INTERACTIVE GAMING APPLICATION SERVICE

For the UCT IMS Network

Prepared By: Allan Mushabe

This thesis is submitted in partial fulfillment of the academic requirements for the degree of

Master of Science in Electrical Engineering

In the Faculty of Engineering and The Built Environment

University of Cape Town

24 November 2008

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

Signature: _____ Date: 24 November 2008

Name: Allan Mushabe

Student Number: MSHALL002

Acknowledgements

I would like to thank my Professor Anthony Chan for helping to supervise this project. I would also like to express my gratitude to the members of the Communications Research Group for all their advice and assistance throughout the duration of the dissertation. To my family for all their support and assistance with this dissertation I thank you all very much. Lastly I would like to thank the Lord God for guiding me through all that I do.

Abstract

The IP Multimedia Subsystem (IMS) is a 3GPP service delivery platform for the implementation of Next Generation Networks' services. These services include video conferencing, IPTV, VoIP, Push to Talk and Presence to mention a few. The University of Cape Town Communications Research Group (UCT CRG), along with various other research facilities, have been studying and developing such services. As part of the CRG IMS service research, an interactive gaming application service was developed.

The interactive gaming application service was developed and it makes use of the UCT IMS Client. A Game Server was also created. The gaming service made use of the IMS Session Initiation Protocol (SIP) as part of its signaling and functionality. Thus, demonstrating the ability of the IMS to deploy rapid service development,. By manipulating the SIP messages, an entire game was formulated. The game in question was a version of the card game Blackjack. The game was designed and implemented to run concurrently with the UCT IMS Client and IMS network. Upon future testing and development, it is hoped that it will become an integral entity of the UCT IMS Client.

Interactive gaming is a major feature of social sectors. Until recently, interactive games have generally been restricted to solitary access networks. In other words, games have to be played on similar systems. The IP Multimedia Subsystem is designed to help with the convergence of different access networks by creating an all IP based network. In so doing, making it possible to have an interactive gaming application that conforms to all forms of access networks. As a result, the only significant requirements would be on the resources available by the access networks or the user terminals performance. This dissertation examines the concept of a gaming application which is accessible by all kinds of users.

Table of Contents

List of Figures

List of Tables

Glossary

3G	Third generation
3GPP	Third Generation Partnership Project
3GPP2	Third Generation Partnership Project 2
AAA	Authentication, Authorization and Accounting
ADSL	Asynchronous Digital Subscriber Line
AH	Authentication Header
AKA	Authentication and Key Agreement
ALG	Application Level Gateway
API	Application Programming Interface
AS	Application Server
AUID	Application Usage ID
AVP	Attribute Value Pair; Audio Video Profile
B2BUA	Back to Back UA
BCF	Bearer Charging Function
BGCF	Breakout Gateway Control Function
BS	Bearer Service; Billing System
BSF	Bootstrapping Server Function
BTS	Base Transceiver Station
CA	Certificate Authority
CAMEL	Customized Applications for Mobile network EnhancedLogic
CDF	Charging Data Function
CDMA	Code Division Multiple Access
CDR	Charging Data Record; Call Data Record
CGF	Charging Gateway Function
CK	Ciphering (Cipher) Key

CN	Core Network
COPS	Common Open Policy Service
CPS	Conference Policy Server
CRG	Communications Research Group
CRF	Charging Rule Function
CS	Circuit Switched
CS	CN Circuit Switched Core Network
CSCF	Call Session Control Function
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DOS	Denial Of Service
DSL	Digital Subscriber Line
DTMF	Dual-Tone Multi-Frequency
EAP	Extensible Authentication Protocol
ECF	Event Charging Function
EDGE	Enhanced Data Rates for Global Evolution
ENUM	Telephone E.164 Number Mapping
ESP	Encapsulating (Encapsulated) Security Payload
ETSI	European Telecommunications Standards Institute
FHoSS	FOKUS Home Subscriber Server
FMC	Fixed to Mobile Convergence
GBR	Guaranteed Bit Rate
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HLR	Home Location Register

HSS	Home Subscriber Server
HTTP	Hypertext Transfer (or Transport) Protocol
I-CSCF	Interrogating-CSCF
IESG	Internet Engineering Steering Group
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IMS-GWF	IMS-Gateway Function
IMS-MGW	IP Multimedia Subsystem-Media Gateway Function
IMSI	International Mobile Subscriber Identifier
IP	Internet Protocol
IPsec	Internet Protocol security
IPTV	Internet Protocol television
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISC	IMS Service Control
ISDN	Integrated Services Digital Network
ISIM	IP Multimedia Services Identity Module
ISP	Internet Service Provider
ISUP	ISDN User Part
MGCF	Media Gateway Control Function
MGW	Media Gateway Function
MIME	Multipurpose Internet Mail Extension
MOBILE IP	Mobile Internet Protocol
NGN	Next Generation Networks
OMA	Open Mobile Alliance
OSA	Open Services Architecture
OSI	Open Systems Interconnection

OSIMS	Open Source IMS Core
P-CSCF	Proxy-CSCF
P2P	Peer to peer
PA	Presence Agent
PDF	Policy Decision Function
PDP	Packet Data Protocol; Policy Decision Point
PEF	Policy Enforcement Function
PEP	Policy Enforcement Point
PLMN	Public Land Mobile Network
POC	Push to talk over the Cellular service
POTS	Plain Old Telephone Service
PRACK	Provisional Response ACKnowledgement
PS	Packet Switched; Presence Server
PSI	Public Service Identity
PSK	Pre-shared Secret Key
PSTN	Public Switched Telephone Network
QOS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RES	Response
RFC	Requests For Comments
RLS	Resource List Server
RNC	Radio Network Controller
RTCP	RTP Control Protocol; Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
S-CSCF	Serving-CSCF
S/MIME	Secure MIME
SCF	Session Charging Function

SDP	Session Description Protocol
SEG	Security Gateway
SIP	SIP Express Router
SGW	Signaling Gateway
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SIPS	Secure SIP
SL	Subscriber Locator
SLF	Subscription Locator Function
SMS	Short Messaging Service
SMTP	Simple Mail Transfer Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UCT	University of Cape Town
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Universal Resource Locator
VOIP	Voice over IP
WLAN	Wireless Local Area Network
XCAP	XML Configuration Access Protocol
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Subject

The Communications Research Group (CRG), at the University of Cape Town (UCT), recently developed an IP Multimedia Subsystem (IMS) client, with which IMS services could be created. The UCT IMS Client, as it is now known, works in conjunction with the Open Source IMS Core (OSIMS) created at FOKUS, in Germany. It was requested that research into future services be performed. These included video conferencing, IP television (IPTV), Presence Capabilities, Voice over IP (VOIP), resource management, security and various others. As part of these new services an interactive gaming platform was designed to help with the growing trend of Next Generation Networks (NGN). This dissertation is part of an ongoing development within the CRG towards creating a service delivery platform (SDP), IMS network at UCT.

This report discusses the various elements and fundamental ideas that went into the development of the interactive gaming application. It also provides avenues for future research and development for this gaming application and other services that could be implemented within the UCT IMS network.

1.2 Background

Online gaming is not a ground breaking innovation. It has been around for more than three decades now. Over the years, the type of game play moved from very simplistic to a more sophisticated arena. The games these days play host to millions of people worldwide. To some gamers, online gaming even befits part of a daily schedule.

Telecommunication has become more easily accessible and with the public's growing trend towards obtaining more and more social based services, the need for more advanced architectural schemes in telecommunications is evident. An example of this would be facebook. Facebook began primarily based on fixed line terminals then expanded toward mobile technologies. The IMS is one of those technologies that try to provide services or a means for such ubiquitous demands to become a reality.

The IMS platform is a relatively new one. It is under continuous development. Various modifications and applications are being studied and added to the platform to make it more universal. The ultimate goal of the IMS platform is to create what we know as an "ALL IP" based network. That is a network where by everything is routed through packet switched networks. This idea and others relating to it will be discussed in later sections of this dissertation [1].

1.3 Objectives

The main objective of this dissertation was to explore the region of online gaming services and thereafter develop a game application server for use by the UCT IMS Client. The application server and the game application in general were to become an integral part of the UCT IMS client. The UCT IMS client is very much still in its infancy and lacks a variety of services. But with the constant research being carried out it has potential to become a well renowned feature on the global scale.

The UCT IMS Client is part of an open software initiative. The client looks to provide others with a platform on which further research can be explored. The gaming application had to be designed to provide the fundamental aspects necessary for anyone to easily understand and expand on. Some of the areas of importance are that of Quality of Service and Security when running the application. The key areas involved with the design of the gaming application are firstly to ensure it was compatible with both the UCT IMS network and the devices that house the UCT IMS Client. Second to ensure that multiple users can operate the application simultaneously while utilizing the other features within the IMS architecture. That is: that the application service could run concurrently with other services developed for the UCT IMS Client.

1.4 Scope and Limitations

The interactive gaming domain is broad and features a number of various game play aspects. The main requirements set out in the scope were to:

- Provide a service addition for the UCT IMS Client.
- Provide a Server on which the service could be housed unless the service generated was of a peer to peer nature and not that of a client/server.

Current online games have substantial and ever increasing complexity. Since the UCT IMS Client is relatively new, less complex gaming was requested. The central idea of the dissertation was to generate a platform on which further work can be conducted. There is a restriction on the size of the data resources required to run the application. This limitation is because for the transmission of data to mobile devices, the data should be small in size. As the devices increase in technological power, so too will the applications which one can run on them.

1.5 Motivation

Over the last decade with the expansion of the Internet and networks alike, online gaming has spread throughout the globe. Currently, fixed line access networks take up 80% of the online gaming industry. Network service providers are hoping to be able to tap into such services in the near future as they see the industry expanding with the advancement of mobile network services.

According to Massively Multimedia Online Gaming, the number of active subscriptions to date is in excess of 17 million worldwide. Figure 1 illustrates the growth of online gaming over the last decade [2]. As the figure demonstrates, this is an exponentially increasing market and by pushing accessibility to all forms of user terminals the growth should be significant. With the IMS architecture it would appear possible to expand the market further. The only major disadvantage is that the games will be user terminal specific. This is because some devices, for example mobile phones, do not have nearly as much memory and processing power as that of conventional devices, such as Personal Computers. This restricts the complexity of seamless online games. However, given the progression of such devices over the last decade in terms of processing power and memory, one shall be able to play more advanced games on any device in the near future.

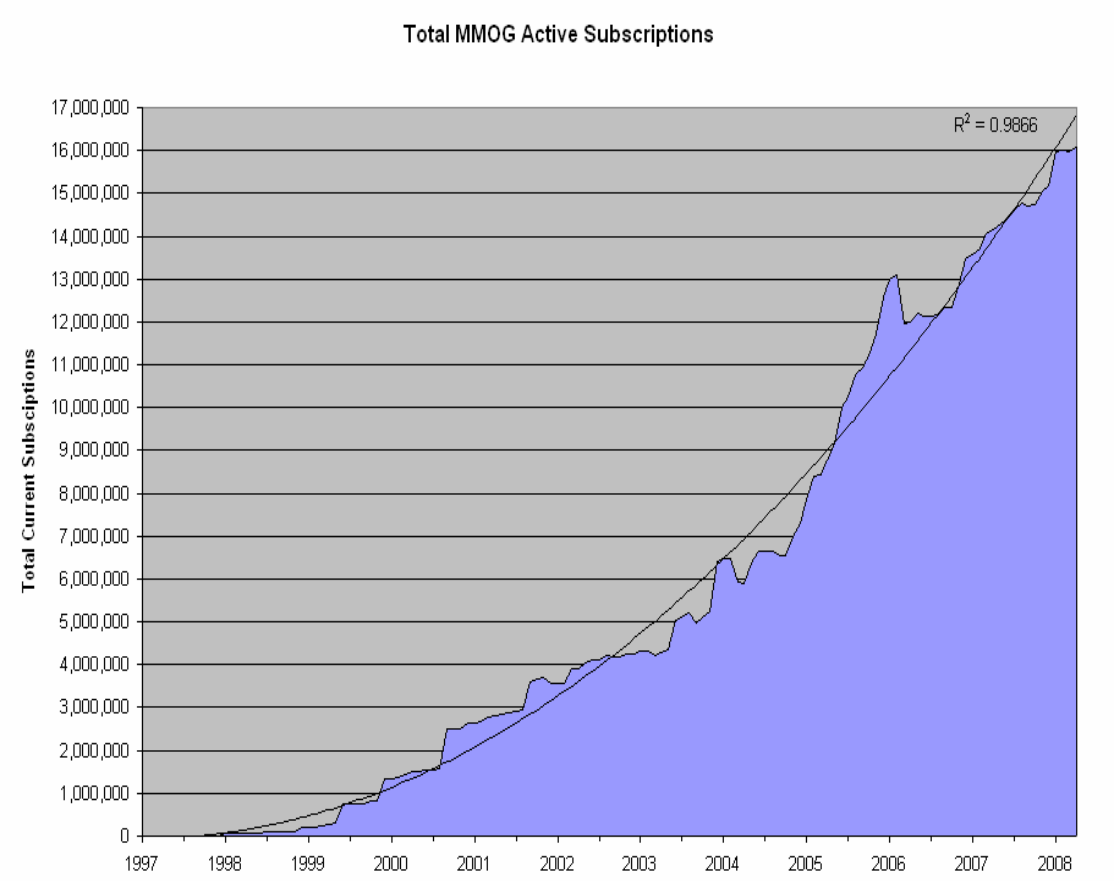

Figure 1: Active Online Gaming Subscriptions [2]

1.6 Structure

The structure of this thesis is as follows

1 The first chapter involves an introduction into the topic. It describes the various objectives outlined during the research of the thesis.

2 The second chapter discusses the literature reviewed for the understanding and conceptualizing of the procedures required to complete the set objectives. It also discusses similar work that is currently being conducted so as to make a brief comparison to the work done.

3 The third chapter looks into numerous design areas in which certain criterion could be applied in the interactive gaming application. It also provides some

further background information into the areas of the IMS platform that are employed in the gaming application.

4 The fourth chapter discusses how the design features researched were implemented in the final design of the gaming application service.

5 Discusses some of the results found with respect to the application testing.

6 The sixth chapter draws conclusions from the dissertation and ideas that were generated along the way to the completion of the dissertation. The chapter also delves into the numerous avenues of future work that can be provided for the gaming application and the entire UCT IMS network in general.

7 The last sections cite the numerous references used in completing this thesis.

Chapter 2

Literature Review

The following section investigates the literature that was studied with respect to compiling this study towards the generation of a design and implementation of an interactive gaming application in the UCT IMS network.

2.1 IP Multimedia Subsystem

2.1.1 Background

The IP Multimedia Subsystem or IMS is an architectural framework for delivering packet based data through an IP network to various communication technologies. It was developed by the 3GPP. It was initially conceived from the idea of evolving mobile networks from their previous GSM model to that of a more packet based mobile network. This would thereby expand the capabilities of mobile networks as it would be merger between the internet and the cellular world. After further research and development it was observed that the IMS architecture could be used as a possible means to create a universal “ALL IP” service delivery platform (SDP) [1, 3].

This SDP would then lead to the merging of fixed and wireless access technologies as demonstrated in figure 2. This merger would then help move away from the “stove pipe” scenario of communication to a more horizontal (figure 3) [4]. The stove pipe

reference was to the fact that each of the various protocols and services could not readily interact with the others. Each would need its own set of protocols with vertically layered stack as in the general protocol stack.

