribution in the high-speed moving situation. The average latency in this situation is about 588ms. The two figures have roughly the same latency distribution. These again attest to the fact that the average latency in the high-speed moving and static situation are eventually the same, even though the latency depends on the network conditions. In different locations, the latency may vary a lot. My tests show that wireless multiplayer gaming is a feasible service even in the high-speed moving situation.

[1] This information comes from the SL website < www.sl.se> and directly consulting with the train drivers.

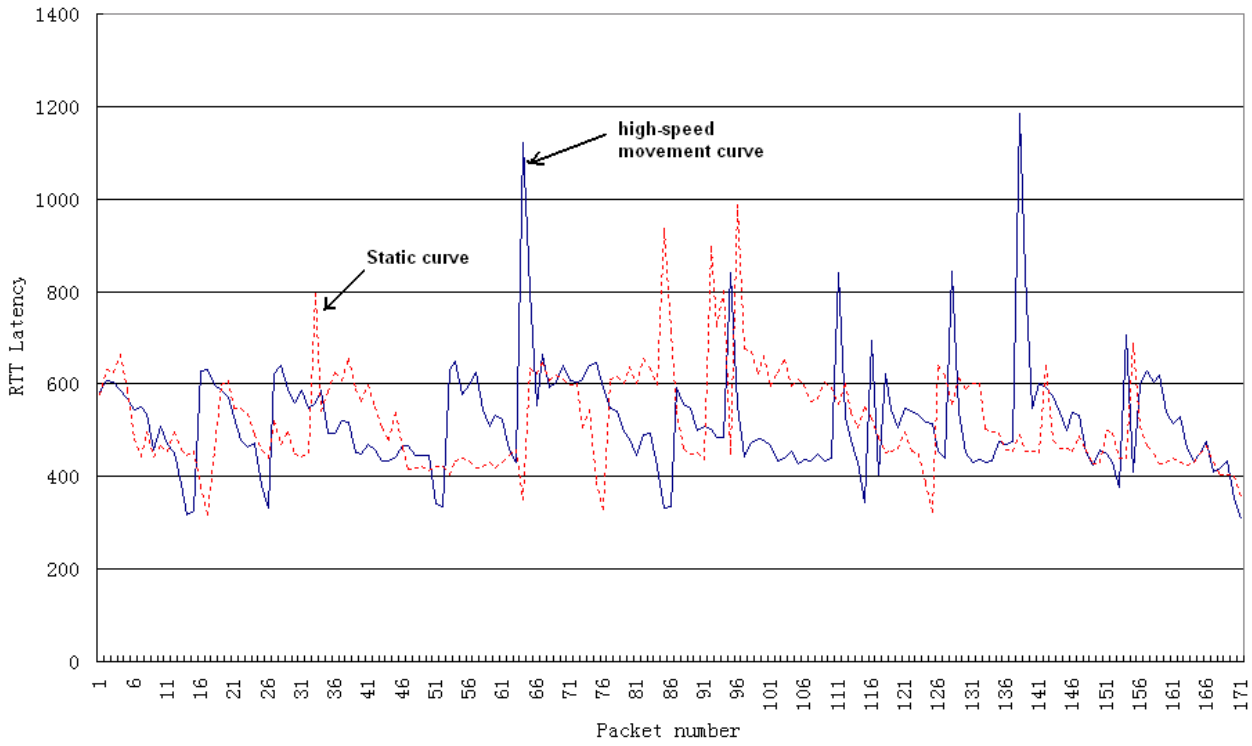


Figure 6-13: Variance comparison in the static and moving situations

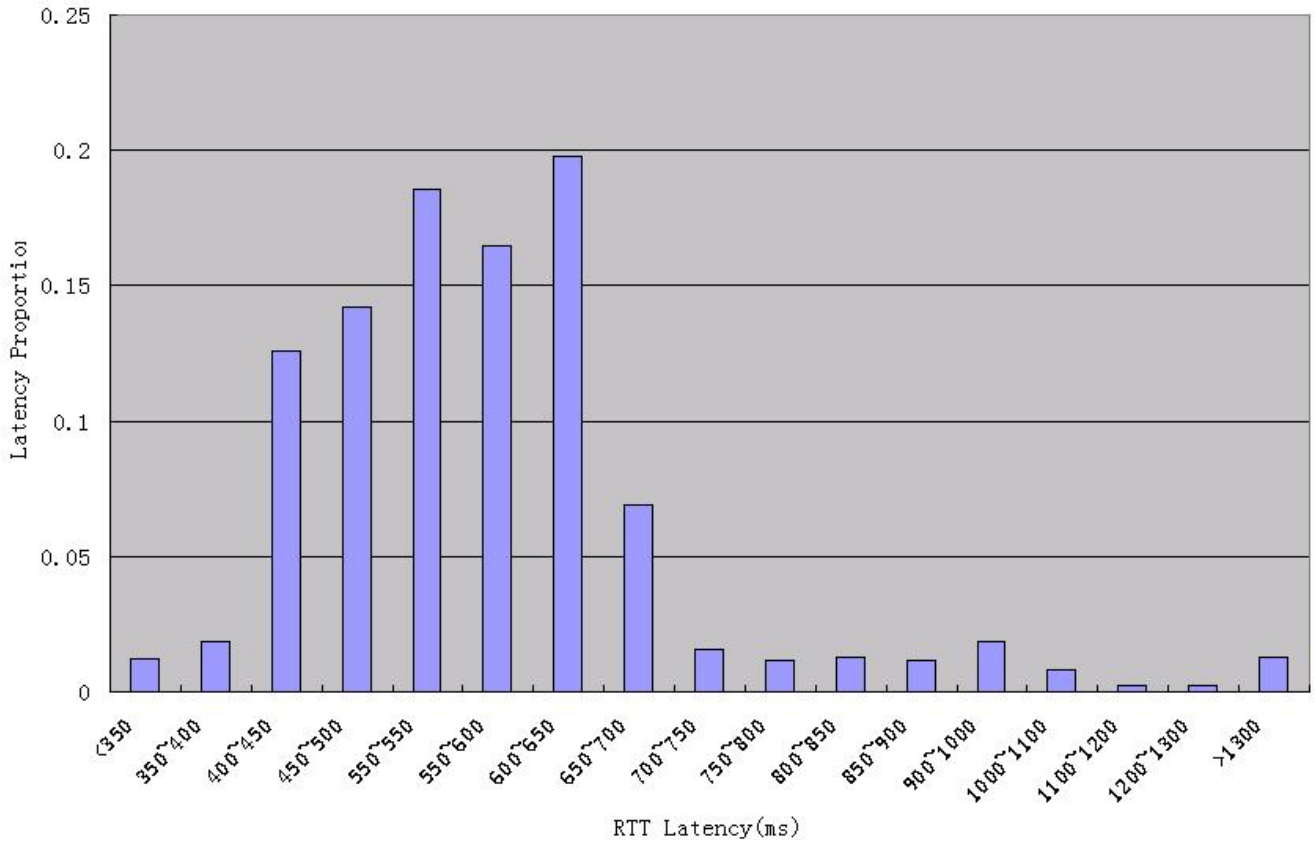


Figure 6-14: Latency distribution in the Static Situation

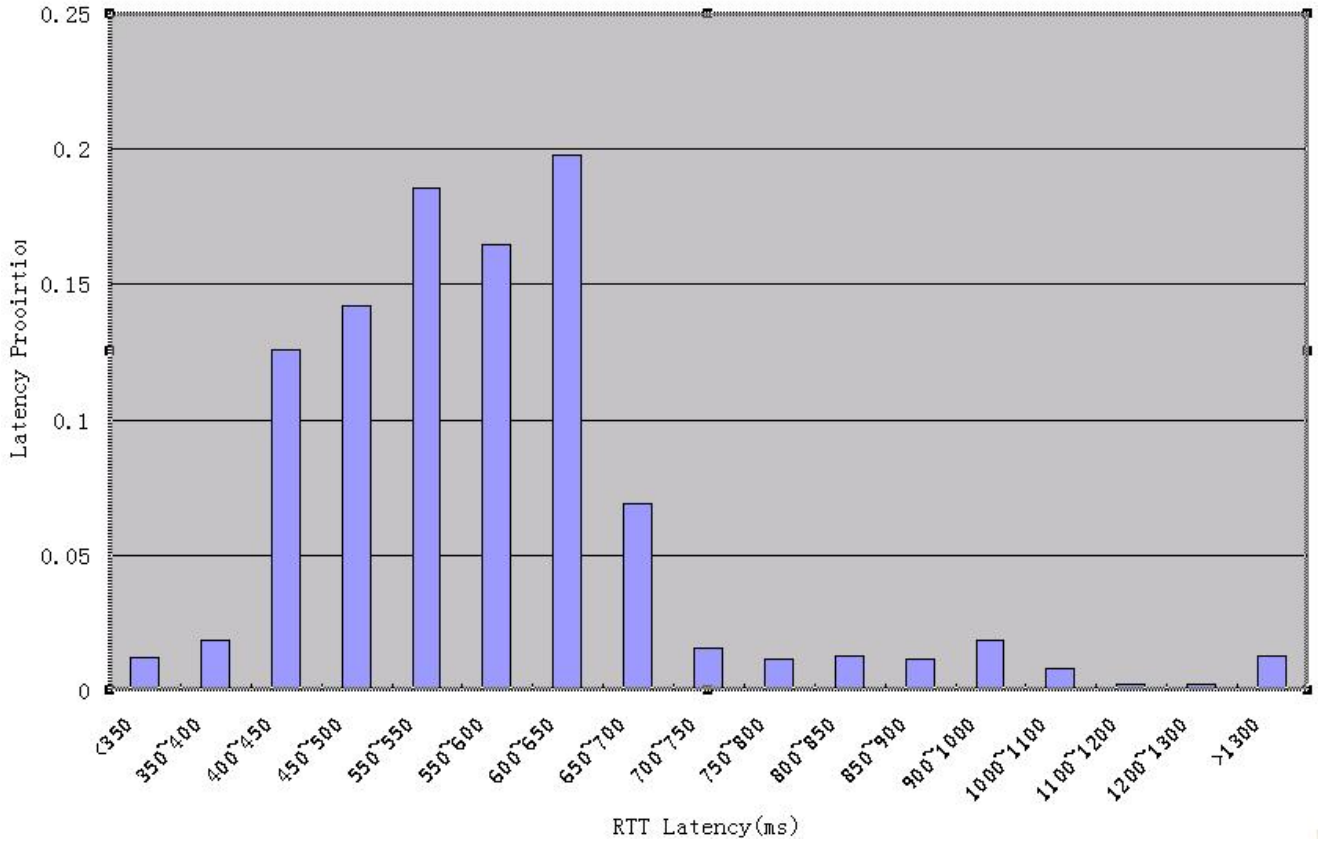


Figure 6-15: Latency distribution in the High-speed Movement Situation

6.3. 4 Performance analysis of TCP and HTTP Connections

TCP and HTTP are the two main connection protocols of most wireless multiplayer games. These two protocols have different characteristics and it's necessary to compare these two protocols to evaluate their use for game connections.