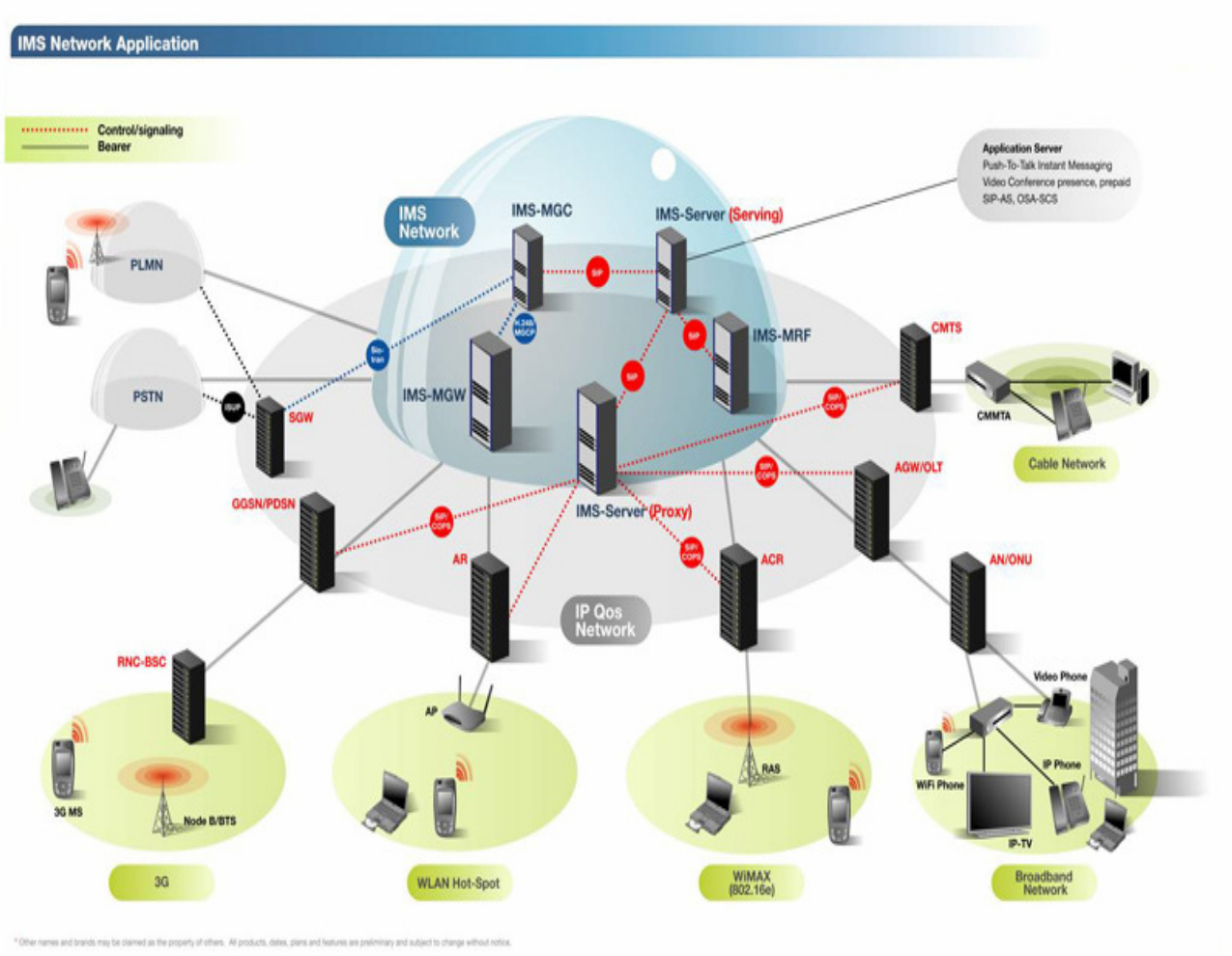

Figure 2: The IP Multimedia Subsystem Network [5]

The figure 2 above illustrates the manner in which the architecture hopes to provide universal convergence of all the communication technologies. One of the main limitations to the above scenario is the bottlenecking effect of having some access technology performing at a higher rate than that of others in the case of inter-networking. However with the innovations in the advancement of mobile devices in

terms of performance and around capability soon communication between various devices will be seamless.

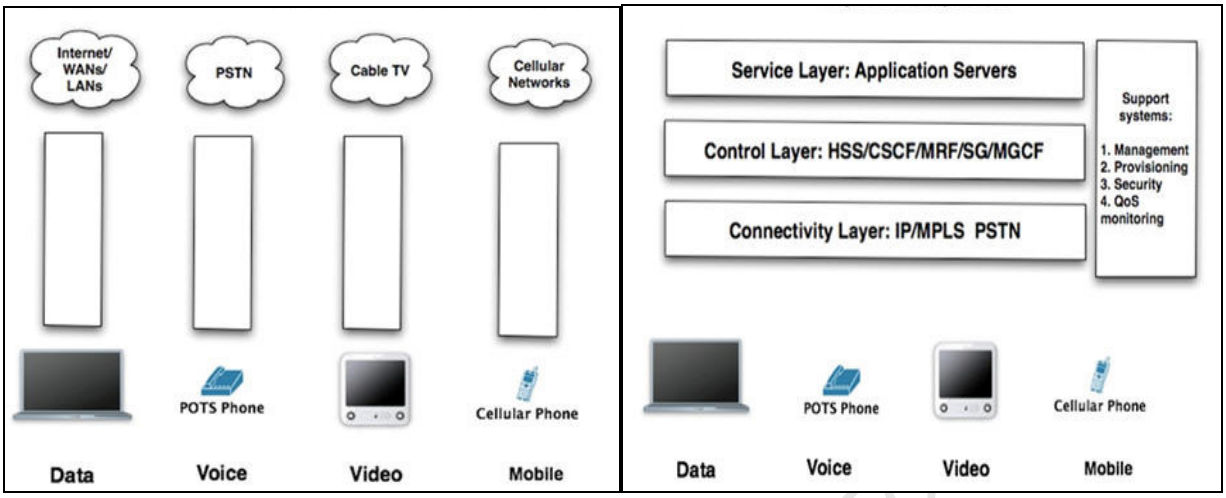

Figure 3: The legacy networks stovepipe and the IMS service orientated architecture [4]

Figure 3 shows the movement of "legacy" networks to more service orientated networks through the IMS. The IMS as shown in figure 3 also has support systems for session negotiations of user entities. These support systems include session management, resource provisioning, security and Quality of Service (QoS) monitoring. More on these aspects will be discussed in subsequent sections [3, 4].

2.1.2 Current Services and the IMS

Currently most service providers can provide the features demonstrated in the IMS platform so most ask why the need for the IMS? The main reasons have to do with the following; firstly for Quality of Service (QoS), charging of services and the previously mentioned integration of the different services [3, 6].

Some of the current services are discussed below:

- Packet-switched networks generally provide real-time multimedia services using a best-effort scheme and no real QoS. That means the networks offer no guarantee that packet will be delivered, that the user receives a certain amount of bandwidth and that the packet which are delivered arrive in a timely manner. This means that certain services such as VOIP for example end up having reduced performance or quality. Should a user try to make use of such a service when network resources are low the user may end up frustrated as they are not receiving the required optimal services. IMS makes an effort to coordinate a session establishment with a QoS provision. This allows a user to obtain a more uniform service as opposed to a best effort scenario [1, 3, 6, 7].

- In the case of charging current operators can not differentiate between services. This leads them to charge generally for the packets transferred. With the IMS, service providers are capable of providing packages based on the service required. An example would be that the service provider charge a fixed amount for each instant message sent and charge another amount for the duration of a video conference as opposed to the amount of data transferred. This would generally be more ideal to the users as the size of the data in a video conference can vary significantly with different encoding schemes, background lighting and other aspects. The IMS platform does not dictate the manner in which services should be charged but leaves it to the operators to decide [3].

- The integrated services scheme as mentioned previously try to merge all service enabling a user to make use of a single interface as opposed to numerous ones based on the service in use. Operators too would prefer to not have to develop service specific devices as this would prove to be more costly to develop. With integrated services, users can have greater freedom when moving locations yet still requiring services.

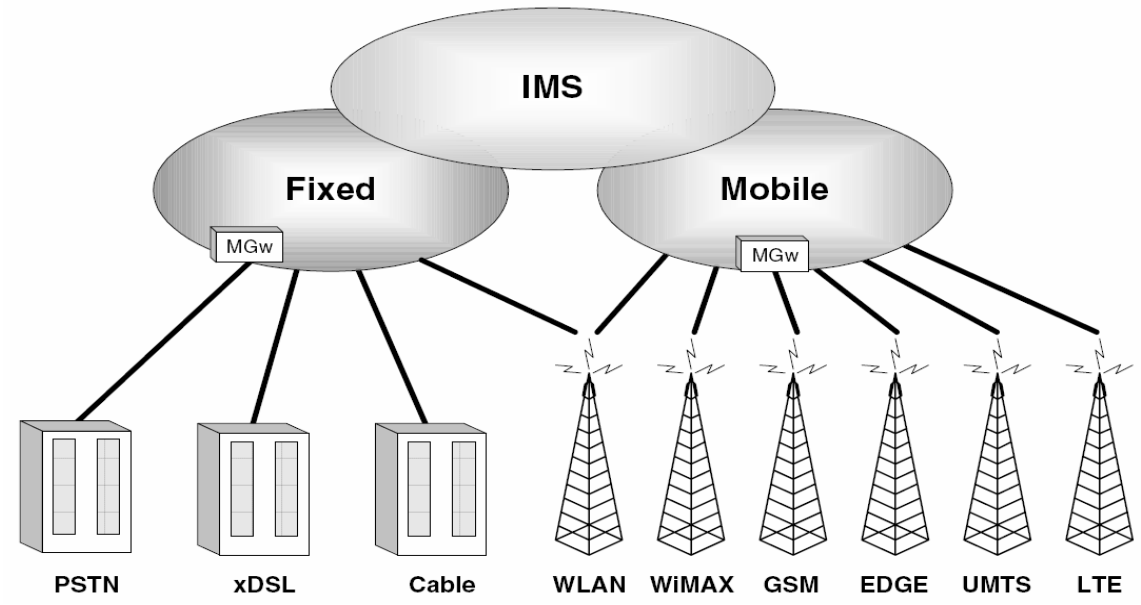

Figure 4: Depicts the merger of fixed and mobile networks [8]

The IMS on the other hand creates an approach that could bring about a more horizontal framework in the Open Systems Interconnection (OSI) model. It allowed majority of the access technologies to communicate with one another via the IMS plane [8].

2.1.3 IMS objectives

When the IMS was in its development and specification certain aspects or objectives were laid down. The following were the set objectives of the IMS platform [1, 3, 6].

- To support IP multimedia sessions:

  The support required to the IP multimedia sessions involves merging or evolving the current multimedia specifications which are predominately on circuit-switched networks to that of packet-switched networks [3, 6].

- Provide mechanisms for the negotiation of QoS:

  The IMS should allow for QoS negotiations; in this way they could help operators differentiate customers. For instance customers who pay for a high class of service might be entitled to more bandwidth during similar sessions [3, 6].

- Provide mechanisms for the interworking of packet-switched and circuit-switched networks:

  The support for interworking is an evident requirement as circuit-switched networks have been around for decades and cannot be readily replaced. Therefore the IMS needs to provide support mechanisms to allow the replacement to occur and at the same time merge various access networks in the packet-switched domain [3, 6].

- Support Roaming:

  As the IMS was initially conceived for mobile networks it was understood that roaming was already a standard feature of mobile networks and to keep in touch with this feature the IMS would also have to have the capability to adapt to user location changes [3, 6].

- Provide rapid service creation without requiring standardization:

  The IMS design aims to allow easy deployment of service applications. This allows for vendors to create and charge for their own services without the need to first standardize their application [3, 6].

2.1.4 IMS Layered Model

The IMS layered model as shown in figure 5. The model consists of the Service Layer (Application Plane), the Control Layer (Control Plane) and the Transport layer (User Plane).

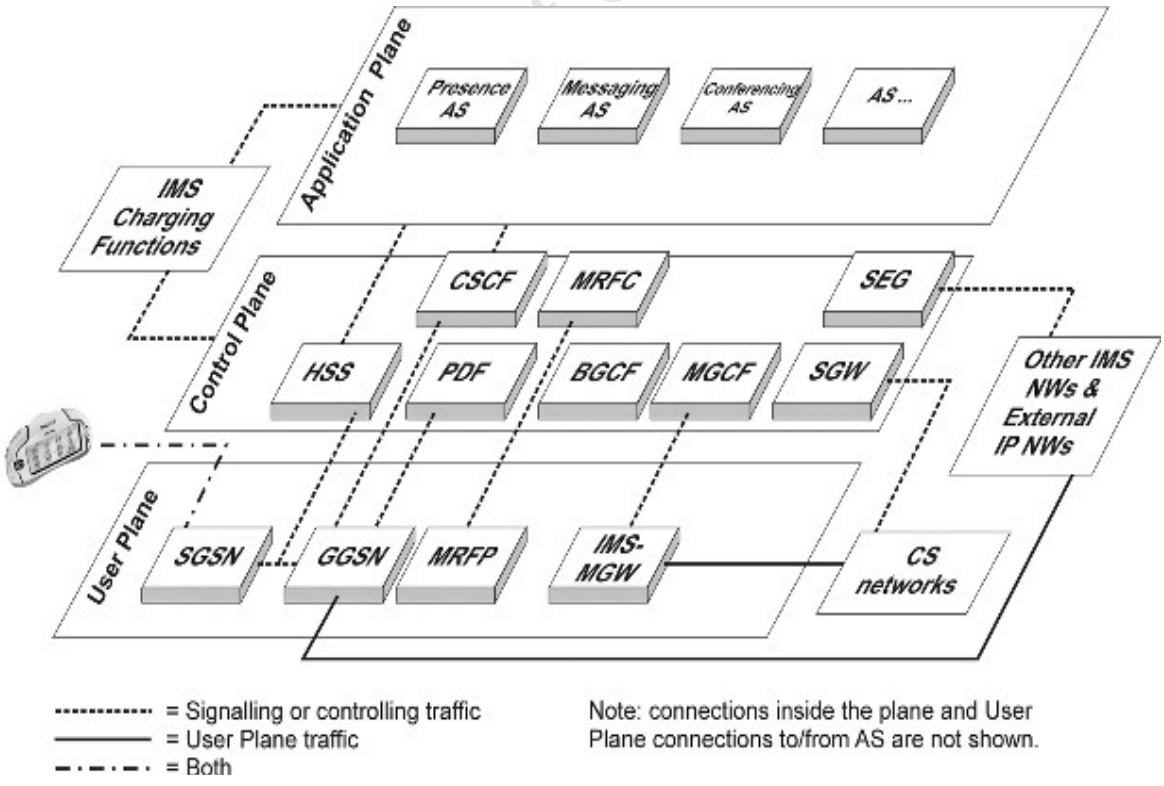

Figure 5: IMS Layered Design [3]

- The Service layer or as it is better known the application plane is where applications and content services, including application servers and Web servers reside. Some services can be strictly peer to peer based and might not necessary progress through the Service Layer. It includes generic service enablers that manage service elements such as user groups and presence. The service elements connect to clients via the control layer. The service layer hosts a majority of the multimedia applications [3, 6, 7, 9].

- The Control Layer/Plane is where the heart of the IMS functionality is located. It is in this layer where the support systems lie. The control layer comprises of policy control elements, authentication and authorization devices, elements that setup and tear down IMS sessions and interworking elements as well as databases [3, 6, 7, 9].

- The User Plane also known as the Transport Layer is where routers, switches, firewalls, and optical transport reside, along with gateways that translate protocols between packet- and circuit-based traffic. [3, 7, 9]

2.1.5 IMS Components

The IMS core is made up mainly of a number of Session Initiation Protocol (SIP) servers; these are more commonly known as Call Session Control Functions or (CSCFs) and will be discussed later sections. There are three main types of CSCFs. There are the Proxy-CSCF, Interrogating CSCF and the Serving CSCF. The IMS core also comprises of a database known as the Home Subscriber Server (HSS), Application Servers (AS), service and support elements [3, 7].

The Proxy-CSCF is the first contact the user has with the IMS core. It is where the initial call setup occurs. The Proxy-CSCF may be located in the user's home network or a visited network. The Serving-CSCF performs routing functions and admission

control within the IMS core. The Serving-CSCF is always located within the user's home network. Generally there is usually more than one Serving-CSCF within a network. When this occurs the Interrogating-CSCF comes into play. When a call is initiated the Interrogating-CSCF decides which Serving-CSCF the call should be routed to [3, 6]

The HSS is the main storage unit of all the IMS users' information and service-related data. It provides authentication, registration and service allocation information about a user when they want to setup IMS communication.