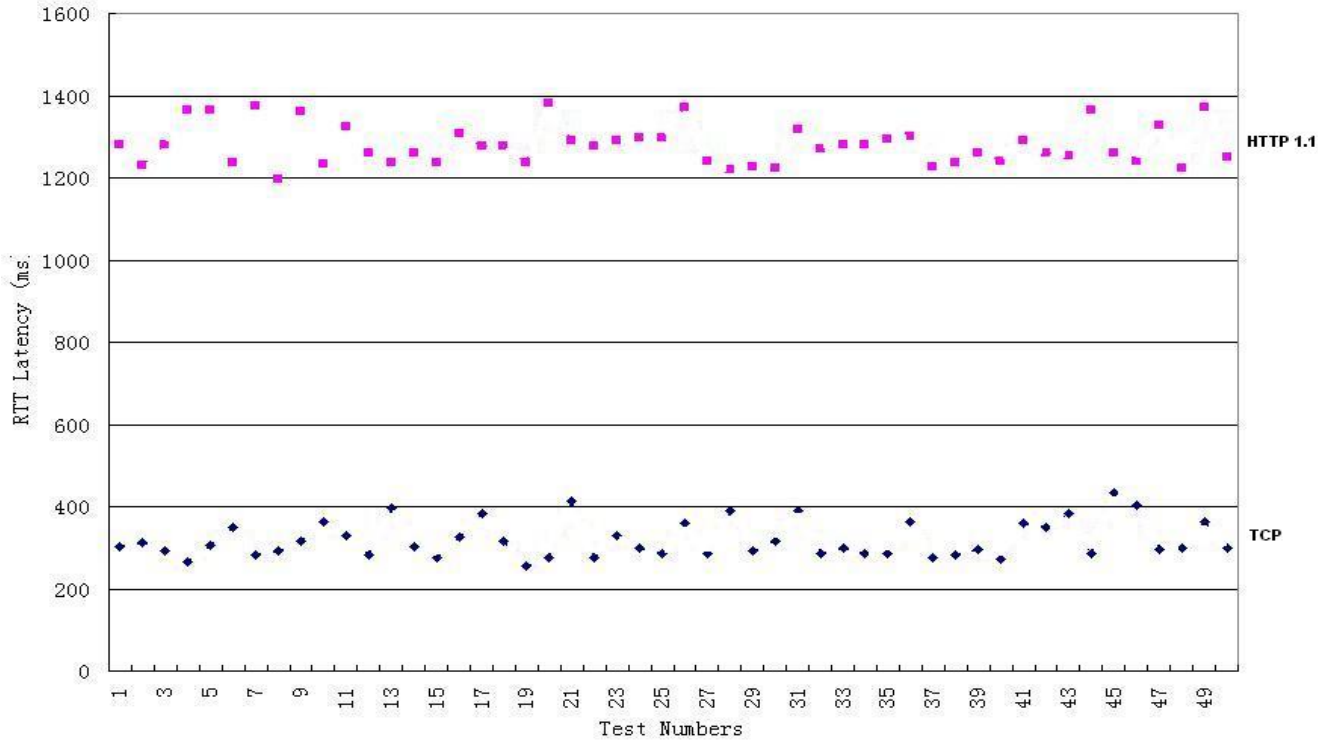


Figure 6-16: Comparison of RTT Latency between TCP and HTTP

Figure 6-16 compares RTT Latency of TCP and HTTP. In my tests, the average RTT latency of a TCP connection is 320 ms, while the average RTT latency of HTTP is 1280 ms. Thus, there is a huge performance difference between a TCP connection and an HTTP connection. After observing the communication with Terraplay system, I found the latency difference of these two protocols mainly comes from their different communication process. HTTP needs a couple of HTTP1.1 request and response, and this causes HTTP1.1 to have longer latency than a TCP connection. HTTP is a protocol which works on session layer of OSI. The request and respond messages is used to negotiate the parameters for conversation. Figure 6-17 shows the HTTP communication process, and Figure 6-18 shows the TCP communication process.

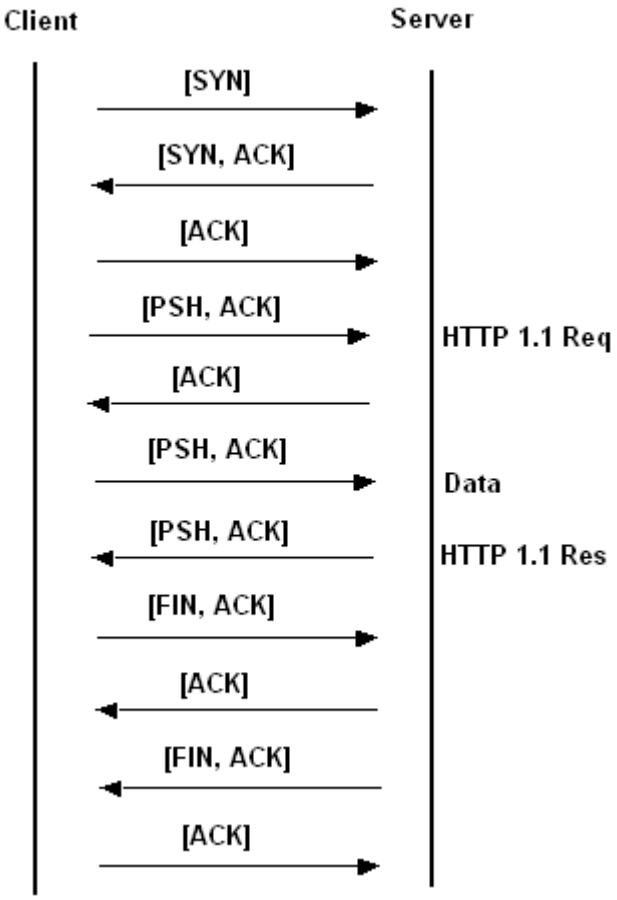


Figure: 6-17 HTTP Communication Process

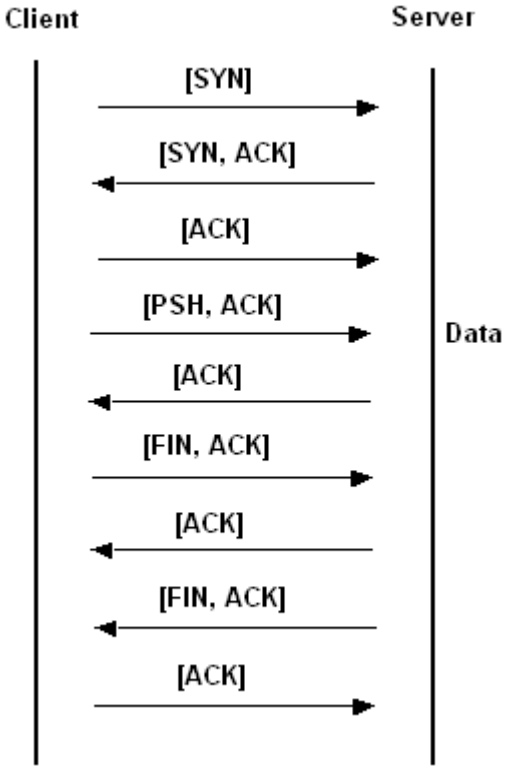


Figure: 6-18 TCP Communication Process

From the tests, it's clearly to see the HTTP is not a good choice for most wireless multiplayer games on Terraplay System. The latency of HTTP is too great to support wonderful gaming performance.