The Application Servers are primary entities in the IMS. They host and execute the various services. They can be SIP proxies, which process and forward SIP messages or SIP User Agents that is they create and terminate SIP messages.

Figure 6 shows the IMS architecture. The Policy Decision Function (PDF) in figure 6 is a logical policy decision element that uses standard IP mechanisms to implement service based local policies in the IP media layer. The PDF basically provides the framework for a policy based admission control scheme. It includes functions such as traffic authorization, resource allocation and media grouping [10].

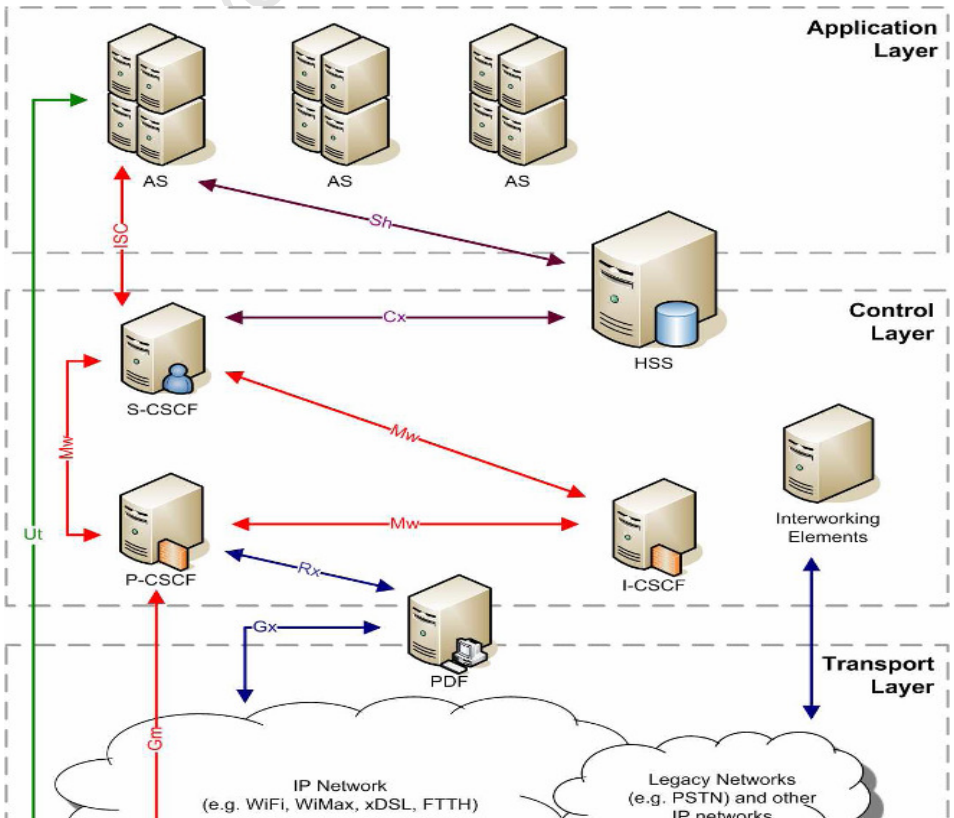

Figure 6: IMS core architecture [10]

2.2 IMS Protocols

There are various protocols employed in the IMS framework some of these play important roles in the all round functionality of the IMS network and others form support systems for the architecture. The various protocols include SIP, DIAMETER, SDP, IPsec, TLS and RTP. The following discussion gives information on some of the primary protocols used in the development of the gaming application.

2.2.1 Session Initiation Protocol

The session initiation protocol or SIP is a light weight text based protocol developed by the Internet Engineering Task Force (IETF) and currently a specification under them. It was primarily designed for VOIP solutions but has since expanded to a wider range of multimedia and data transport over an IP network. It also allows for multiple sessions to occur regardless of the media content. Currently there are numerous vendors worldwide developing hardware that supports SIP based services as they believe that SIP is venturing toward providing a means to create a universal platform for communication. There are large ranges of devices currently on the market such as IP-phones, network proxy servers, media servers and application servers that all utilize SIP. Due to its prominence along a wide variety of communication facilities it was determined that the IMS architecture base majority of its functionality on the SIP protocol [11].

SIP was designed as a protocol very similar to that of the HTTP or Hyper Text Transfer Protocol developed for the internet or web based applications. It has some similarities to that of the Simple Mail Transfer Protocol, SMTP used in emails. SIP however does have certain limitations. But because of its interoperability with various other protocols, it is ideal as a base on which to build improved service platforms for Next Generation Networks (NGN).

SIP is an out-of-band protocol, this is the reason why it can concurrently run different media streams during a single session by using the different ports available (see figure 7). It is an application layer protocol based on the OSI model and is also transport layer independent, this feature helps with its interoperability appeal [11].

SIP was designed to carry out the following:

- Provide means for establishing calls between a caller and a callee over an IP network.
- It informs a callee that a caller would like to create a session between them and it also allows the two entities to agree on the media encodings each of the two will utilize during the session.
- Allows for users to have different IP addresses as users may move from one access point to another. The manner in which SIP allows this to occur is by giving each user a unique address in mnemonic form such as email address. As they move from terminal to terminal, a DHCP for the local area would then assign an IP address to the user.
- Help to merge different access networks by providing interworking between them.
- Provide mechanisms for call management. An example of this is that SIP allows for the addition of various media streams during a session or call. It allows for encoding changes, addition of new participants, call transfers and call holding during a session [11].

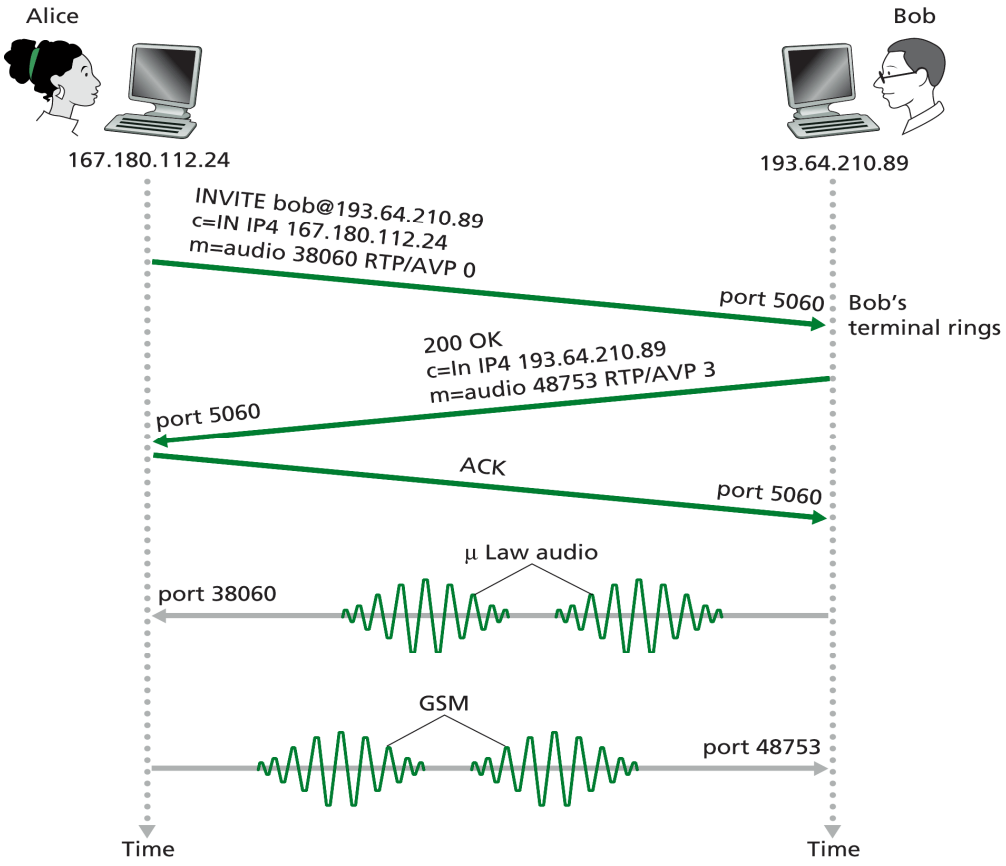

Figure 7: SIP call establishment [12]

Figure 7 illustrates a basic SIP call invite. In the figure, Alice already knows Bobs address and sends a request message for a call to be set up. In the SIP message she also indicates the media encoding she plans to use during the call and the port number through which she will be establishing the call. Bob acknowledges the call request and also attached the media encoding he plans to use and the port number. Once Alice acknowledges the call session specifications by sending an acknowledgement (ACK) the call commences. To end the call either one of the participants need to send the other a BYE message, and the call will be ended once the other acknowledges receiving the BYE message and returning a response (200 OK) message. As can be seen SIP tends to form a similar type of handshaking agreement like that of various other communications protocols. That is that every message request needs to be acknowledged upon its delivery to a SIP entity [12].

As previously mentioned SIP tends to use mnemonic address for its users. In figure 7 caller knows the users domain IP address. Usually this is not the case as bob might have multiple SIP devices each with its own IP address. Normally, the caller only knows the users SIP address which could be in the form of bob@uct.ac.za. In order to obtain the actual IP address, SIP uses a form of name translation and users location. Before a SIP application can be used by any user they first have to register themselves and the devices they are on with a SIP registrar. In the case of the IMS architecture that is the Home Subscriber Server (HSS). The SIP registrar as mentioned in the case of the HSS has information relating to a user such as service restrictions and encoding schemes. It also contains the relative addresses of users within its domain. During a registration process the user sends a request first to a SIP Proxy similar to that of the PCSCF. This proxy then assigns the SIP user an IP address. Thereafter forwards the information to a SIP registrar server which then acknowledges the registration and awaits further session requirements from the user or other SIP users looking to communicate with the user. SIP also makes use of Redirect servers which aid in the inter domain communication [12].

The following is a list of the features of SIP that separate it from other signaling protocols [8, 12]

- SIP is a text based protocol, and so allowing for better implementation as its easily readable and easier to debug.
- SIP uses similar attributes to that of the MIME protocol used in emails. The SIP message body may contain various media types. By using the description provided in the MIME protocol it can allow for various media to travel through the same session.
- SIP is considered backwards compatible enabling it to work with already established protocols.
- It is transport layer independent. Therefore, the underlying transport could be anything from TCP, UDP or ATM.

- It allows for multi-device leveling and negotiation. If a user's device supports both video and voice and another's only supports voice then it can setup a session that allows the transfer of only the voice aspect of the session.

The following Table describes the various SIP request methods and their basic functionality.

METHOD	DESCRIPTION
INVITE	Indicates a request from a user to another to participate in a call session
ACK	Confirms the reception of an INVITE request
BYE	Terminates a call
CANCEL	Cancels any requests issued
OPTIONS	Asks for the capabilities of a SIP server
REGISTER	Registers a SIP user with a server
PRACK	Provisional acknowledgement
SUBSCRIBE	Subscribes to an event of notification
NOTIFY	Notifies a subscriber of an event
PUBLISH	Publishes an event
INFO	Sends information on a session but does not change the state of that session
REFER	Asks the receiver to issue a SIP request
MESSAGE	Used for instant messages in SIP
UPDATE	Modifies a session state without changing a dialog state

Table 1: SIP request methods [14]

The OSIP and EXOSIP

The OSIP is a SIP stack that is used mainly for SIP Proxies. The eXosip is an extension to the OSIP libraries. The eXosip library deals mainly with the implementation of SIP functionality around the end points which comprise mainly of user clients and application servers. Hardly any of the core functionality is based on the eXosip libraries. The UCT IMS Client and the test bed use the osip and eXosip

library for a vast majority of their functionality. The eXosip functionality supports a majority of the SIP methods. These are REGISTER, INVITE, INFO, OPTIONS, UPDATE, REFER, PRACK, SUBSRCIBE, NOTIFY, PUBLISH and MESSAGE. The library though does not support Real Transfer Protocol (RTP), audio interfacing and Session Description Protocol (SDP) Negotiation [15, 16] .

2.2.2 DIAMETER

DIAMETER is an Authentication, Authorization and Accounting (AAA) protocol. It was developed by the Internet Engineering Task Force (IETF). It is used to provide AAA services for access technologies. DIAMETER is very similar to its predecessor the Remote Authentication Dial In User Service (RADIUS). The DIAMETER protocol can be divided into two fields: the base protocol and the application [3, 6, 17].

The base protocol is used for transferring DIAMETER data, negotiating network capabilities, error handling and extensibility. The application part describes application specific functions and their data. Each application is specified unilaterally by the DIAMETER protocol. Diameter makes use of the Internet Protocol Security (IPSec) and Transport Layer Security (TLS) to secure the connections [17].

DIAMETER is used by the IMS network to provide access to the HSS within the IMS network. As part of its improvement from RADIUS, DIAMETER can use both UDP and TCP as transport layer protocols. [17]

2.2.3 Real-time Transport Protocol

Real-time Transport Protocol is an IP protocol that supports multimedia data such as voice and video. It is widely used in the internet for streaming and in the IMS it is used for similar purposes. It is used to identify the media codec used during a

session, sequence numbering, time-stamping and delivery monitoring. RTP is an end to end delivery protocol and does not provide QoS guarantees [18]

2.2.4 Internet Protocol Security

IPSec forms part of the main security measures used in the IMS network. IPSec offers protection for upper layer protocols, IMS is mainly based on SIP which itself is an upper layer protocol. IPSec provides the following protection services:

- Access Control
- Data integrity
- Data Origin Authentication
- Anti-Replay Protection
- Confidentiality
- Limited Traffic Flow

It generally operates in two modes: tunnel mode and transport mode. Transport mode provides security for upper layer protocols, where tunnel mode is used to provide security between two gateways in the IP layer [3, 19]

2.2.5 Transport Layer Protocol

TLS is a security protocol used in the transport layer for internet applications. It supplies them with confidentiality and data integrity. It is a reliable protocol, meaning it functions or runs on TCP [20].

2.3 IMS Core

The following section describes the current state of the IMS platform which the CRG employs in conducting its research. It also discusses some of the details that go into the various components. As mentioned earlier the IMS core is made up of various components they are the CSCFs, the databases, Service elements and Support elements.

2.3.1 The PCSCF

The Proxy CSCF is the first point of contact between a user and the IMS core. Every SIP message passes through this part of the IMS network when a user is communicating with an IMS Core. The PCSCF conducts various tasks when it is accessed by a user. It provides security associations for a user with the IMS core. It conducts signal compression on SIP messages for efficiency through the IMS Network. It receives service information from the SIP messages and passes them to the Policy Decision Function (PDF). The PDF then conducts QoS provisioning and admission control for a user [3, 10]

2.3.2 The SCSCF

The Serving CSCF is the one of the main components in the IMS Core it mainly conducts processes similar to that of SIP Registrars. Its functions user registration, call setup, maintaining session states and it stores the user's service profiles. During

the registration process the SCSCF communicates with the HSS to obtain information regarding a certain user. This process will be described when the registration process is discussed. The information required includes a user's credentials such as what services are they allowed [3, 6, 10].

2.3.3 The ICSCF

The Interrogating CSCF is a stateless proxy. It is used mainly in inter domain communication. Every IMS domain has an ICSCF. The ICSCF handles the routing of SIP messages to a subscriber within the domain of the ICSCF. It acts similar to a Domain Name Server (DNS) in the internet. When a user registers with an IMS Core the ICSCF communicates with the HSS and then assigns the user an appropriate SCSCF. It can also act as a buffer between inter-domain communications by denying another domain access to its domain [3, 6, 10].

2.3.4 The HSS and SLF

The Home Subscriber Server forms part of the central user database in an IMS network. It contains user profiles within its domain. The profiles contain information ranging from security information, service information, location information and the SCSCF associated with a particular user. The main part of the user profile information held within the HSS is the users Private and Public user identities. The Private user identities are distributed by the network operators. They are used for registration, authorization and the subscriptions assigned to a user. The Public user identity is what other users use to contact a user. It also takes part in the routing of SIP messages to and from a user. The Public User identity takes the form of a SIP URI (Uniform Resource Identity). In a large network there maybe more than one HSS in the network, this then requires the introduction of a Subscription Locator Function. The SLF's main function is to identify which HSS is associated with a certain user. A range of its processes and that of the HSS will be further discussed within the explanation of the registration process [10].

2.3.5 AS and MRF

The Application Server (AS) and Media Resource Function (MRF) are the main service elements within the IMS network. Application Servers provide a range of services required by a user. Such services include conferencing, Presence, Gaming, IPTV and various other applications. The MRF provides media at the transport layer. It links up with the SCSCF and provides bearer related services such as conferencing [10].

2.3.6 Support Elements

The support elements are primarily made up of a number of devices that help with aspects such as charging, policy control and interworking within the IMS network. In the case of charging and policy control the, 3GPP release 7 defined an architecture based on the combination of two previous architectures and formed the Policy Control and Charging architecture. These two architectures were the Service Based Local Policy which is used in resource control and provides linking between the signaling and transport planes, and the Flow Based Charging architecture. The PCC architecture is centrally applied in two elements these are the Policy Decision Function (PDF) and the Policy Execution Points [10].

The interworking elements comprise of the Border Gateway Control Function (BGCF), the Media Gateway Control Function (MGCF) and the Signaling Gateway. These devices assist with the backward compatibility of the IMS network especially with regards to circuit switched based networks such as that of the PSTN and ISDN [10].

2.4 IMS Events

2.4.1 Registration

The IMS registration is conducted when a user would like to join their current IMS network and thereby allow the user access to the IMS services. During the registration the user's Public Identity is bound to an IP address issued to the user's current terminal. The IP address is issued to a user via the PCSCF which is the first point of contact with a user. In order to register with the PCSCF the user would need to know the location of the PCSCF beforehand. Also within the registration process a user can subscribe to various services and request for authorization to those services [10].

Figure 8 is an example of an initial registration message in the IMS Network. In the message the user places their IP address in the Via header, the PCSCF's address in the Route section and its Public User Identity in the From and To section. The Contact contains the IP address to be bound to the Public User Identity. The Expires header indicates the length of the bind in seconds [10].

Figure 8: IMS REGISTER MESSAGE

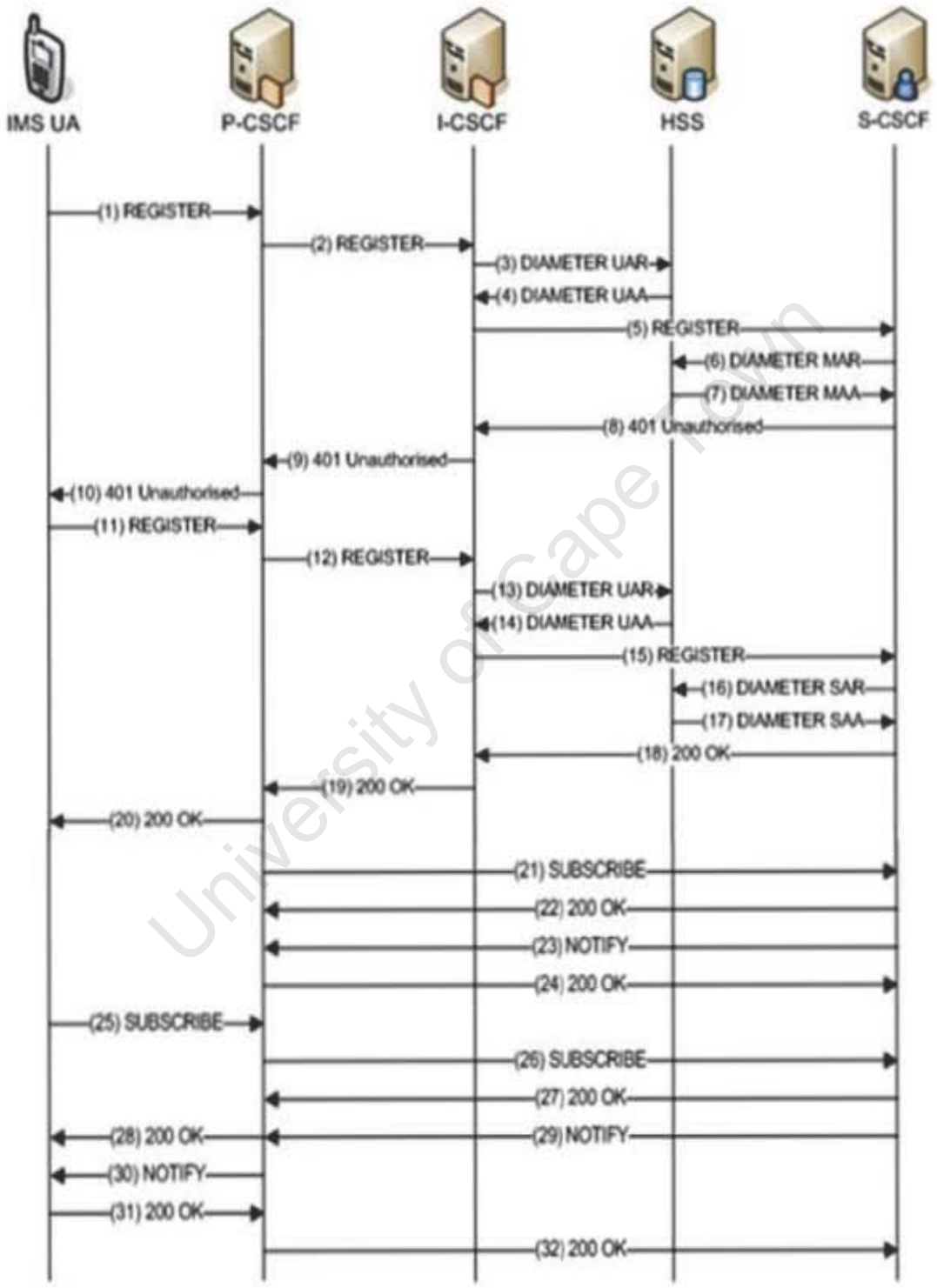

Figure 9: IMS Registration Process [10]

Figure 9 illustrates a basic registration procedure of a user with an IMS core. When a registration request is received by the PCSCF it replaces the Route header with the address of the ICSCF and adds its address to the Via header. It then proceeds to forward the message to the ICSCF. The ICSCF receives the REGISTER request and then queries the HSS using the DIAMETER protocol to find out which SCSCF is associated with the given user. Once that is done the SCSCF retrieves the authorization codes or vectors for that user from the HSS and forwards them to the user using the 401 Unauthorized challenge. The user then resends the registration request but now with the authorization vectors from the SCSCF the user is able to compute a nonce and a response. When the second register message is received by the SCSCF is replies with a 200 ok. It also retrieves the users profile from the HSS during this process. Once the registration is complete the user subscribes to a registration event which then allows other users in the network to know that the user is now registered with the network. The subscription will be discussed further in the following section [1, 3, 10].

2.4.2 Subscription

When a user is finally registered they can then proceed to receive IMS services. One way in which to do this is by sending a subscription request. The subscription request is generally the best means to link with an application server for a service. When a subscription request is issued to an application server (AS) from a user it first navigates via the PCSCF and ICSCF to the SCSCF. The SCSCF examines the subscription request header and determines through aspects such as the event header, the service profile (which describes the type of services a user has within an IMS network) and the trigger point to which AS the subscription should be forwarded [1, 3, 10, 12].

Trigger points make up what are known as Initial Filter Criteria (iFC). These iFCs are stored in the HSS and are normally linked to a user's service profile. Trigger points also allow for multiple service sessions to occur simultaneously. Each service usually has its own assigned trigger point this allows for organized routing of the SIP

messages. When a user registers with an IMS network its user profile is downloaded by the SCSCF. Part of the user's profile is the service profile. The iFCs basically differentiate the various services allowed to a user. When a user for instance subscribes to a conference service, the SCSCF receives the request analyses the subscription request and forwards the message to the Conference Server. The Conference Server then acknowledges the request and replies with a 200 ok response. With this response a dialog is established between the user and the AS. The AS then beings to send its messages usually via NOTIFY messages but still maintaining the dialog state. When user wishes to unsubscribe they merely send a BYE or Remove request and when the AS replies with a 200 ok the dialog is ended and the so to the subscription. Figure 10 below an example of subscription message. It is important to note that this is the initial subscription message; also significant is the Expire field. One way in which the IMS ends a subscription is by re-issuing the subscribe request and setting the expire value to zero [10].

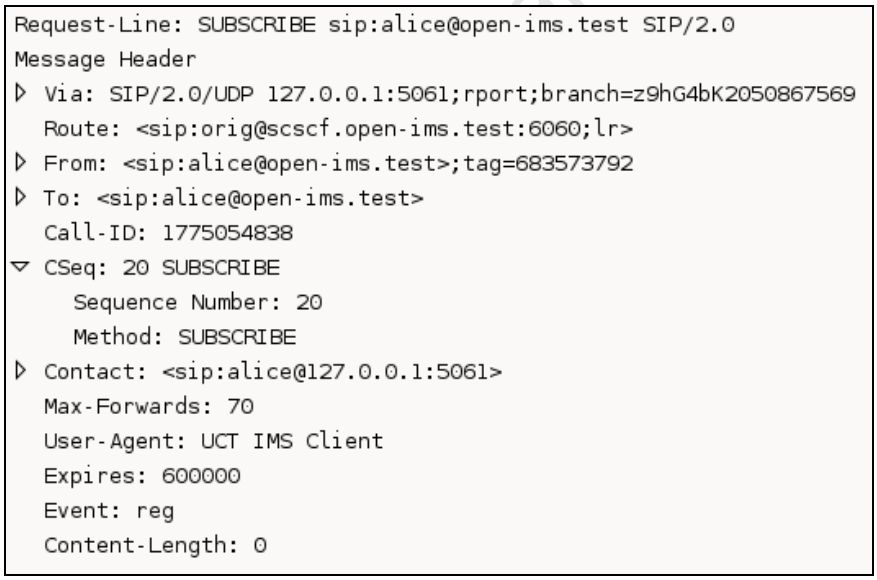

Figure 10: IMS SIP subscription

2.5 Electronic Game Play

The gaming industry is a large distributed industry. It provides a form of innovative entertainment. The growth of the gaming industry in the last two decades has been astounding. Currently more people are becoming computer literate and technologically apt, therefore the industry is on an exponential growth curve. Some of the biggest video gaming providers such as Sony, Microsoft and Nintendo feel as though they might be heading towards a saturation stage of the gaming industry and would prefer to keep it on the rise. They are aiming to accomplish this is by expanding to the mobile arena. The main problem with that scenario is that users will not readily prefer to carry a handful of mobile devices; they would prefer to carry a single device with multiple capabilities. For this reason these companies are now working concurrently with mobile companies such as Samsung, Ericsson and Nokia/Siemens in developing devices to handle most if not all forms of mobile communication [21, 22].

Currently consoles have network facilities available to them; they may be connected to the internet and can allow for high quality online game play. A disadvantage with the consoles is the lack of mobility, considered a hindrance in the expansion of the emerging online gaming markets. Should they manage to effectively expand their product to the mobile markets they would look to drastically improve their revenues.

Companies such Playstation and Xbox offer high-speed internet connectivity for their users, although the game-play is solely maintained within their own networks. This would mean the online game-play of for instance a game which is on the Playstation and on the Xbox would be contained within the respective networks. It is assumed that should a user wish to play against someone who was on a different network it would not be possible. The IMS through various gateways could be used as a means to get around such issues and thereby expanding the arena of play for members of both networks.

2.6 Related Work

Currently there is a lack of material on related work as the IMS architecture is still in its infancy. There is though a group that has worked on a gaming for mobile devices in IMS. The main concept of their work is based on the notion that players on mobile devices are more flexible. They have the freedom to play their games in any location and on any network. The players would also have the ability to move about while a gaming session is in progress. Game developers are currently looking into ways in which to spread their product. The ability for them to have a game playable anywhere and everywhere is greatly desired [23].

The research discusses how with the help of the IMS service capabilities they can achieve greater performance, scalability and provide good all round Quality of Service (QoS) to all games. They proposed and designed a prototype gaming platform for the IMS. With this platform they attempt to expand the social element of online gaming by making use of some of the other IMS features available to the users. By including features such as Instant messaging, presence, voice and video conferencing they attempt to make the gaming experience more complete. While doing so, they also try to push the marketability of the games by addressing a vast majority of the customers' social needs [23].

The prototype that was developed was designed to adapt current games on mobile phones to the IMS infrastructure. Their research also delves into ways in which one can create games for use with their platform [23].

Chapter 3

Design

3.1 The FOKUS Test Bed

The Open Source IMS is a free open source software package of an IMS network. It is designed for research, development and testing. It was developed in November 2006 by the FOKUS group in Fraunhofer.

The package has the main components necessary to implement a basic IMS core. That is it has a PCSCF, ICSCF, SCSCF and a HSS. The CSCFs have been written in C code and run using the SIP Express Router (SER). These take care of the IMS signaling and security. The support and service elements such as the PDF, PEP and Presence Servers have recently been added to the test bed at the UCT Communications Research Group (CRG). The HSS is a java based implementation. It is extremely user friendly and allows a developer to easily and rapidly implement new services to the IMS network. Figure 11 depicts the current state of the UCT IMS Network. Due to the continuous development certain services have yet to be incorporated to the network, but after more thorough and robust testing is completed they will be fully integrated into the basic design. [24, 25]

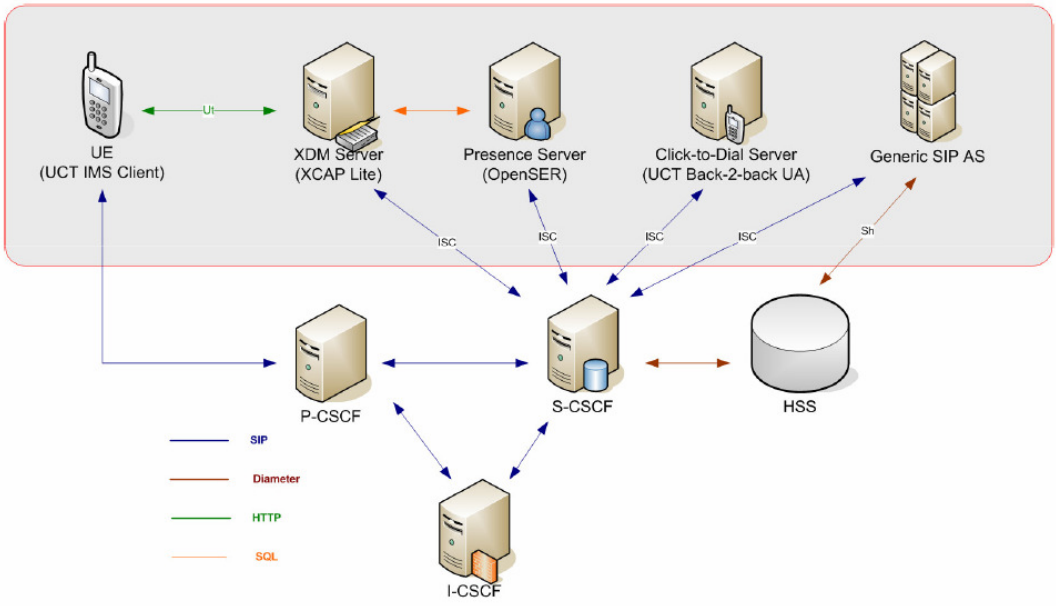

Figure 11: Current UCT Sandbox or test bed [10]

3.2 The UCT IMS Client

The UCT IMS Client was developed by the UCT CRG as a platform onto which various services could be added. It runs in conjunction with FOKUS test bed. They basically complement each other. The UCT IMS Client has a number of services but it is under continuous development so services are constantly being added. Some of these services include Presence, Instant Messaging, Conferencing, IPTV, VoIP, online address book and most recently interactive gaming. These services can easily be added to improve the gaming experience by offering aspects such as using the conferencing feature to interact with the other players. The UCT IMS client and the IMS network in general has the ability for all these services to run concurrently.