Chapter 7

Conclusion

In this thesis project, I tested the performance of wireless multiplayer games on Terraplay's MOVE platform from the viewpoint of game developers. According to the Nokia report[9], latency that is less than 700ms is accepted to all game types, and a latency of less than 900ms is sufficient for a strategy game. From my test results and analysis, it can be seen that commercial multiplayer game service is acceptable on Terraplay's system. The Turn based games can run with the latencies up to 1400ms. A TCP connection shows an average latency (for most multiplayer games) of less than 700ms. Even in the high-speed movement environment, the latency still remains acceptable if there is sufficient high quality signal coverage.

It has been shown that the bigger packet size (300 bytes) has no significiant effect on the performance of Terraplay enabled wireless multiplayer games for current WCDMA networks. This point is important for game developers. Since gwSendStreamObejectNotification is the main message containing game data, the larger the permitted packet sizes, the greater potential of a successful game design. Greater packet size is sometimes necessary for designing 3D games. My project clearly shows that game developers don't need to worry about packet size when they design wireless multiplayer games on Terraplay system.

In my project, I also showed that the packet transmission interval can affect the performance of games. For some kinds of games, such as racing games and FPS games, the reaction speed is very important. The game programmers hope to send and receive quicker responds by sending packets at a highe rate. But from my tests, I found too short an interval would cause higher latency, which seriously damages game performance. In my project, I think that the interval should be in range of 200ms~500ms (i.e. 2 to 5 packets/s). For some game genera, such as turn base games, their natural interval is often quite long. This rate often results in changing radio channel, connection tear down, or even disconnection. To prevent this, sending an empty heartbeat packet is necessary to maintain the connection with the Terraplay Move system. For such packets the gwSendStreamObejectNotification has a zero length data part. Resulting in a gwSendStreamObejectNotification message of 76 bytes. These attidional packets will increase the network traffic a little.

For most wireless multiplayer games, HTTP is not an appropriate choice for a game connection. From my tests, the average latency of HTTP is always greater than 1000 ms.

Such a high latency is only acceptable for some turn based games. However, HTTP is the connection type that all phone manufactures must support according to JTWI standard and TCP is only supported by some mobiles, this can directly affect the popularity of some games; but this is likely to cause users to select another phone.

Chapter 8

Future works

According to Anttila and Lakkaorpi[20], for most multiplayer games, even 30% packet loss during the handshaking phase can still be tolerated. In the game phase, the packet-loss effect on game performance depends on the game genera. By using a UDP connection, the average game latency and the traffic payloads can be decreased, this is very attractive for the latency sensitive games such racing and FPS games. But UDP will face the problem of packet loss. Current Terraplay doesn't support UDP on their platform, but they will support it in the near future. Although I can't test the game performance of UDP connection directly on the Terraplay MOVE now, many experiments have been done on internet. According to previous tests the packet loss rate varies from 0.14% to 40%.[21]. I believe in a wireless network, that the packet loss rate will be higher than that. Therefore, future measurements are needed to see if it is possible to decrease the game latency by using UDP despite the higher packet loss. Further, I would like to see how much packet loss is acceptable to each game genus.

In this project, I discussed the situations when the packet size is smaller than 300 bytes. My experiments show that there is no obvious latency increase when the packet size is increased to 300 bytes. But when I increase the packet size to 500 bytes, the average latency increase about 25%. If I continue to increase the packet size to arrive at 1000 bytes, then the network connection always broke during the testing period. Currently the packet size of mobile multiplayer games is smaller than 200 bytes. In the future, when the bigger packet size is needed, new research is necessary under the future technology environment.

Reference

[1] Terraplay System AB , “Terraplay White Paper”, 20 December, 2004
[2] Tina Xu , “118 Million 3G Subscribers in China by 2008”, 16 June, 2004, <http://www.3g.co.uk/PR/June2004/7904.htm>
[3] Frost & Sullivan , “Europ Mobile Gaming Revenues $7 Billion By 2006”, 16 October, 2003, < http://www.3g.co.uk/PR/Oct2003/5965.htm >
[4] Terraplay System AB , “Terraplay in 2 minutes”, 4 December, 2000, <http://www.terraplay.com/index.asp >.
[5] Sun Microsystems , “Java Technology for the Wireless Industry (JTWI); JSR 185 Overview”, January 2003 < http://java.sun.com/products/jtwi/overview.html >
[6] Nokia Corporation , “Multiplayer Game Performance over Cellular Networks”, 20 January, 2004
[7] Terraplay Sysmtem AB , “Developer Freedom”, 30 May 2005
[8] WMLClub, “Wireless Profiled TCP”, 31 March ,2001 < http://www.wmlclub.com/docs/especwap2.0/WAP-225-TCP-20010331-a.pdf >
[9] Nokia Corporation,“ Introduction to mobile game development”,January 2003 Nokia Corporation, “ Overview of Multiplayer Mobile Game Design”, December 2003
[10] Timo Halonen, Javier Romero, and Juan Melero, John Wiley& Sons, LTD, “GSM, GPRS, and EDGE Performance Evolution Towards 3G/UMTS.” Second Edition, ISBN: 0-470-86694-2,October 2003
[11] SonyEricsson , SonyEriccson SDK 2.2.3, September 2005 < http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp>
[12] Nokia Corporation , Nokia J2ME Developer Suit 2.1,
[13] Terraplay System AB , “Terraplay MOVE 2.1 Game Developer’s Manual – Guidelines”, 24 Jun, 2004
[14] Terraplay System AB, “Terraplay MOVE 2.1 Game Developer’s Manual, Appendix1-Reference Manual”, 24 Jun, 2004
[15] Wikipedia Encyclopedia, “Racing game”, last modified on 8 October,2005 < http://en.wikipedia.org/wiki/Racing_game>
[16] Sun Developer Network, “JDBC Technology”, < http://java.sun.com/products/jdbc/>
[17] The Apache Software Foundation (ASF), Jakarta Apache Tomcat Project, <http://jakarta.apache.org/tomcat/>
[18] Sony Ericsson Mobile Communications, SonyEricsson1010, <http://www.t24.at/storys/textcomputer/sonyericsson1010.htm>
[19] A. Lo, G. Heijenk, C. Bruma, "Performance of TCP over UMTS Common and Dedicated Channels", Proceedings IST Mobile & Wireless Communications Summit 2003, Aveiro, Portugal, June 15-18, 2003, pp. 138 - 142.
[20] Johanna Anttila, Jani Lakkaorpi, “On the Effect of Reducted Quality of Service on Multiplayer Online Games,” IJIGS, Volume 2, 2 November, 2003
[21] Robin Walker, “Packet Loss”, 1 Setpember, 2002 , <http://homepage.ntlworld.com/robin.d.h.walker/cmtips/loss.html>

Appendix A: Terraplay Test Toolkit Develop Manual

A.1. INTRODUCTION

Terraplay Test ToolKit (TTK) is a result of this thesis project. It is a J2ME application which is used to estimate end to end network latency. This document describes the game model and package structure of TTK. It's assumed that the reader is familiar with the Terraplay MOVE development documentation.

Abbreviation	Full Name	Description
MOVE	Terraplay MOVE	The Terraplay wireless solution
MLBS	MOVE Lobby Server	A server managing game sessions
MGW	MOVE Gateway	A server managing data distribution
TWP	Terraplay Wireless Protocol	
TWLP	Terraplay Wireless Lobby Protocol	
TWNP	Terraplay Wireless Networking Protocol	
J2ME	Java2 Micro Edition	
WTK	Wireless ToolKit	Toolkit for developing J2ME app.

A.2. TEST TOOLKIT GAME MODEL

The TTK game model is an asynchronous model. It can emulate almost all genres of multiplayer games for both GPRS and 3G networks. Here, a **player** means a human playing a game. A **client** is the game application running on a wireless device. A **user** is the game developer using TTK to test a given network.

A.2.1. Model Overview

In a real multiplayer game, the whole game procedure involves: players creating/joining game session, discovering other players, and starting game-play which consists of one or more rounds, such as races or shooting matches. Players can decide upon the properties of the next round before starting it. New players can join the game between rounds. A round consists of a series of updates. During an update, all players make a move which should be transmitted to the other players. These moves could be picking up a new poker card, speeding up the car, throwing a bomb, or shooting targets. At the end of the game, players leave the sessions and quit the game.

In the TTK game model, the procedure is simplified. TTK emulates two clients (A and B) on **one** mobile. A creates a public session with a certain name, then B joins this session. Following this, the round is started; B keeps sending messages and A receives them. These steps occur automatically. Finally, the two clients disconnect from the server.

The model is asynchronous since in the round the updates are not performed in lock steps. B does not wait for a response, but keeps sending with the specified frequency.

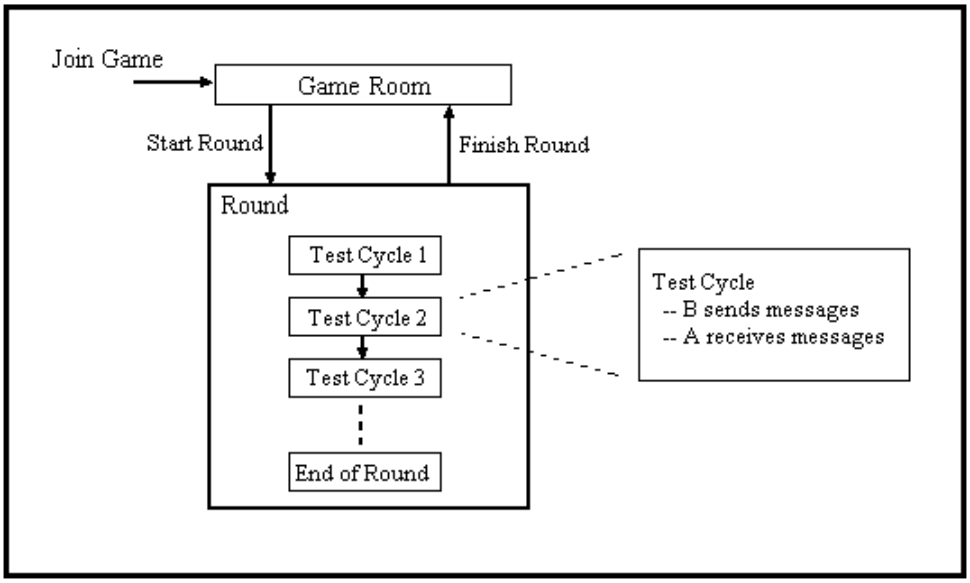


Figure A-1, Overview of the TTK Game Model

A.2.2 Implement

This section describes how the model is implemented.

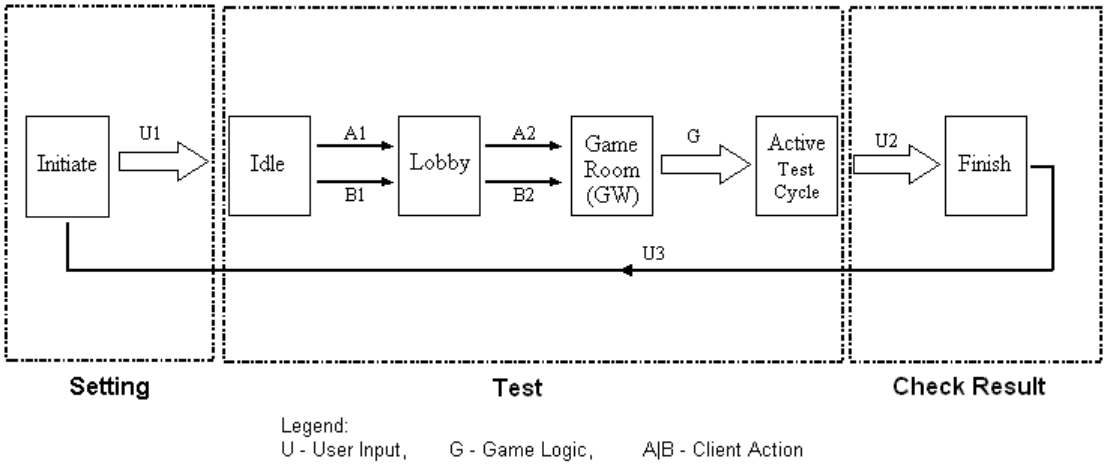


Figure A-2, Diagram shows the three (Setting, Test, and Check Result) client phases and six (Initiate, Idle, Lobby, Game Room, Active Test Cycle, Finish) tool states.

A.2.2.1 Phases and States

From the user point of view, the procedure can be divided into three phases: setting, test, and check result; the last is not necessary in a real game.

From the software point of view, the model can be divided into six states: waiting, idle, lobby, game room, Active, and finish.

A.2.2.2 Setting Phase

In this phase, users can decide upon the test parameters: data size and sending interval.

The specified size is used for creating messages. The specified sending Interval defines the waiting time between test cycles. U1 refers the user action of inputting test parameters and starting the test phase.

Initiate State
This tool state corresponds to the **Initiate** phase. This state consists of some automatically pre- and post-game activities, for example, setting player names, getting the lobby IP address and port number from information file (a JAD file), etc

A.2.2.3 Test Phase

In this phase, users can read the rolling messages from the message canvas which is shown on mobile screen. These messages show the information about the tool states and current test results. The only available user input here is to stop testing.

The test phase has four tool states: Idle, Lobby, Game Room, and Active. All transitions between these states occur inside game logic. Thus these states are totally transparent to users.

Idle State
At this state, the clients are not connected to either the Terraplay MOVE lobby server, (MLBS) or the Terraplay MOVE gateway (MGW). They initiate Internet service and try to setup connections with the MLBS and creating/joining a game session. Users will see a message indicating A and B is waiting for MLBS response.
Transition A1 indicates the creation a public session (create and send lbsJoinPublicSessionRequest), then handle responses.
Transition B1 indicates that B joins the public session (create and send lbsJoinPublicSessionRequest): handle responses.

Lobby State
When each client has joined the session, then the clients enter the lobby state. They will try to connect MGW and find each other. Users will see a message when the MLBS connects.
Transition A2 indicates the connection of MGW (GwConnectRequest); handle responses.
Transition B2 indicates that B connects MGW (GwConnectRequest); handle responses.

Game Room (GW) State
When the clients have connected to the MGW, they enter the game room state. Because the test scenario has only two clients, they do not need to wait for other clients to join. They will immediately try to start a round. Users will see a message indicating connections to the MGW.
G: the two clients start their stream tasks which create stream objects (GwCreateStreamObjectCommand), stream keys (GwAddStreamKeyCommand), and subscribing to the server (GwAddStreamKeySubscrCommand).

Active (Test Cycles) State

When all preparations are ready, the clients enter the active state. In this state, A sends heartbeat messages, GwHeartbeatRequest, to keep its connection alive. It listens for the messages sent from B. B keeps sending packets containing the message GwSendStreamObjectNotification. The size of this message is determined in the Setting Phase. Users will see messages that A received packets and the respective latencies. Users can input a command, U2, to finish the round and disconnect.
Transition U2: A and B send the GwDisconnectMeNotification message.

2.2.1 Check Result Phase

When the test has finished, the tool enters the check result phase. The statistic test results are displayed. Users can save the results and restart new tests, U3.

Finish State
In this state, clients aren't connecting to MLBS or MGW. The test results are saved in memory.

2. PACKAGE STRUCTURE

This section describes the package structure of the Test Toolkit.

2.1 Top Level Package Structure

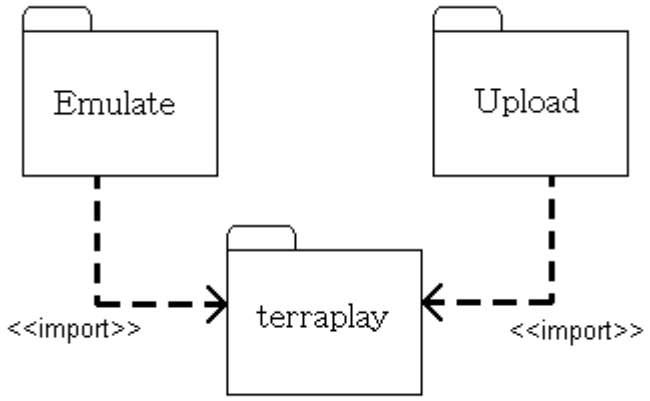


Figure A-3, Top level package structure

- Emulate – this package contains the Test Tool classes.
- Upload – this package contains the classes to Upload the Score.
- Terraplay – this package contains the Terraplay API used for the Test Toolkit

.

2.2 Emulate

- TestTool – a control class for switching forms and canvases, handling the start and stop of tests, and controlling the Record Management System (RMS) operations. This is the main class.
- Emu – a separate thread that emulates the entire game procedure. It also contains the method that sends latency data to the Servlet. The run method in the class is a game

loop emulating two players playing a game until the **stop** method is called. It performs the following operations:

- ✧ setupLobby: create a session
- ✧ setupGateway: connect to MGW
- ✧ streamTask: create stream objects and subscribe stream keys
- ✧ processEvent: send **GwHeartBeatRequest** and the messages of **GwSendStreamObjectNotification**
- ✧ receiveEvent: receive responses from MGW and calculate the latencies
- ✧ Sleep long enough to maintain frame rate and to reduce system load (rather than looping).

- MessageCanvas – where the messages are displayed
- TextMessage – a container class for messages to be displayed on the screen
- LogoCanvas – where the Terraplay logo and tool's name are displayed for three seconds.
- GenPayload – a control class for generating message payload based on the selected data size.
- Stat – a container class for gathering statistics (not implemented in the current version)
- GameTestEntity – a function class used to encode and decode data
- RMSUtil – a function class used to operate RMS.

Figure A.4 shows a UML class diagram of the Emulate package.