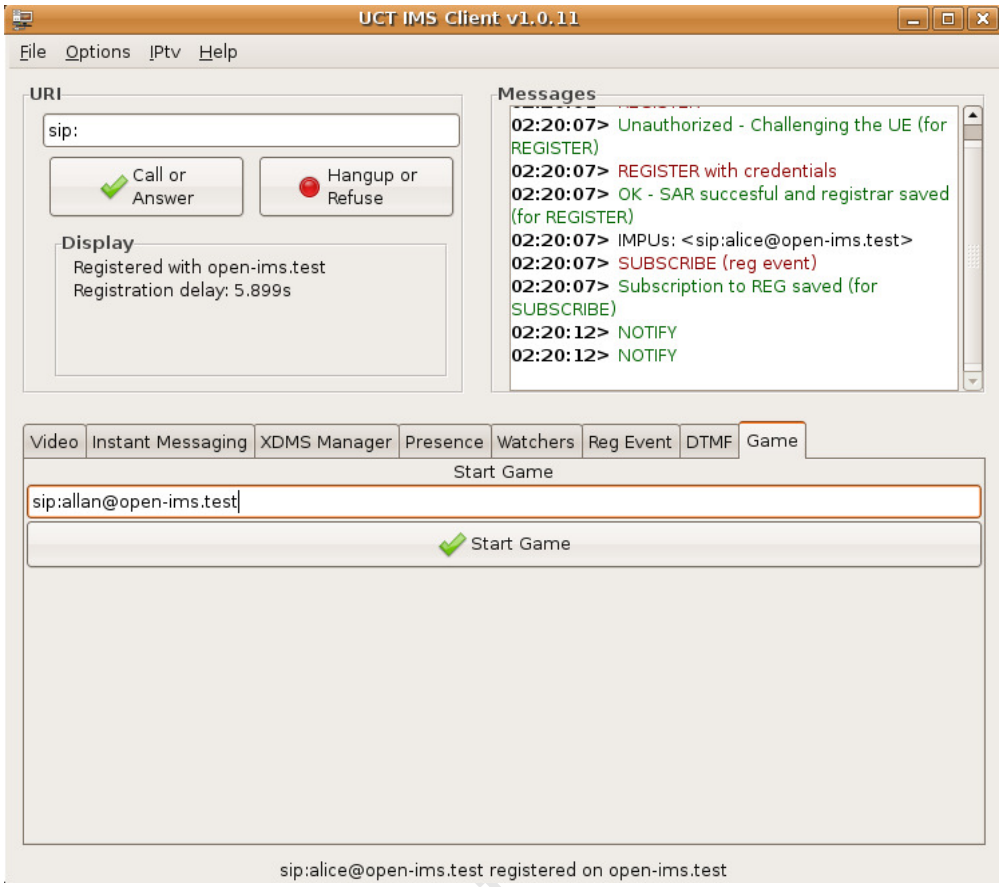

Figure 12: The UCT IMS Client

The figure 12 above is a depiction of the current UCT IMS Client. There have been other recent alterations to the design. As with the UCT IMS Network the latest version is yet to be incorporated. This version contains the gaming application client side as is shown more on the actual features is discussed in the subsequent chapters. The basic client has a range of features for instance it allows the user to decide on to which PCSCF they would like to be associated and therefore which IMS network to join. IMS networks differ in services offered therefore if a user is registered on numerous networks they may obtain different services based on their service profiles within each network. It is possible to merge the various services offered by different networks thereby negating the need for multiple registrations to various IMS networks [25].

3.3 Online Gaming Aspects

Online gaming currently allows two or more members to partake in a gaming experience. These members can be two or more users in a peer to peer format, users and a server in a client/server format or a hybrid model, where there maybe two or more users and a server involved. The following section goes into the various means in which gaming can take place in an online arrangement. As demonstrated in earlier chapters online gaming is an expanding market in the social entertainment industry. It is seen by many as an easily accessible means for people to occupy themselves [24, 25].

Online gaming has a variety of obstacles that determine whether or not a game can be successful. The objective of current gaming attempts to place gaming into a ubiquitous arena. Gaming providers strive to have their various games available to consumers where ever they may be. Due to network and hardware limitations of the past this has always proved to be difficult. The IMS Network tries to reduce the limitations previously imposed by providing a service delivery platform in which services such as gaming can easily be added. The following is a list of objectives that gaming designers observe in creation of online games.

In terms of the network

- Communication Range
- Speed
- Network Coverage
- Bandwidth
- Latency

In term of client devices

- Processor Power
- Memory

- Graphics
- Ergonomics

Online gaming or gaming in general can be classified by the following based on delay and latency [26].

- Real-time action games, which generally require network guarantees such as minimum delay and latency.
- Real-time strategy games, which also require some guarantees on network resources but are not as stringent as action games.
- Turn-based games, which are interactive but can allow for longer delays.

ONLINE GAME TYPE	**Required Round Trip Time (RTT)**
Action Game Good	RTT<200ms
Action Game Acceptable	200ms<RTT<600ms
Real-time Strategy good	600ms<RTT<900ms
Turn-Based Game	RTT>900ms

Table 2: Round Trip Time required by online gaming [26]

When considering the type of game to design all the above factors need to be observed. Games with a great deal of graphics cannot readily be played on mobile devices with low resolutions, low processing power and modest memory. Game providers must first decide on the target or main social sector before designing. Based on the requirements that the game to be designed should attempt to be ubiquitous in nature it was apparent that it would need to be relatively simplistic in nature. Online games like all games are governed by a set of rules. These rules determine how a game is played, how a victor is decided and how the results are

distributed. With these basic ideas in mind the creation of a game can be conducted [21, 23].

The IMS in the online gaming domain will look to broaden the experience of users from playing via a single terminal to that of multiple platforms. That is, users can for instance play a game on a mobile device against a user connected to a fixed line device or terminal. The IMS looks to offer a whole range of other services to the online gamer in conjunction with the online gaming service. Some of the areas that might cause issues with the IMS scenario are that of device compatibility though research is alternatively being studied by others on ways to address such issues.

3.4 Communication Models

In designing an online gaming application some of the most important aspects are choosing a communication model. As mentioned in the previous section these can be divided into a client/server model, peer-to-peer model and a hybrid model [21, 24].

3.4.1 Client-Server Model

In the client/server model the server houses majority of the functions required to run the game. The user would have to log on to the server and thereafter compute a series of interactions with the server during the game play. Once the game play is completed the server would then close the connection and move back into its idle state. The server should preferably have the ability to add more than a single user. Figure 13 below illustrates how the model could be implemented in the IMS Network.

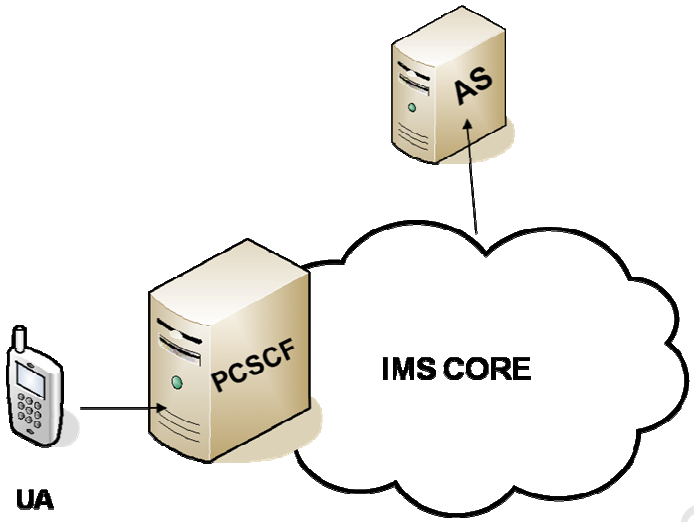

Figure 13: Illustration of the Client-Server Model

The client/server has various advantages and disadvantages based on how the game is distributed. Two methods for the game play distribution were considered. The first method requires that all the main functionality be placed at the user end of the established client/server connection. The server would only contain the rules that govern the means by which the user access those functions. The data required for actual game play would reside on the user or client side of the connection. Therefore the server would merely instruct the client on what actions to take. The server would also wait for a response from a client and then decide which instructions to send next. The following is a list of likely advantages and disadvantages of such an implementation.

Advantages:

- Generally this implementation utilizes few network resources as a server can simply send instructions the size of mere kilobytes which the client side can interpret.
- It allows for increase gaming complexity with more graphics as the latency is reduced.

- The server can implement greater encryption on the instruction messages without significantly reducing the performance.

Disadvantages:

- This method can be impractical for devices with limited processing power and storage capacity such as mobile devices. The games would have to be placed on the user's device and may used up a large amount of their usable storage.

The second client/server implementation observed was that of having the server have a majority of the functionality and the rules that govern the game play. The client would then have to download the game in order to play. The server may also continuously send information regarding the game such as a new level or a new component for the game play. As with the previous implementation this too has its own advantages and disadvantages.

Advantages:

- It is a more practical implementation for devices with limited processing power and storage capacity as they only need to house the gaming data while a game session is occurring. They may delete the game from their device once the session is finished.

Disadvantages:

- This method looks to use a greater deal of network resources.
- With increased need for data transfer game play over a persistent connection maybe hindered by the need to transfer large amounts of data.
- It is limited to the amount of extra security measures that can be applied before reducing game play performance.

3.4.2 Peer to Peer Model

The peer to peer model is shown in figure 14 is one where by the gaming service is conducted without the use of a central server. In this scenario one of the users is assigned the duties that would have been generally bestowed upon a server. The user or client would allow invites or subscription requests from other users and create a session amongst itself and the other users. The client acting as the server would then proceed to issue out the list of instructions as to how the game would commence and how the winner would be determined.

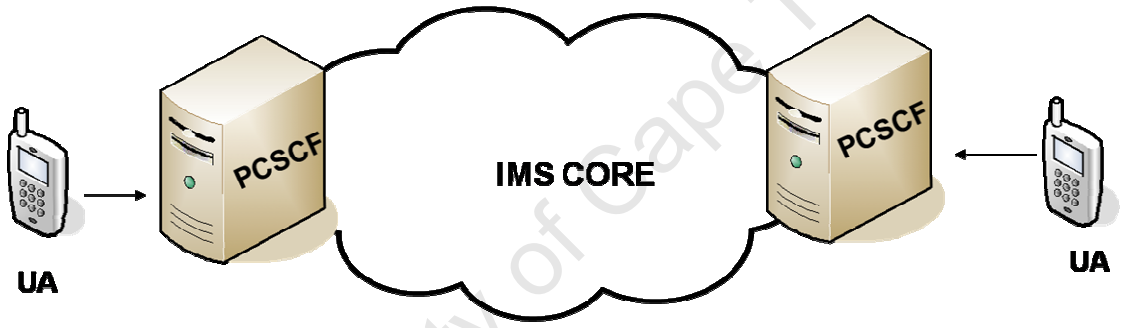

Figure 14: Peer to Peer connection model

As with the client/server model the peer-to-peer model has its own advantages and disadvantages

Advantages:

- There is no need for a central server users can utilize services such as the presence service to see whether other users are available and setup a gaming session at their own leisure.
- There are generally lower device compatibility issues as usually the devices on which the games are played are similar in nature.

- Network resource use is not as significant as the users would generally have the games on their respective devices and only the set of rules or instructions that govern the game would be sent through the session.

Disadvantages:

- The games would usually have to be on each of the clients' respective devices if not they would have to download them. This as mentioned previously can be impractical for certain devices.
- If a client acting as a server has multiple users with gaming sessions running simultaneously this can lead to a drop in performance of the entire session as the load on the client acting server increases.

3.4.3 The Hybrid model

The hybrid model as depicted in figure 15 is a model in which majority of the aspects in both the client/server model and the peer-to-peer model are incorporated. The model generally would make use of the strengths of both the models.

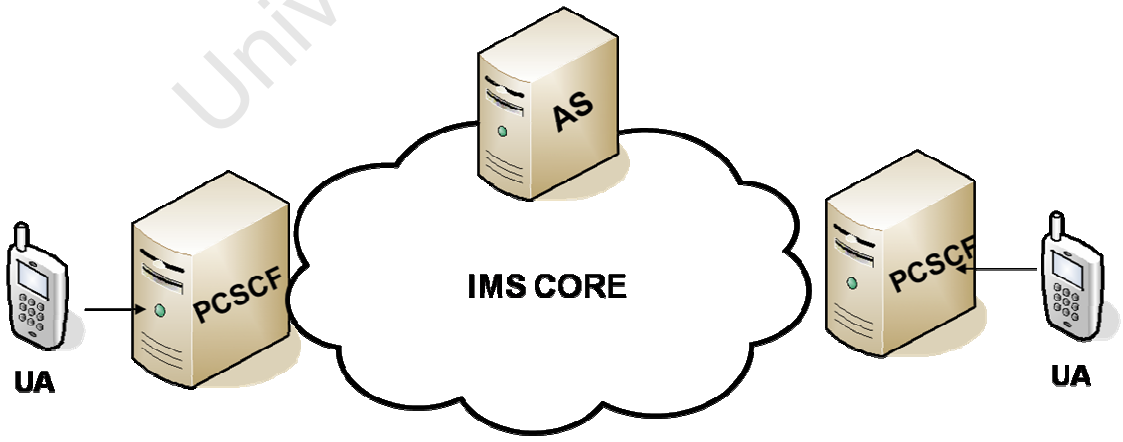

Figure 15: Hybrid communications model

All these communication models have various advantages and disadvantages but these aspects ultimately depend on the game types which shall be discussed in the next section.

3.5 Game Types

Games fall into the following categories; Puzzles, Arcade, Action, Strategy, Sports, Simulation, Cards, Casino, Role Playing Games and Word games. After considering the various game-types based on complexity and ease of implementation, cards were chosen as the game-type for the research. There is still a great market for such games and they can readily be applied to various user terminals in an IMS network. Card games can also be easily implemented and require few hardware, software and network resources as opposed to other games, therefore increasing their ubiquitous appeal [24, 25, 28].

Based on the fact that the gaming category chosen for the application was that of cards the rules that govern card games are relatively simple to implement. Online card games require that the user see pictures of the cards as played. These are generally offered in jpeg format, bitmap (bmp) format or similar file types. Pictures of cards, based on average resolutions and format, tend to range from a little kilobytes to a few megabytes. Therefore the communications model adopted for the gaming application could be any of the ones previously discussed. For the gaming application it was determined that a hybrid communications model be applied to application. The hybrid model would offer the following advantages and disadvantages to the game type proposed:

Advantages:

- It would first have a central server that would act as an “always on host”, thereby users need not be online to provide game play to other users as they can just play in a server/client format.

- The server would house the general rules of the game play and the users' terminals would house the actual games this would allow for greater scalability as more users could be added to the game without drastically reducing the performance of the server.

- With the hybrid model and the current game type chosen users can decide on whether or not they would like to play solely against each other, against the server only or against their peers and the server combined. That way introducing a whole range of variables thereby expanding the game play probabilities. With card games the server can randomly issue cards and the peers and the server can decide on which combinations to play. An example of this would be a game of poker; users receive the cards and decide on how to play them based on what they believe their opponents have.

- Other advantages would include the low use of signaling resources as the games would be housed on the users' devices which would lead to other advantages such as increased security on the messages sent to and from the server.

Disadvantages:

- The one major disadvantage with the model would be compatibility issues between users' devices. Some may perform worse than others and therefore reduce the performance of the entire game. Generally speaking games using this model would usually perform based on the performance of the lowest rated terminal or device generating a bottleneck effect.

- The games would be housed on the user terminal devices therefore creating some resource challenges for certain user devices as stated in the previous section.

3.6 The Game Server

The Game Server design allows for various types of card games to readily be implemented by just adjusting the rules for the various playable games. The main functions of the Game Server include waiting for incoming subscriptions, adding members to the subscription list, shuffling a deck of cards and sending the cards to the user or users. The game server would also notify users of others players within the game. The server should also receive notifications of the current status of a game from all the users and continuously update the members of a gaming session.