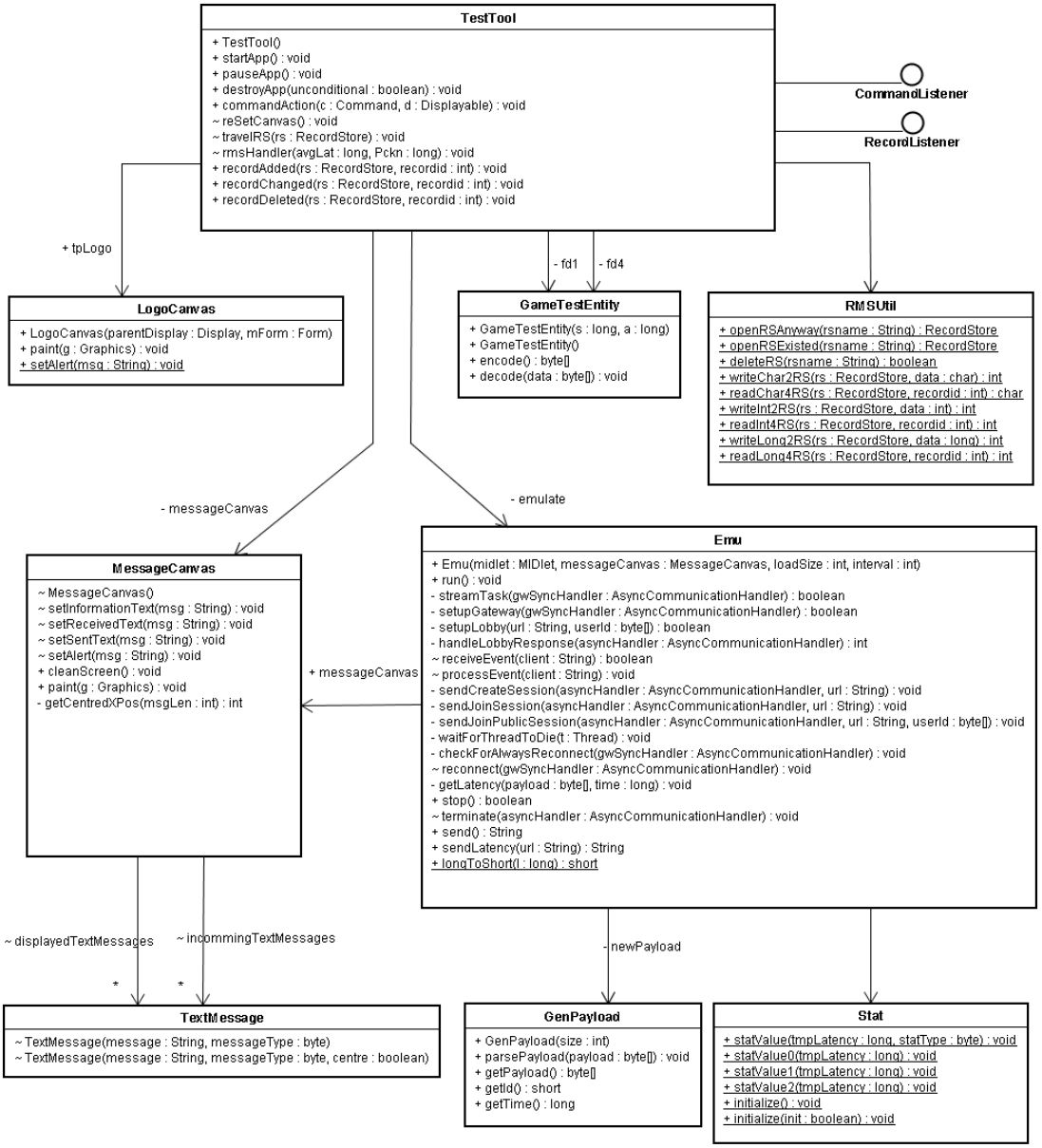


Figure A-4, UML class diagram of Emulate package

2.3 Upload

- UpLoadScore – a control class for switching forms and handling user actions. This is the main class.
- PutScore – a control class for setting up a connection with MLBS and uploading scores.

Figure 5 shows a UML class diagram of upload package

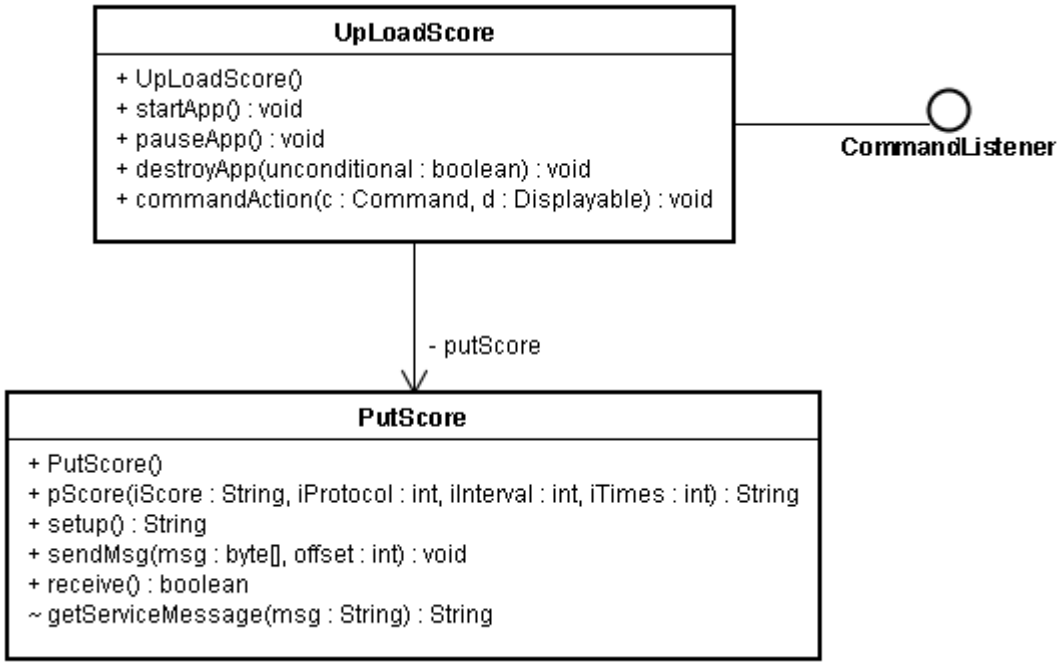


Figure A-5, UML class diagram of the upload package

2.4 Terraplay

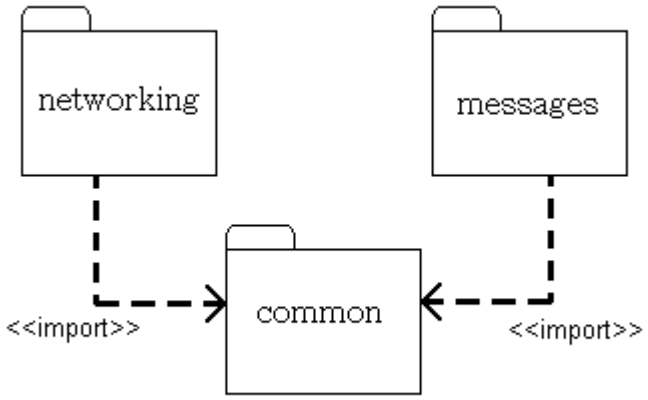


Figure A-6, the structure of Terraplay package

All classes included in the three packages come from the Terraplay's CHAT J2ME sample application. [A1] Only one file, AsyncCommunicationHandler, was modified to communicate only through TCP connections. For more information about these classes, see [13] and [14].

Terraplay.common

- Configuration
- ConfigurationDefault
- ConfigurationInterface
- ConfigurationFactory
- Constants
- HelpFunction
- InvalidMoveParameters

Terraplay.networking

- AsyncCommunicationHandler
- FailsToConnectToMgwException
- LostConnectionException
- SendReceiveHTTPDefault
- SendReceiveInterface
- SendReceiveTCPDefault
- SyncModelInterface

Terraplay.messages

- Message
- OutgoingTWPMessageInterface
- TWLPMessage
- TWNPMessage
- Mgwbasicobjecthandling
  - ✧ GwBasicObjectNotification
  - ✧ MgwBasicObjectHandling

- Mgwclientandclientgrouphandling
  - ClientInformationElement
  - GwClientConnectNotification
  - GwClientDisconnectNotification
  - GwClientGroupMembersConfirm
  - GwGetClientGroupMembersConfirm
  - GwGetClientNameConfirm
  - MgwClientGroupManagementMessages
- Mgwconnectionmessages
  - GwConnectConfirm
  - GwConnectRequest
  - GwDisconnectMeNotification
  - GwDisconnectNotification
  - GwHeartbeatConfirm
  - GwHeartbeatRequest
  - GwResumeConnectionCommand
  - MgwConnectionMessages
- Mgwmisc
  - GwCommandConfirm
  - GwMisc
  - GwReject
  - GwReportScoreNotification
  - GwSessionLockedNotification
- Mgwstreamobjecthandling
  - GwAddStreamKeyCommand
  - GwAddStreamKeySubscrCommand
  - GwCreateStreamObjectCommand
  - GwSendStreamObjectCommand
  - GwSendStreamObjectNotification
  - GwStreamObjectNotification
  - MgwStreamObjectHandling
- Mlbshighscoreadministration
  - LbsGetHighScoreConfirm
  - LbsScoreReportCommand
  - MlbsHighScoreMessage
- Mlbsmisc
  - LbsCommandConfirm
  - LbsRejcet
  - LbsServiceMessageNotification
  - LobbyMisc
- Mlbssessionmanagement
  - LbsCreateSessionRequest
  - LbsJoinPublicSessionRequest
  - LbsJoinSessionConfirm
  - LbsJoinSessionRequest
  - LobbySessionManagementMessages

Appendix B: Terraplay Test Toolkit User Guide

B.1.TERMINOLOGY

All figures shown were captured from the SonyEricsson z1010 simulator of the SonyEricsson Wireless Toolkit. There will be some differences from real mobile phones or other types of mobile phones.

B.2. INTRODUCTION

Terraplay Test Toolkit, TTK, is a set of J2ME applications designed for measuring the attributes of wireless communication networks. Those attributes are expected to be the most important factors that will impact the performance of mobile multiplayer games. The TTK consists of two tools, Test - Upload Score and Test Tool. It can be started with default built-in settings. However, it's recommended to read this document, specifically the section **B.5 *configurations and restrictions***.

Note: Running TTK needs Terraplay MOVE System or Terraplay MOVE SDK supported.

B.3 DOWNLOAD AND INSTALLATION

B.3.1 Prerequisite - Terraplay MOVE System

Before starting TTK, please make sure that a Terraplay MOVE System or Terraplay MOVE SDK is running. You could install the system yourself or get support from your local operator, content provider, or Terraplay Systems AB***.***

B.3.2 Terraplay Test ToolKit

TTK can be downloaded from the website of Terraplay System AB:
http://www.terraplay.com Or contacting: iw03_xch@it.kth.se

To support the Server functions of TTK, a Servlet needs to be implemented on a web server. The Servlet file developed in the thesis project can also be downloaded from Terraplay's website. The implementation of the Servlet is described in section **B.5 *configurations and restrictions.***

B.3.3 COMPONENTS

The two components of TTK are Upload Score and Test Tool.

Figure B-1, Launch screen, selecting the component to launch

B.3.3.1 Upload Score

Upload Score can test the network latency in the simplest way, calculating the time from sending scores to getting a confirmation. It can be used for getting a rapid, direct, and simple result without complicated settings.

B.3.3.2 Test Tool

Test Tool provides complex test scenarios, more detailed statistic result, and more additional functions. It can emulate different genres of games by setting corresponding parameters. The recommendatory settings are presented in the part: ***Recommendatory Settings***.

B.4 PROCEDURES

In this section, we will examine how the TTK works, step-by-step.

B. 4.1 Upload Score

B.4.1.1 Start application

On the “Launch Screen”, select the “Test – Upload Score” to start the Upload Score application. You will see the main form: “*Test (Upload Score)*”.

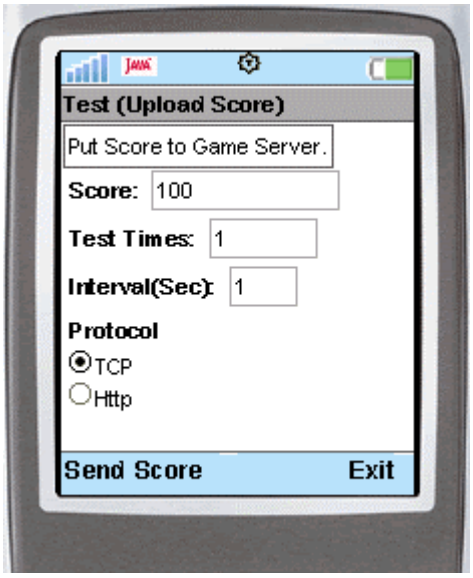


Figure B-2, Main Form for specifying the test parameters

B.4.1.2 Input parameters

In the main form, you can input and choose four test parameters: Score, test times, sending interval, and protocol.

There are some restrictions on these parameters, see section **B.5 *configurations and restrictions***

B.4.1.3 Start test

After decided upon the parameters, press **Send Score** to start the test. If you do not see the command button after modifying the parameters, please press the **More** button and select **Send Score**. Here you can also press **Exit** button to quit the application.

B.4.1.4 Wait

Figure B-3, Waiting Form

While the tool sends scores to the game server, the Waiting form is displayed. In this form, you can press **Retry** button to go back to the previous form (main form) or press **Exit** button to quit the application immediately.

If the application receives any response, the Result Form will be displayed automatically.

B.4.1.5 Get result

If errors occur or scores are sent successfully, the Result Form will be displayed. For more information about the errors, please see the section of **B.7 Error Message and Troubleshooting.**

The success messages will be shown like as in figure B-4 below. Of course the messages will be different if the test parameters were different.

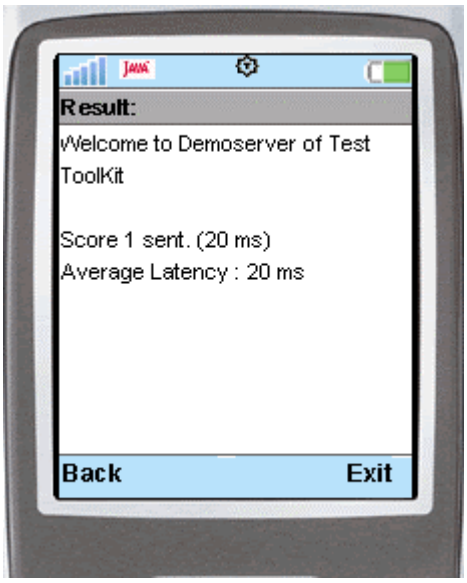


Figure B-4, Result Form

The messages of “Score X sent” indicate that X scores were successfully sent to the game server. The number inside the parenthesis is the measured latency for this update. The last sentence of result form shows the average latency of all tests.

On the result form, you can press **Back** button to go back to the main form for restarting the test or press **Exit** button to finish the test.

B. 4.1.6 Check result on the server side

After finishing tests, you can check the results on the server side. MOVE System stores the highscores in a database. MOVE SDK stores the highscore in a text file. Because the SDK’s highscore file can record the ten highest scores at most, it’s recommended to increase the test score in each test cycle. The path to the highscore file is:
\Terraplay MOVE SDK home folder\highscores\TTK-highscores.txt
For example: “C:\Program Files\Terraplay\Terraplay MOVE SDK 2.1\highscores\T…”

B.4.2 Test Toolkit

B.4.2.1 Start application

Select the Test Tool to start the test. Then you will see the Terraplay logo and the name of this tool. This screen is displayed for last three seconds. The next screen is the Welcome Form.

Figure B-5, Terraplay Logo

Figure B-6, Welcome Message

B.4.2.2 Welcome message

Press **CONTINUE** button to display the parameter form, or press **Exit** to quit the tool. You can also press **READ RECORD** button to read the previous test results.

B.4.2.3 Input parameters

Figure B-7, Settings Form

In this form, you can input two test parameters: sending interval and data size. The sending interval is the time (millisecond) that the tool waits before sending the next data packet. The data size is the number of bytes of data that will be sent to other players, such as the current position on the game map.

After inputting parameters, you can press **Confirm** button to go to the confirm form or press **Exit** to quit the application.

B.4.2.4 Confirm

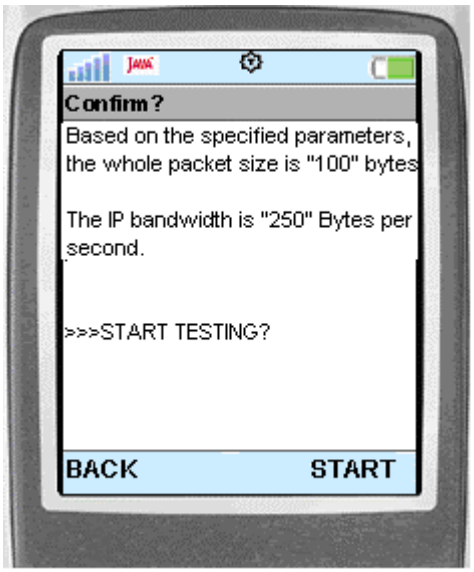


Figure B-8, Confirm Form

In this form, the whole packet size and the using uplink bandwidth are displayed. The packet size is calculated by adding to the data size, TCP header, Terraplay message header, and another six bytes of Heartbeat. The structure has shown on Figure 6-3. The uplink bandwidth is mathematical. The value in the real environment is different because the real sending interval will not be a stable value.

You can press **Back** to reset the parameters, or press **Start** to start testing.

B.4.2.5 Testing

While testing, the message canvas is used for display all important information and messages. These messages tell you what happens during testing and the final results. The test will last until you press the **Stop** button. These buttons will automatically be reflected to specified number keys on particular mobile phones according to the setup of phones.