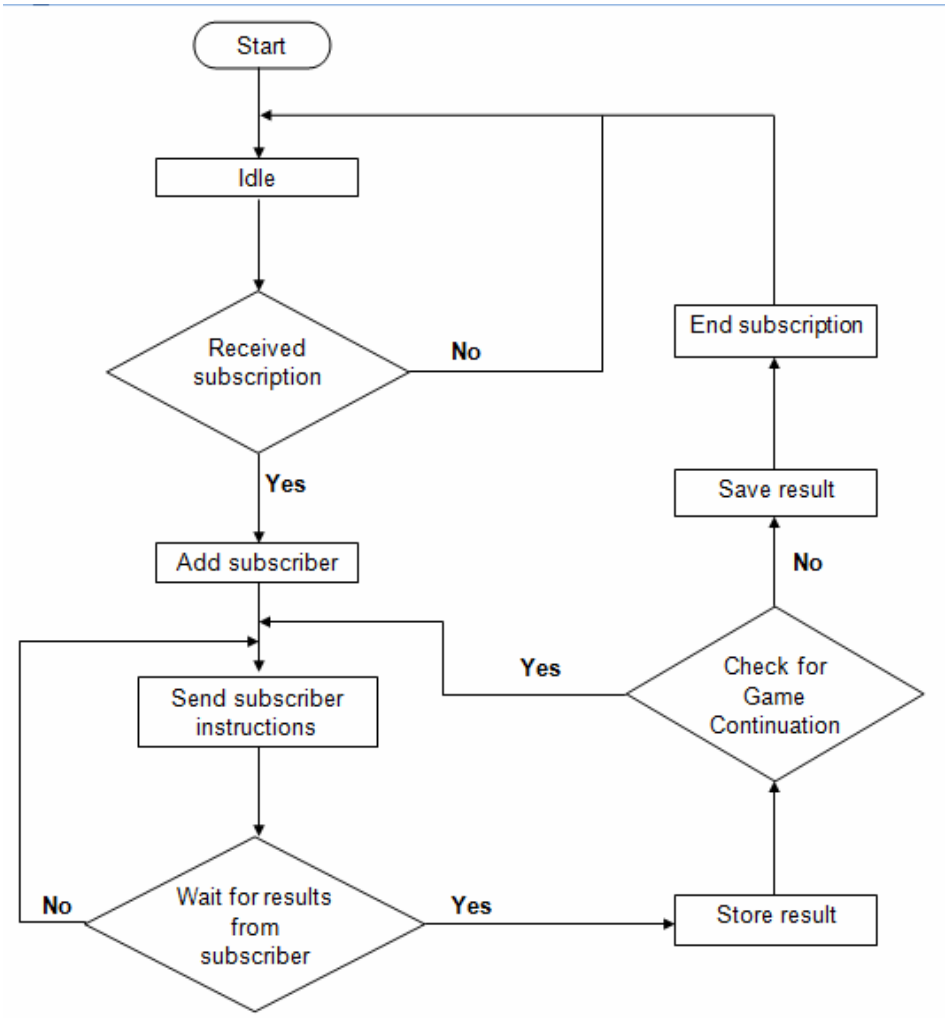

Figure 16: Flowchart of Game Server Function

The flowchart displayed in figure 16 illustrates the general flow of the functions to be carried out by the Game server for a particular client. The server starts in an idle state, and then waits for reception of a client subscription. Once the subscription is received the user is added and game play commences with the server sending instructions to the user. The server will then wait for results if none have been received it will send further instructions. Once a result has been reached the result is stored and the game server then checks to see if the user would like to continue. If the user wishes to continue the server will send another set of instructions. If the user does not wish to continue the server saves the results and ends the user's subscription. It will then return to its idle state.

3.7 The Gaming Client

The UCT IMS client gaming user interface has the following functions it sends an outgoing subscription stating the type of game it would like to play. It then waits for a response thereafter it receives notifications from the Game Server with the gaming rules, other players within the current session and the status of the current sessions. Figure 16 illustrates the format in which these functions would be used.

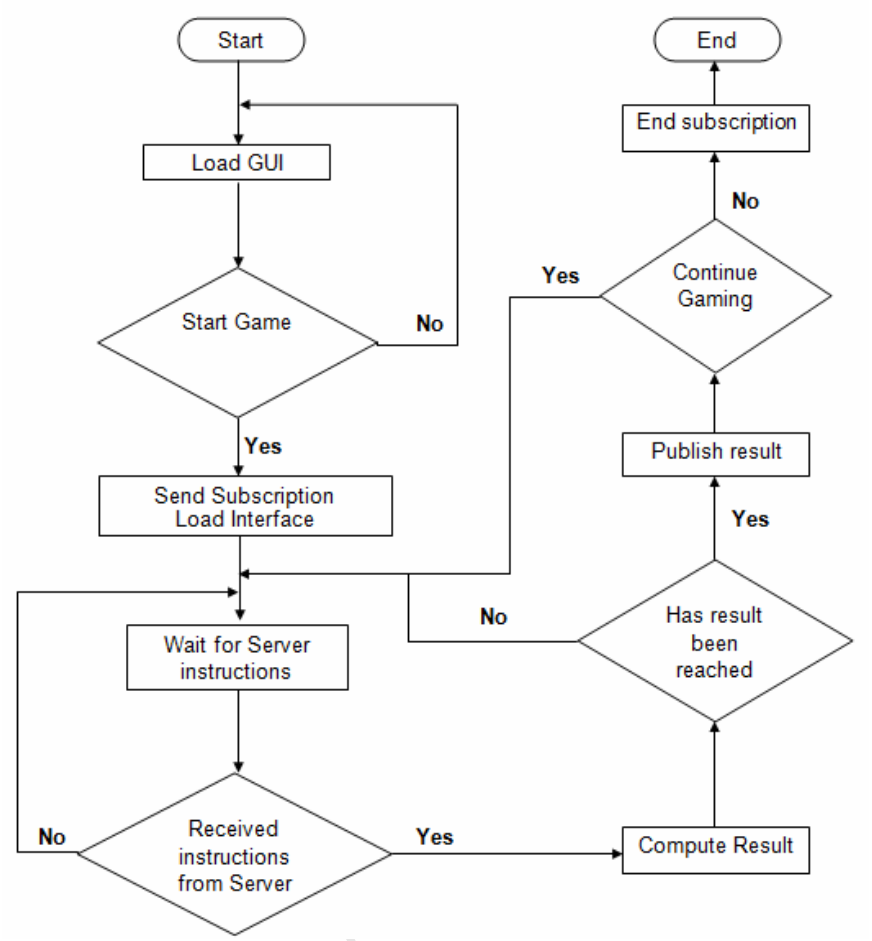

Figure 17: Flowchart of Game Client Function

The flowchart in figure 17 depicts the client side of the gaming functions. When a user initiates a gaming service the UCT IMS client would load a Graphical User Interface (GUI) for a particular game. When the user opts to start the game a subscription will be sent and the particular game interface would be loaded. The client would then wait for instructions from the server on what actions to take next. Once the instructions are received the client would compute a result. If no result is reached it would then wait for further instructions. If a result is reached it would notify the server. If the user would then like to continue with the game play they would merely wait for further instructions if they would like to end the game they would notify the server and end the subscription.

Chapter 4

Implementation

The implementation of the gaming application was conducted using the various tools available. The research and development was all conducted using the Linux Ubuntu 8.04 operating system. The programming was done using C. The Graphical User interface (GUI) was done using the same programming tool as that of the UCT IMS Client which was Glade Interface Designer 2.12.2 with its respective libraries gtk and gnome. The signaling conducted by the gaming service was done by manipulating the various IMS SIP Methods available in the UCT IMS Network. This meant that no excess signaling would be needed in the implementation process. The Linux based network tool Wireshark was used for the analysis of the signaling messages.

4.1 The IMS Gaming Architecture

In the design of a game there are four main aspects to consider game play these are the control descriptions, playfield descriptions, rules of the game and what are the requirements for completing a game. The gaming application developed as previously mentioned was based on the Hybrid model but it tends more towards a Client/Server model than that of a peer-to-peer model.

In the gaming application created the clients each have the majority of the functions necessary to run a game. The server strictly determines what cards to send to the respective users. The UCT IMS Clients and the server, based on the game being played, determine who the winner is. After which appropriate actions are taken. Aspects of the game play such as what a player obtains for succeeding were not added but can readily be applied to the gaming application. The application was designed such that expansion and improvements can be readily applied without great difficulty.

Figure 18 illustrates how the implementation of the gaming application was conducted. The figure shows the required signaling that takes place in conducting a gaming session. It was determined that for a gaming session to occur a user or client would need to send a subscription to the gaming server. The server would then complete various tasks with the client during the session, these tasks and others alike will be discussed later.

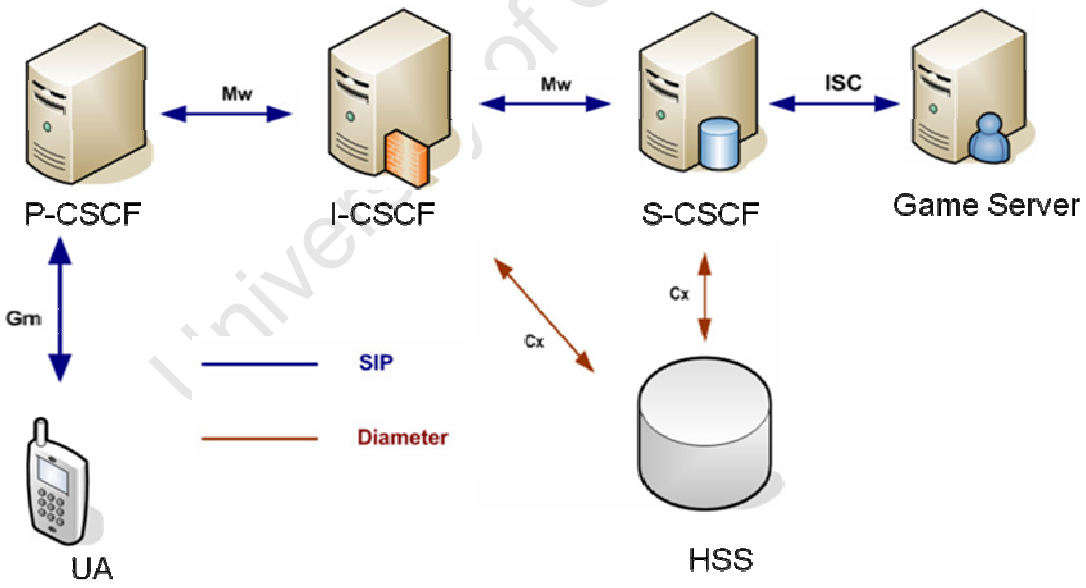

Figure 18: Implementation of Gaming Application Components

As the gaming session is created on a subscription basis a user would first have to register itself with the IMS network. During this process the user's service profile

would be downloaded by the SCSCF from the HSS. In the service profile as mentioned before is a section indicating which services are available to a user. The service profile will contain game event trigger point. This trigger point shall indicate to the SCSCF that any SIP messages from a user with an event type aptly named “game” are to be forwarded to the Game Server. Figure 19 depicts the current trigger points implemented in the UCT IMS network for a gaming event. The trigger points are housed in the FOKUS Home Subscriber Server (FHoSS) used by the UCT IMS Network. The SCSCF forward SIP messages with either a PUBLISH, MESSAGE or SUBSCIPTION method from the UCT IMS client to the Game Server. As to not confuse the messages with other services utilizing similar methods each method will need to have the event type attached to the SIP header. Figure 19 displays the manner in which this is done. The SIP Header content entry is where the event type is placed.

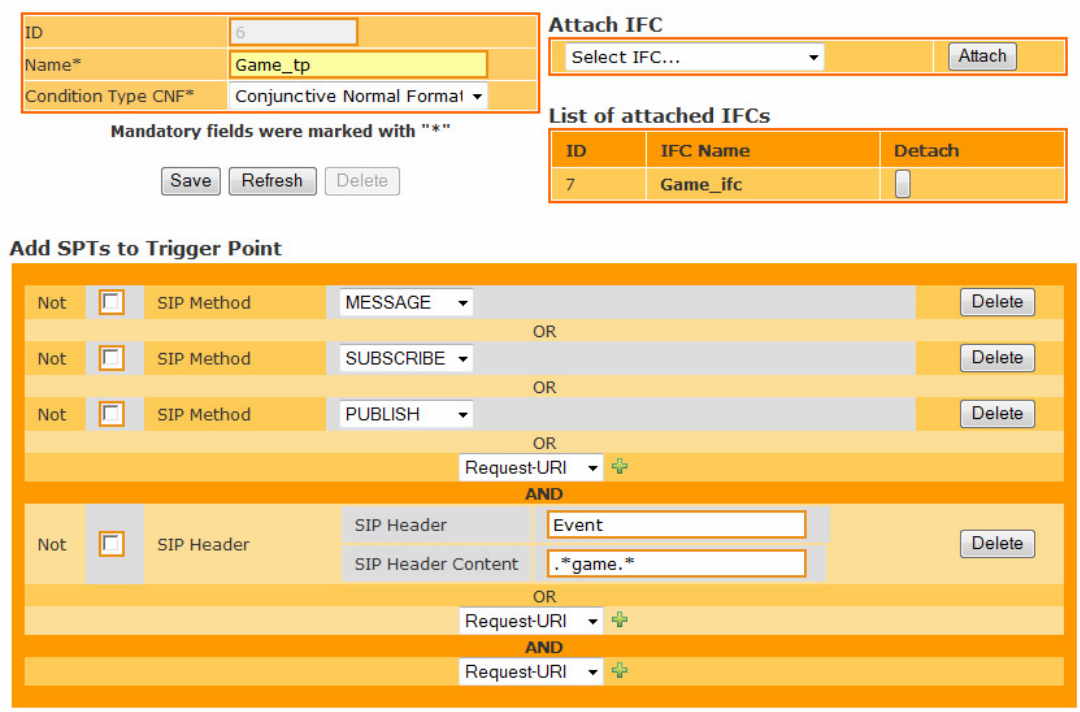

Figure 19: Game Application Trigger points in the FHoSS

4.2 The Game Server

The Gaming Server was designed with the ability to have its attributes expanded based on the type of game play required. The functions within the Game Server are merely a set of functions that govern how the signaling during game play is conducted. It also decides on the course of action when it receives input from the user. The following is a list of the functions done by the Game Server:

- The Server idles and waits listens for SIP messages on its port.
- When a SIP message arrives on its port it observes the header to determine whether the message is part of a gaming event, if it is it forwards the message to the relevant functions.
- It stores the list of users currently subscribed to a gaming event.
- It notifies the users of current state of the game.
- Tracks the progress of users throughout the game play.
- It deletes all the relevant data once a game session has been completed

More on how these are actually implemented is explained when the basic game implementation is explained. The game rules and how the winners are decided have been designed in such a way that the operators of the server can easily adjust them to suite their respective needs. The client side is designed in a similar fashion. It has the fundamental functions in place for the creation of an interactive gaming application. It is left to the operator to put in place their rules and regulations.

4.3 The UCT IMS Client Game Functions

As mentioned UCT IMS client contains the basic requirements for conducting a card game. At the client houses a majority of the functions for determining the game play. The first is it houses a database with a bitmaps that represent pictures of cards. It then awaits instructions from the Server on which ones to display on the client game graphical user interface. The interface like the other aspects can be easily expanded and changed in to favor the needs of the operator. This will be discussed further when the implementation of a practical card game is discussed.

4.4 Game Play

4.4.1 Card Game Implementation

The card game chosen for the gaming application demonstration was that of a Blackjack Game. Other card games could also be easily implemented using the same architecture chosen such as hearts, solitaire and poker. Due to time constraints the game of Blackjack was the only one fully implemented in the programming section of the dissertation, although implementation of a poker game prototype is also included but not thoroughly tested due to time constraints.

Blackjack or 21 as it is also known is a game whereby the player after receiving two to five cards attempts to compute a value close to but not above 21. The objective of the game is to beat a dealer or another player by having a value higher than them. The value of a card is dependent on the card's face value. Cards with face value 2

through to 10 have their face value that is a card with the number 6 written on it is valued 6. The cards with Jack, Queen and King all have a value equal to 10. The Aces can have a value equal to either 11 or 1 depending on which number the player decides to use in calculating a value close to 21.