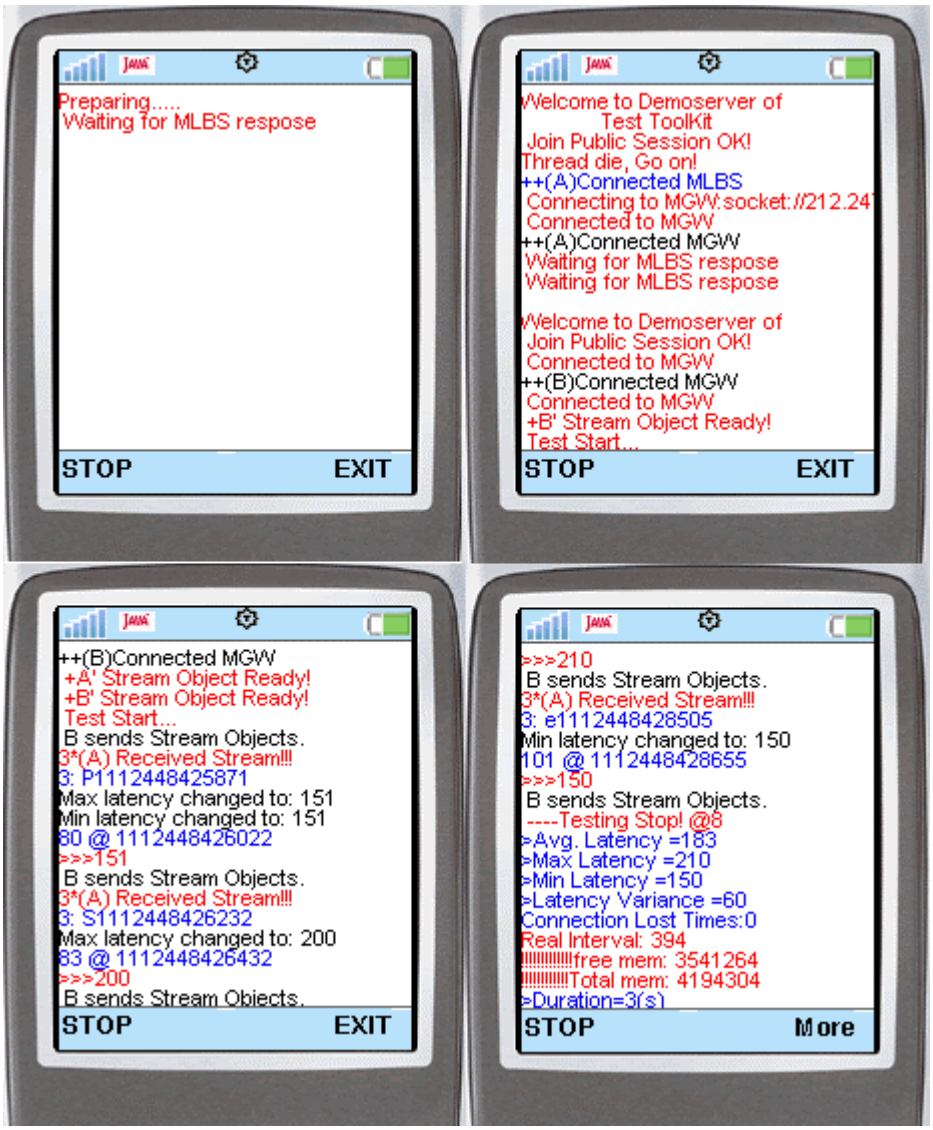


Figure B-9, message canvas

After the test stopped, you can press **More** button to choose more options.

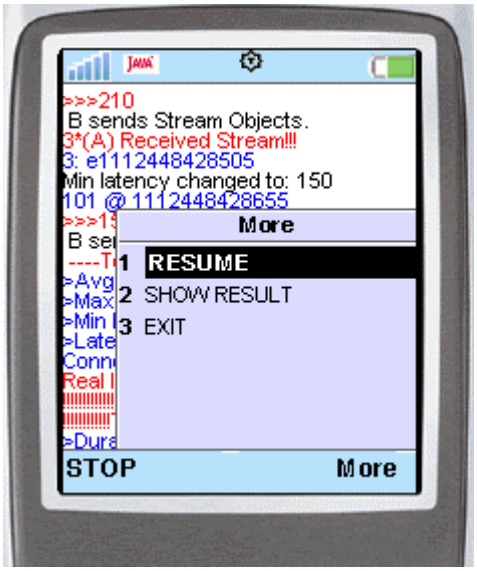
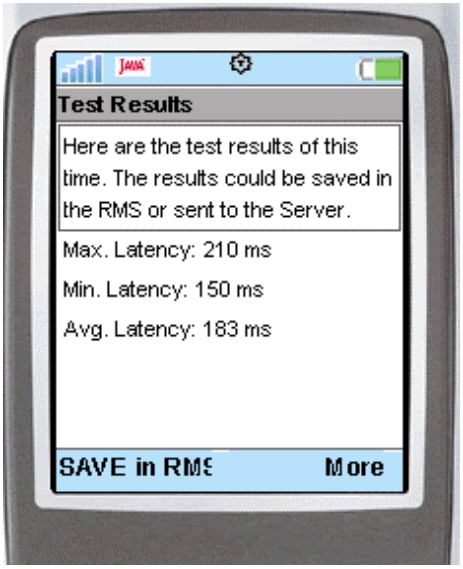


Figure B-10, More Options

You can press **RESUME** button to continue testing, press **SHOW RESULT** button to display the result form, or press **EXIT** button to quit the tool.

4.1.1 Test Results



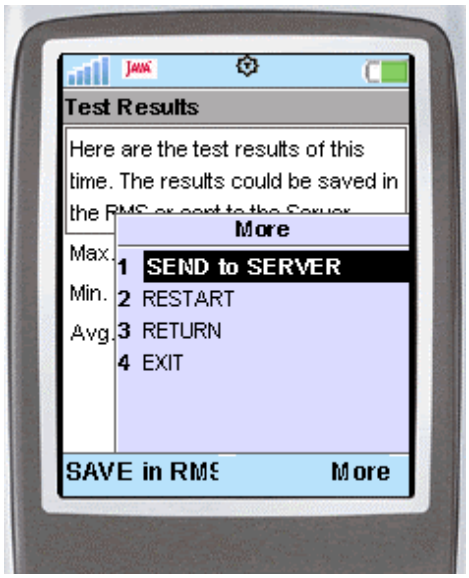
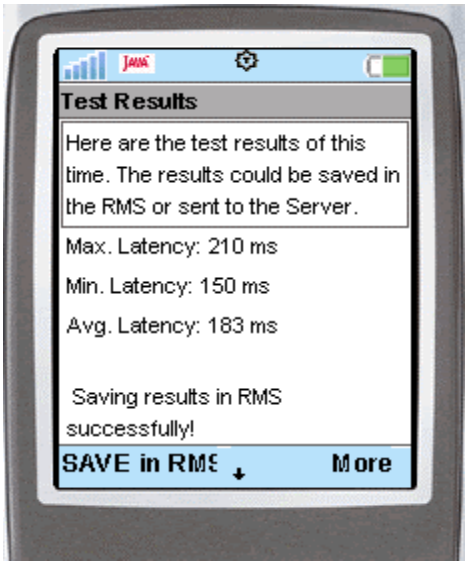


Figure B-11, Test Results

The test result concludes maximum latency, minimum latency, and average latency. You have two options to deal with these data: **SAVE In RMS or SEND To SERVER**. You can get corresponding messages after executing these two commands.



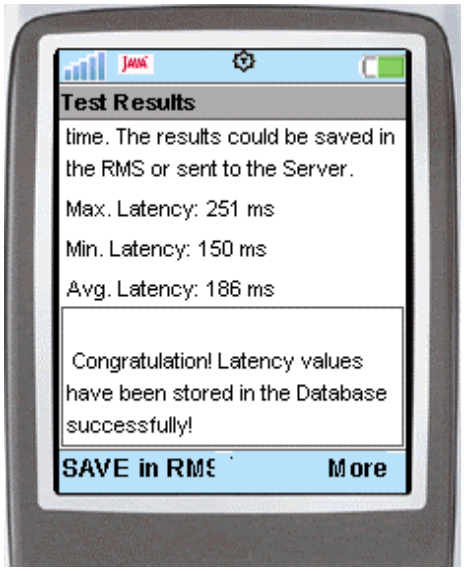


Figure B-12, Save

Figure B-13, Send

You can press **RESTART** button to go back to the main form and restart from the beginning. Pressing the **RETURN** button, the previous message canvas will be shown. Press **EXIT** button to finish test.

B. 4.2.7 Check database on the server side

All the latencies got in tests could be sent to the server and stored in a database. Please check System Configuration section to get the information about how to check the test result on the server side.

B. 5. System Configurations and Restrictions

B.5.1 Restrictions of Upload Score

Sending Interval

Each MOVE enabled application needs to be registered in the MOVE system. The configuration file is “mAppData.cfg”. It defines many parameters impacting game performance. **PublicSessionTimeout** defines the lifetime of a public session. In TTK, the timeout is 90 seconds. Hence, the Sending Interval cannot be greater than this value.

B.5.2 Restrictions of Test Tool

Sending Interval

The maximum value is 90000 milliseconds, because of the session lifetime.