Figure 20: Playing cards

To illustrate how the basic rules are applied the figure above (figure 20) has a 6 of Diamonds, an Ace of Spades and a Queen of clubs. In the game of blackjack the value of the suits make no difference to the result. The value of the above set of cards can be either 17 or 27 depending on the player. The player would most probably choose the value 17 as it is below 21.

4.4.2 Game Rules

As all games are governed by a set of rules the following are the set of rules conducted by the service in completing a single game play. In the gaming application the shuffle function housed in the Game Server determines which cards are sent to the players. When a gaming service begins the shuffle command is executed. After which the game server begins to notify the user on which cards to use where. The deck of cards used in the game play contains 52 cards the bitmaps to these cards are all housed on the client's terminal. Each card/bitmap has an index number

ranging from 1 to 52. Therefore in order for the game to be played the server needs to only send the index number associated with the relative card as opposed to sending the entire bitmap to the client. As illustrated in the SIP NOTIFY message in figure 26 the index is represented as plain text and when the client receives the NOTIFY is extracts that value and displays it on the clients interface as a bitmap. The following flowchart (figure 21) illustrates how the shuffle command was implemented. The shuffle function starts by generating an empty integer array of 52 cells. It then uses the mathematics functions in C to generate a random number between 1 and 52. It then checks to see if the number was previously generated if it was it generates another, otherwise it will add the number to the array. After adding a number to the array it would then check to see whether the array is complete if not it would continue the process of obtaining all 52 numbers randomly placed. This array of numbers would then act as the indexing numbers for the card game. Each index number has a corresponding bitmap image or card on the client side of the session. When the index is sent by the Game Server the client applies the index to its bitmaps and displays the card associated with the index.

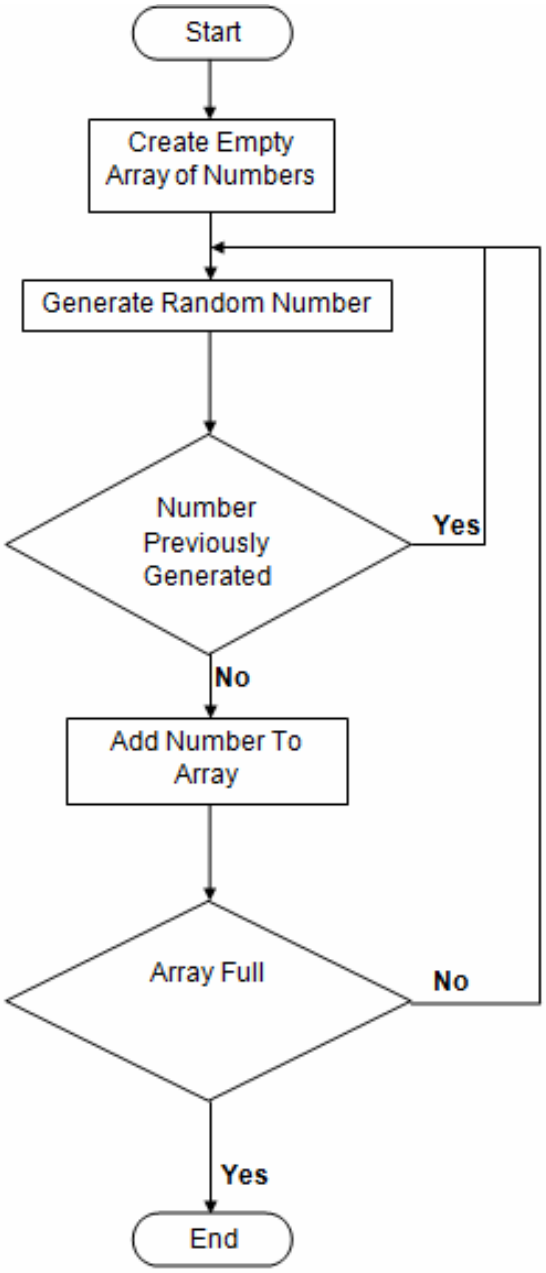

Figure 21: Flowchart of the shuffle function

4.5 Multiple Players

When there are multiple players the users can decide on whether they would like to partake in the same game or decide on whether they would prefer to play in a Client/Server mode.

In the first scenario players both subscribe to the Game Server as illustrated in figure 18. Once this is complete the server will then send a set of index numbers to the players in the game. Certain cards would be destined to be used only by a specific player. Each player though would receive all the others' cards but that of their opponents' first card. This would allow players to guess what cards are needed for a possible win, as they would not know their respective opponents initial card.

4.6 Signaling Implementation

The following section describes the general signaling implemented during a Client\Server gaming session. In order for a game session to occur a user needs to first be registered. Once the registration is done the user may perform a subscription request with the Game server. Figure 22 shows the type of signaling done during a normal gaming session.

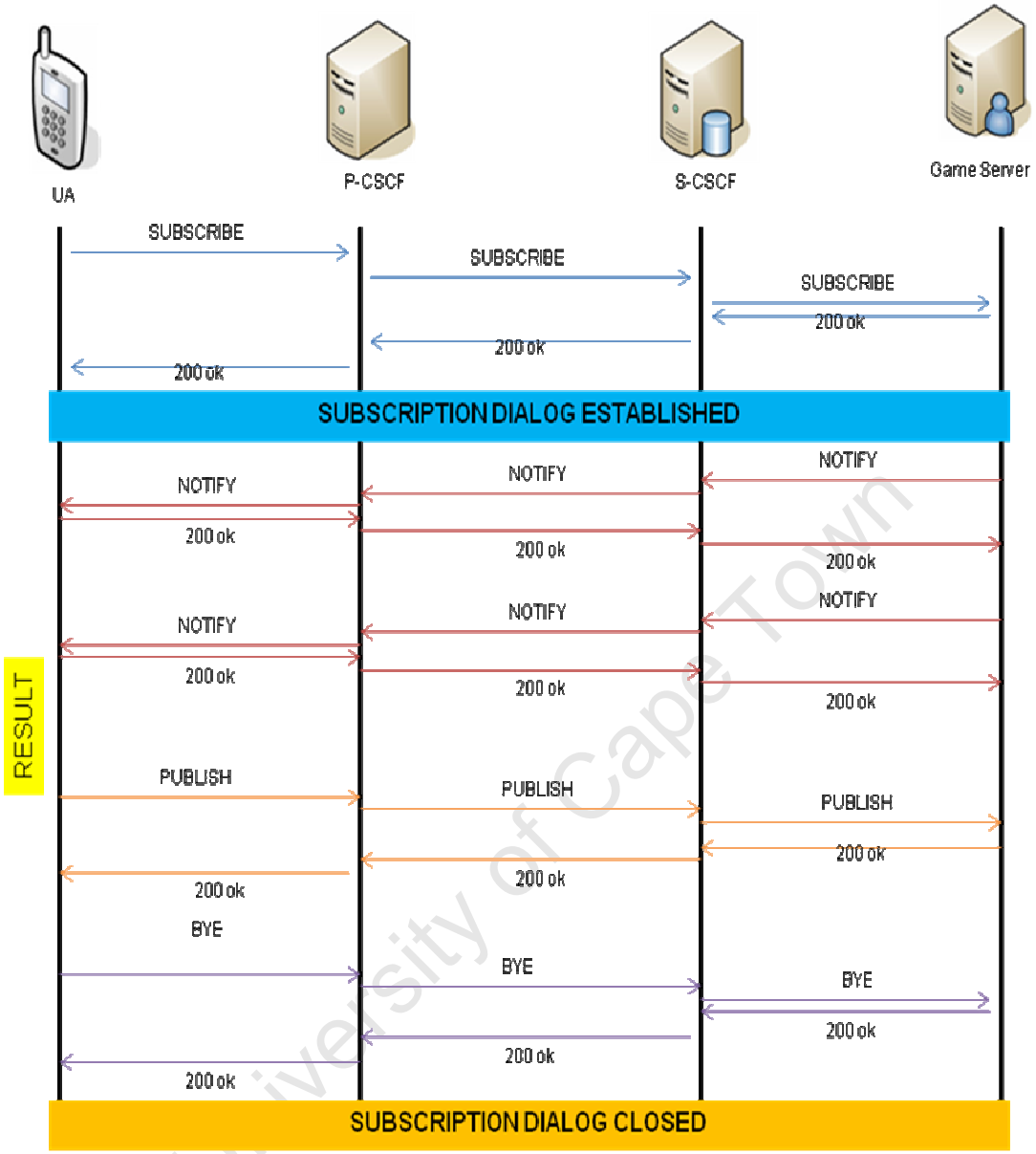

Figure 22: General Signaling during a gaming session

- The client sends a SUBSCRIBE message via the SCSCF and awaits a reply. If no reply is received it will resend the SUBSCIRBE until the subscription timeout occurs. Should a timeout occur the client will be notified by the SCSCF. Timeouts can occur if part of the network is faulty. An example would be if the Game Server is down.

- If for an acknowledgement from the Game Server in the form of a 200ok is received a SIP dialog is established and messages can be transmitted through this dialog.
- Once the dialog is established the client awaits instructions from the server on how to conduct the game these are sent through subscription NOTIFY messages.
- The client then computes the instructions received and determines if a relevant result has been reached if it has it sends a PUBLISH message which is outside the actual dialog to the server indicating the result.
- The PUBLISH message would also contain a set of instructions on whether how it would like to proceed with the game play.
- Should the client wish to continue the game play the client would then wait again for a NOTIFY and then compute another result.
- Once the game is complete the user can end the dialog by sending a BYE message.
- The server then replies with an acknowledgement and deletes the client from its data base.

4.6.1 The SUBSCRIBE Messages

The figure below (figure 23) depicts a typical subscription message used by the gaming service. The message as shown indicates the various addresses of the PCSCF, SCSCF and the clients address. It also has the Event type described has the string “game” indicating to the SCSCF the required application server to which the subscribe message needs to be forwarded. In the subscription it is as mentioned in chapter 2 to notice the expiry length. Should this value reach zero the subscription will be ended and a new subscription would need to be issued it is important to the operator or client to ensure that this time is sufficient for the game play. Another key point to notice is the Call- ID, this value will be used throughout the game play. It

corresponds to all messages relating to the dialog. Messages that fall outside the dialog will have differing Call-IDs.

Figure 23: Game Subscription Request

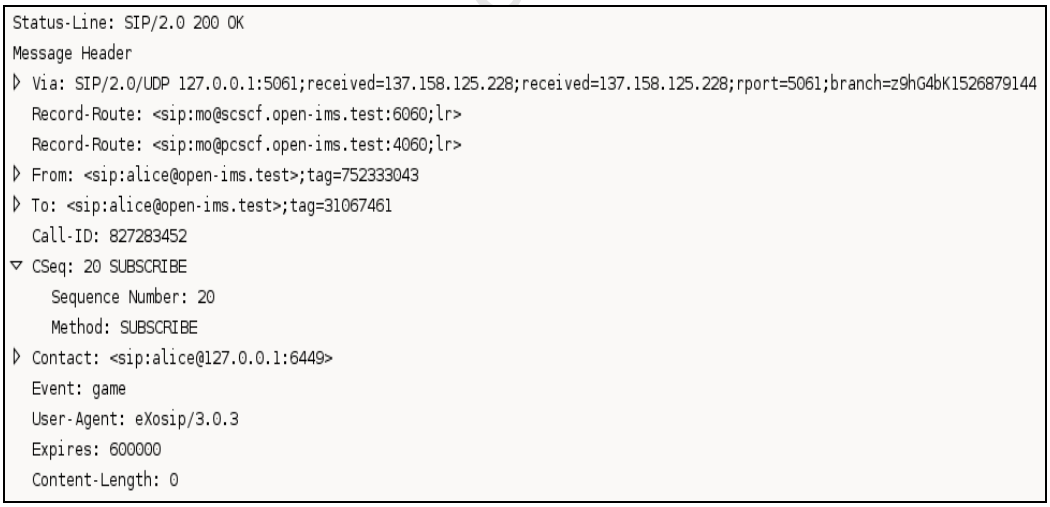

Figure 24: Game Subscription Response

Figure 24 displays a typical response to a subscribe request. It maintains the event type, and expiry time. The response makes use of the generated Call-ID to

determine where the data should be sent. The response also contains the address of the Game Server in the Contact field.

Figure 25: Game BYE message

The games BYE message is in actual fact another subscription request with the expiry time set to zero as mentioned previously. Once this is issued the IMS network releases all resources dedicated to the dialog.

4.6.2 The NOTIFY Messages

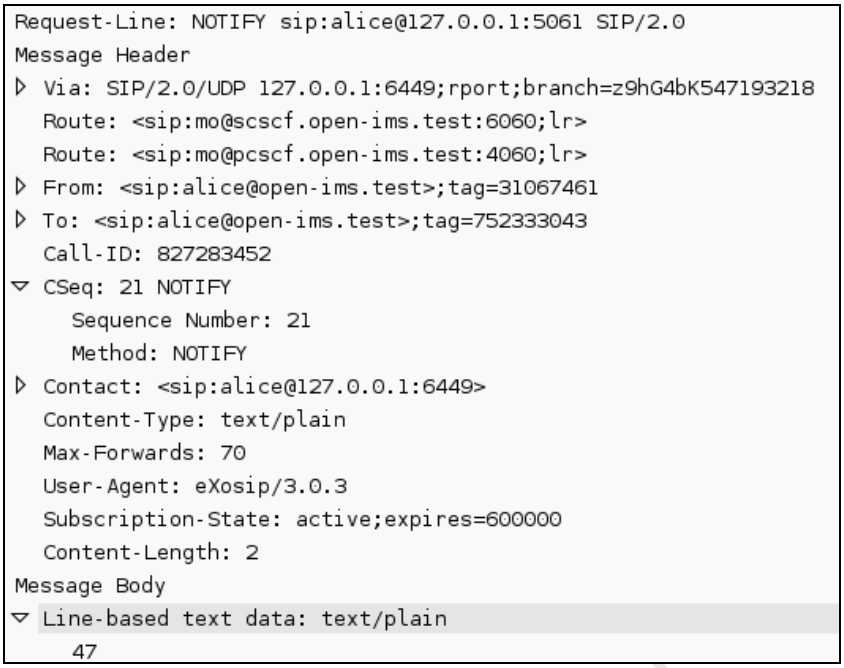

Figure 26: Subscription Notify message

The NOTIFY message has as shown similar details to that of the subscription messages the main difference is in the content type field. SIP makes use of features similar to that of the MIME protocol used in emails. It allows for various formats of data to be sent as part of the message body of the SIP message. With this feature is possible to transfer the cards as jpegs or bitmaps. For performance issues it was determined that the message merely send the index numbers of the cards sorted on the client side of the session. As is illustrated the format is specified as text/plain. As the NOTIFY is seen here as plain text is possible to apply simple encryption algorithms for additional security measures at the application layer without drastically reducing performance.

4.6.3 The PUBLISH messages

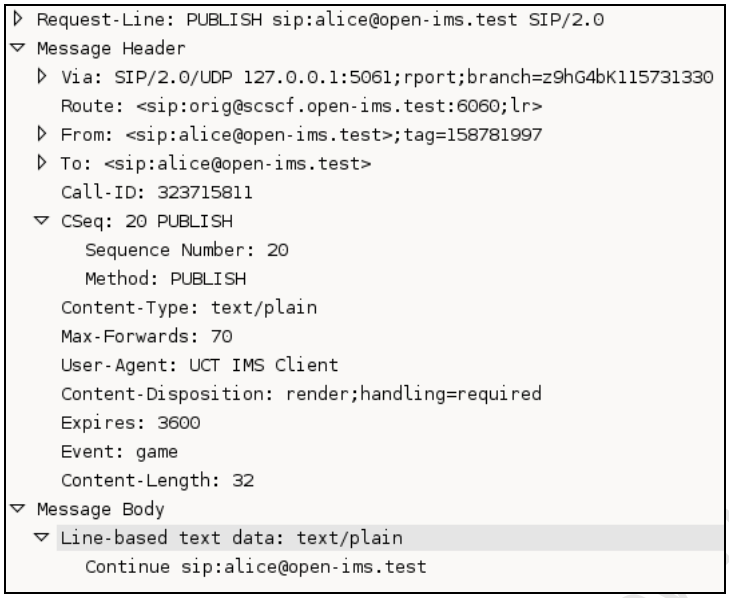

Figure 27: Game Publish message

The PUBLISH message shown in figure 27 illustrates that the message falls outside the dialog of the subscription as indicated by the Call-ID. It too makes use of the MIME feature of SIP, to send its content.