Data Field Size

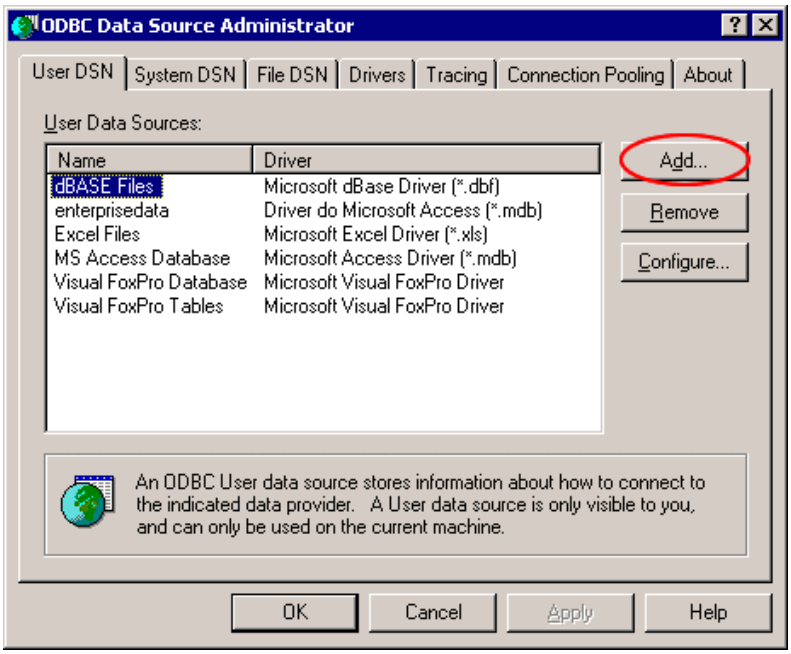
This value is the maximum payload size of Terraplay gwStreamObjectNotification message, which is used for carrying data.

B.5.3 Configuration of Servlet

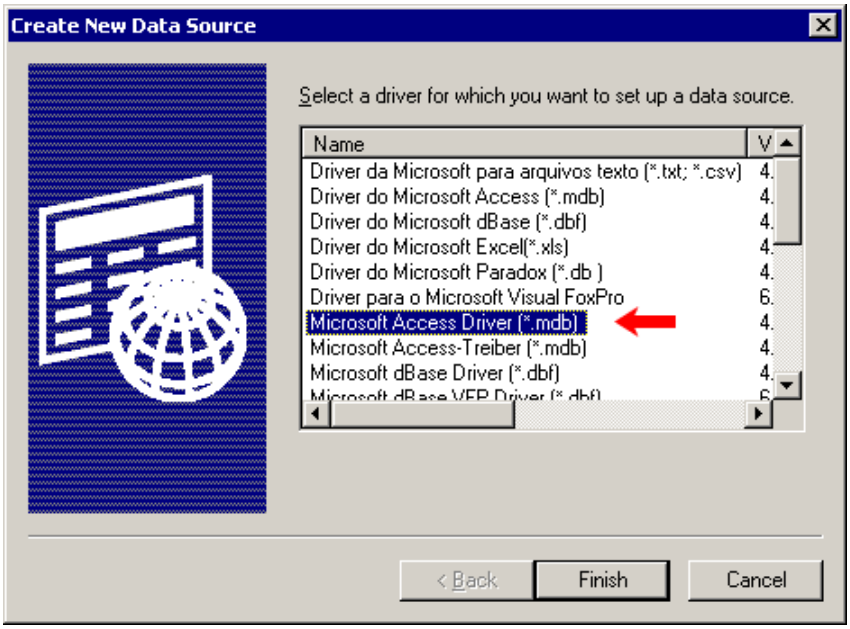
Sending all latencies to a server and storing them in a database is an additional function of TTK. You can check all the values in the database file and do further operations, such as importing the data into professional statistical software.

The name of the database file is “data.mdb”. Do the following steps to configure the data source. First of all, Go to "Control Panel >> Administrative Tools >> Data Sources (ODBC)" on the Web server.

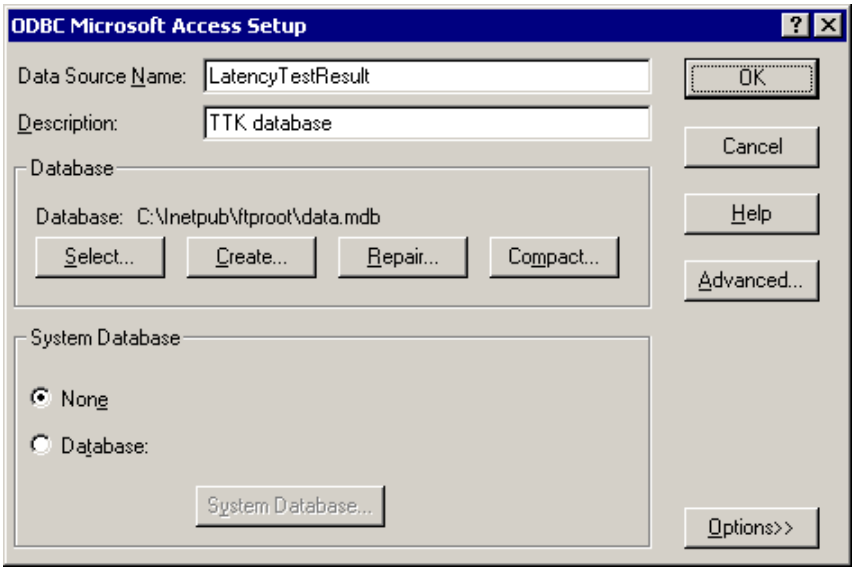
1. Add a new "***User Data Source***"



2. Select ***Microsoft Access Driver (*.mdb)*** and press Finish



3. Input “LatencyTestResult” in the field of ***Data Source Name***; write some information text in the ***Description*** field; select the correct path of database file, **data.mdb.**



4. Press ok to finish configuration.

This data.mdb file and the Servlet class can be downloaded from the website [1]. Please make sure that this file is saved in the web server.

[1]：www.terraplay.com

B.6. Recommended Settings

In this section, the traffic models of different game genres are presented. You can use these models to test or create new models based on your demand. All traffic models are extracted from real games, with some modifications.

B.6.1 Real-time Games

Racing Game/ Sport Game
Packet Interval: 500 milliseconds
Average Packets per Second: 2
Size of data field (Packet Size): 24 (100) bytes

Action Game
Packet Interval: 300 milliseconds
Average Packets per Second: 3.3
Size of data field (Packet Size): 16 (92) bytes

Strategy Game
Packet Interval: 1500 milliseconds
Average Packet per Second: 0.7
Size of data field (Packet Size): 14 (90) bytes

First Person Shooting Game
Packet Interval: milliseconds 400 milliseconds
Average Packet per Second: 2.5
Size of data field (Packet Size): bytes 24 (78) bytes

B.6.2 Turn-based Game

The turn-based games are different from the real-time games. Game application sends one type of messages to keep connection/session alive in a stable frequency, and sends the data messages, such the position of a new chessman, when a new game action happens.

Card Games /Board Games
Packet Interval (keep connection): 5000 milliseconds
Average Packet per Second: 0.1
Data Field Size (Packet Size): 9 (85) bytes

B.7 Error Messages and Troubleshooting

B.7.1 Upload Score

1. *Can NOT setup connection with lobby server!*
   Possible reasons:
   - The mobile Internet service doesn't set correctly.
   - Lobby server isn't running, or its IP address or port number has changed.

   If you are using a simulator, such WTK, the following reasons are also possible.
   - Network interface doesn't work well: cable unconnected, no IP address or private IP address.
   - You computer cannot connect to public network.

2. *Cannot create CreateSessionRequest message or JoinSessionRequest message*
   Possible reason:
   - The mobile doesn't have enough memory.
   - The application file is damaged. Download and install again.

3. *Connection Lost while sending session request message! Please Try Again!*
   Connection Lost while sending score! Please Try Again!
   Fails to reconnect TCP
   Possible reasons:
   - Happening network problem while sending messages

4. *Sending message: Open URL failed or send message failed!*
   Sending Score: Open URL failed or send message failed!
   Possible reasons:
   - Lobby's IP address or port changed, or Lobby server crashed by accident
   - Too much traffic online

5. *Fail to receive Notification or any Confirm from Lobby.> X*
   X is the number of Lobby Reject Reason. For more detailed information, please check [REF3].

6. *Cannot receive session confirm message*
   Sending Score failed, Please Try Again.
   Possible reason:
   - Do not receive any confirm of successfully sending. The Lobby is too busy to reply.

B.7.2 Test Tool

Test tool has similar functional module of communicating with Lobby. Please check the previous section. In this section, you can get the information about the errors occurring while communicating with MOVE gateway (MGW).

1. *Fail to connect MGW, Reason is X*
   Possible reason: please check [REF3]

2. *Failed to create stream object: Reason: X*
   Stream Tasks Fail
   Possible reason: please check [REF3], as the same section as the reject reason of MGW

3. *Operation X failed! Reason is Y.*
   Possible reason: please check [REF3] for operation ID and reject reason.

4. *ProcessEvent: Sending packet failed*
   Possible reason:
   - Invalid Move Parameters, please check the restriction of parameters.
   - Connection lost

5. *ReceiveEvent: Receive packet failed*
   Possible reason:
   - Connection lost
   - MGW is crashed by accident

6. *HTTP response code: X*
   Possible reason:
   - The Servlet isn’t implemented correctly.
   - The webserver is not working.

7. *Not an HTTP URL*
   Possible reason:
   - The URL of the Servlet is wrong.

IMIT/LCN 2005-22

www.kth.se