4.7 Gaming Application

The figure 28 illustrates the current GUI of the gaming window on the client side of the session. The window has a range of buttons and image holders placed on it. With the Glade Programming Tool operators and vendors alike can easily add further features to the interface. It is also possible for them to add advertising templates too which can be also used as another source of revenue. In figure 28 the GUI has two rows of card place holders but can readily be expanded based on the number of players. The number of rows is although limited to 6 in the current implementation due to the average resolution of computer monitors. In the figure the top row of cards represents the dealer or the user's opponent's set of cards. The bottom row represents the player or users set of cards. The Player button requests for more cards for the user from the Game Server. The Dealer button is used to request cards

from the Game Server for the dealer section. The Game Exit button is used to end the subscription. There is a text editor and a text notebook display below it. The display issues a status about the game to a player and the text editor is used to send messages to the server about choices that can be issued. For instance the server may send a request to continue the game on the display. The player would then type in a yes or no to the continue request and then click on the player button to complete the task. The GUI is designed to be able to display all the cards available and to make it known to the user depending on the game chosen their opponents hand. Its design is scalable to any card game type and can be easily adapted to other formats based on the rules that govern a particular game. For instance in the game poker the user is not allowed to view the opponents initial two cards but can see the cards dealt out by the dealer with which they are to play with. It also has a text box editor whereby a user can place a bet. As mentioned other boxes and buttons can readily be applied to suite all forms of games.

Figure 28: Game Graphical User Interface

The following chapter looks into some of the aspects that would relate to the performance of the gaming application. The results received were documented and analyzed. After which conclusions are drawn and recommendations made.

Chapter 5

Results

The following chapter evaluates various performance elements in providing the gaming service on the UCT IMS network. On completing the gaming application certain tests were conducted to give a relative idea of its performance in a real world scenario. For each area tested a discussion on the performance analysis is made. The following set of graphs and figures illustrate features such as average packet size and relative latency during game play. The figures and graphs were all taken using the Wireshark network statistics tool in Linux. It performed analysis on a gaming session conducted on the same personal computer. That is the Server, the IMS Core and the client were all on the same machine and they were merely

separated by their respective port numbers. This then meant that the results received would primarily be based on the computers performance especially in terms of processing power and memory.

5.1 Gaming Session Summary

The window in figure 29 is a screenshot of a summary to a gaming session. The logging of the results begins from the registration process through the gaming subscription and dialog to deregistration. The important column to note is the Displayed column. The Displayed column refers to all the SIP messages sent and received during the entire process. The main areas of concern, as mentioned earlier, are that of the average packet size and latency. These results would increase should the packets have to traverse more networks. But in these scenarios they are but a few bytes and microseconds respectively.

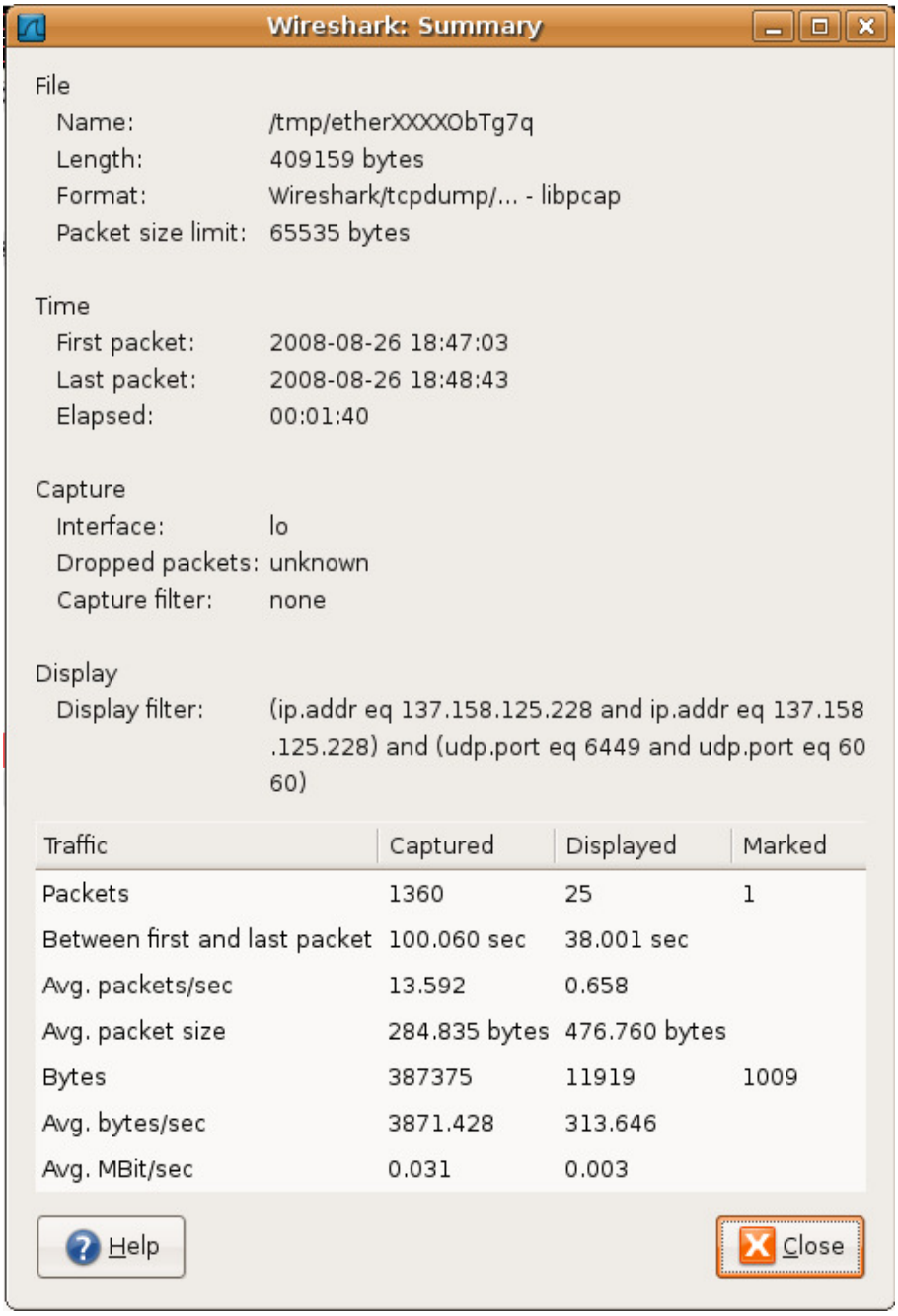

Figure 29: Summary of a Gaming Service

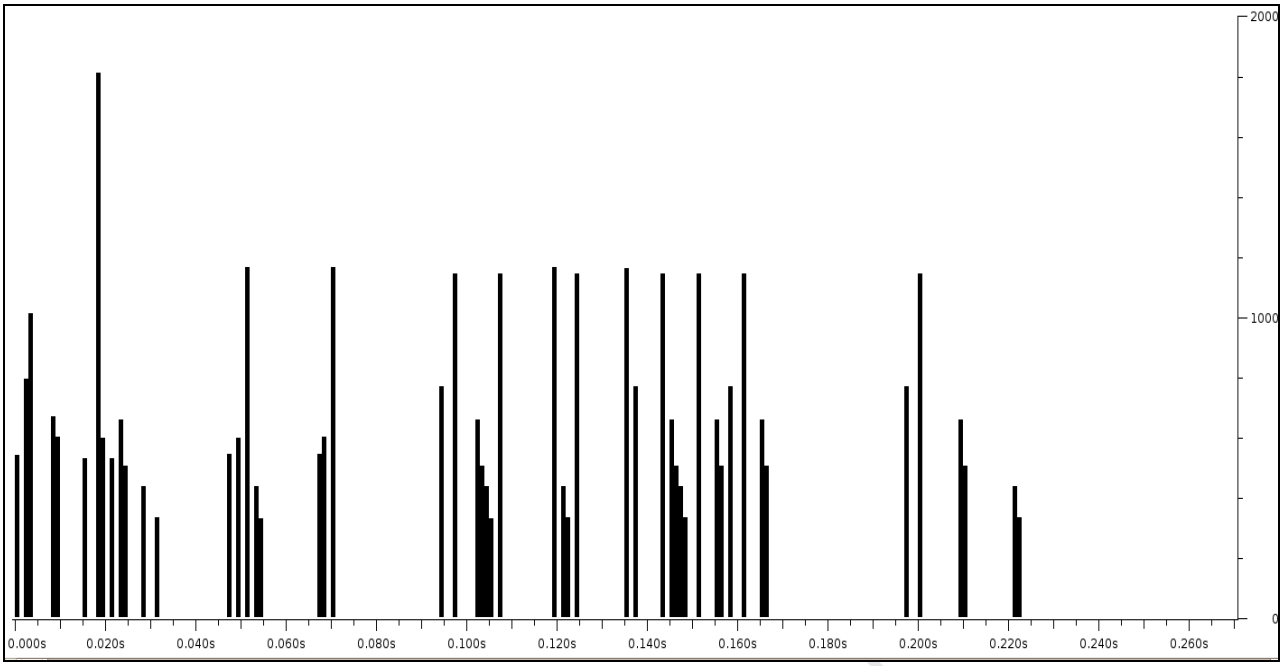

Figure 30: Gaming Session Establishment: X-axis: seconds | Y-axis: Bytes/ms

Figure 30 illustrates the signaling done during the gaming session establishment. The messages contain the subscription and the first set of cards or card indexes sent by the server. The graph also represents the response messages received. As can be seen in the figure the messages are only a few kilobytes and the time delay is relatively short.

5.2 Game Play Performance

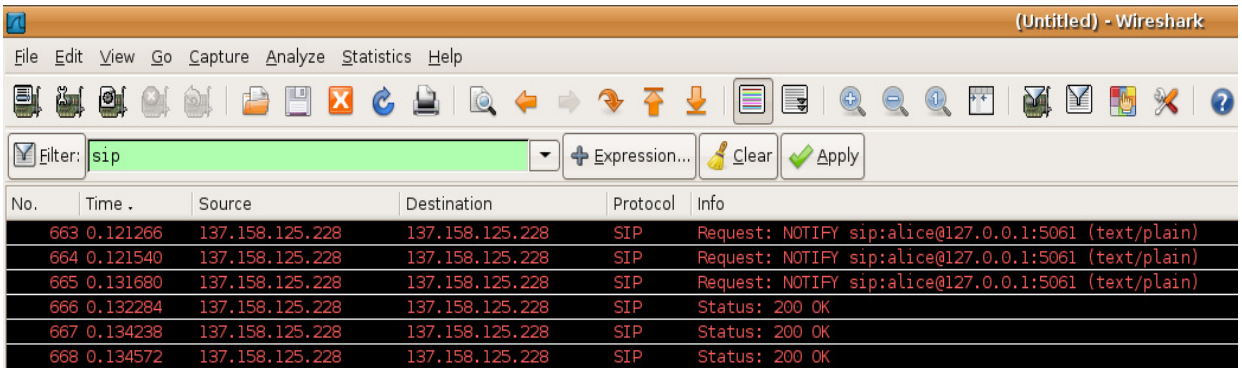

Figure 31:Wireshark depiction of NOFITY request and response

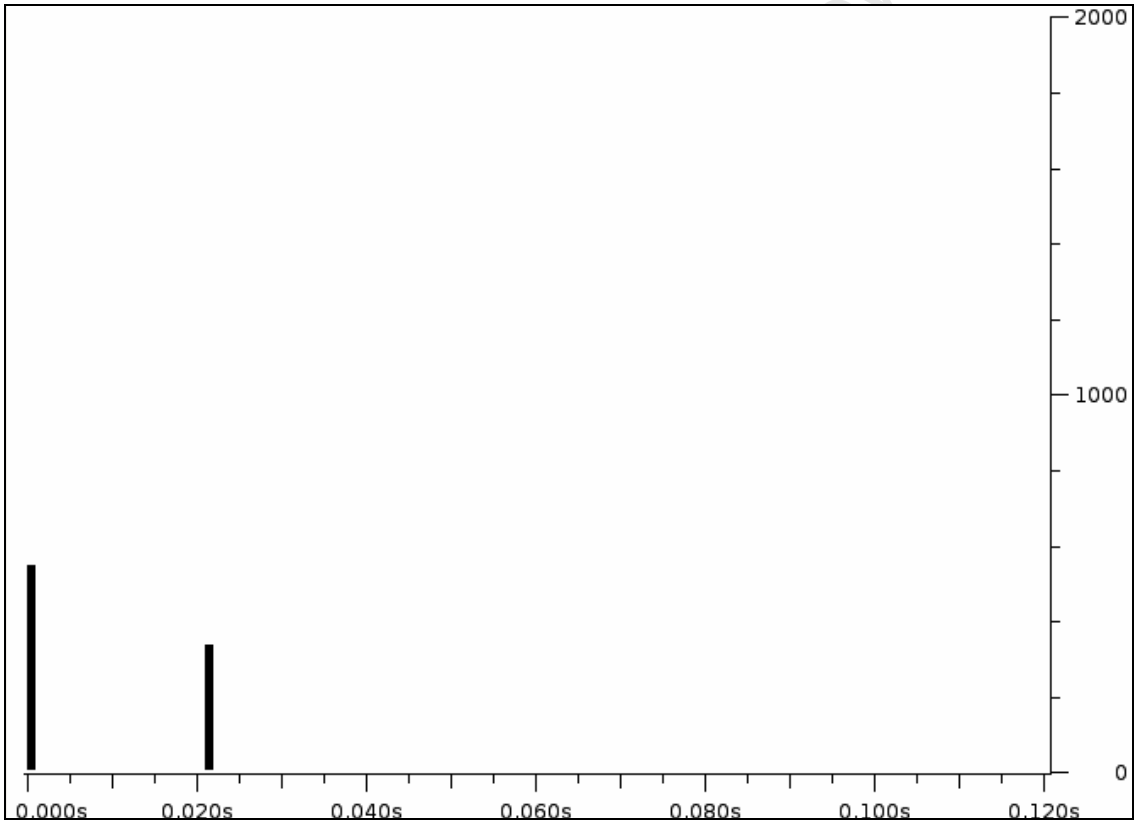

Figure 32: The NOTIFY Round Trip Time

Figure 31 and 32 illustrates the average response time of the SIP NOTIFY used during the game play. Figure 31 indicates how the messages are transferred through the architecture implemented. The following is a description of the sequence of events taking place in figure 31:

- Message No. 663 is a message sent from the Game Server to the SCSCF. Within the message body is the card index to be used by the client.
- Message No. 664 is between the SCSCF and the PCSCF. ICSCF is by passed as at this stage the registration process had been complete.
- Message No. 665 is between the PCSCF and the client.
- Messages No. 666, 667 and 668 are response messages (200ok), traversing back towards the source of the initial request that is the Game Server.

Figure 32 indicates the relative round trip time for such a process. In the figure it takes roughly 0.021s for the server to receive a response from its initial request. It is also important to note that the first message has a larger packet size due to the fact that it carries more information. The extra information includes more routing information as the packet is at the beginning of its journey and the data in the message body.

5.3 Game Server Performance

The only major performance issue related to the server side is the relative speed at which it receives messages and computes instructions to the client. Figure 33 illustrates the relative speed of dealing with a SUBSCRIBE message and issuing out instructions to the client. The time between the red dot and the first blue dot is the time between the first subscription message and the first NOTIFY issued by the Game server, this is represented by the blue dot. The processes that occur during this period include acknowledging the subscription request, establishing the dialog and computing the shuffle command.

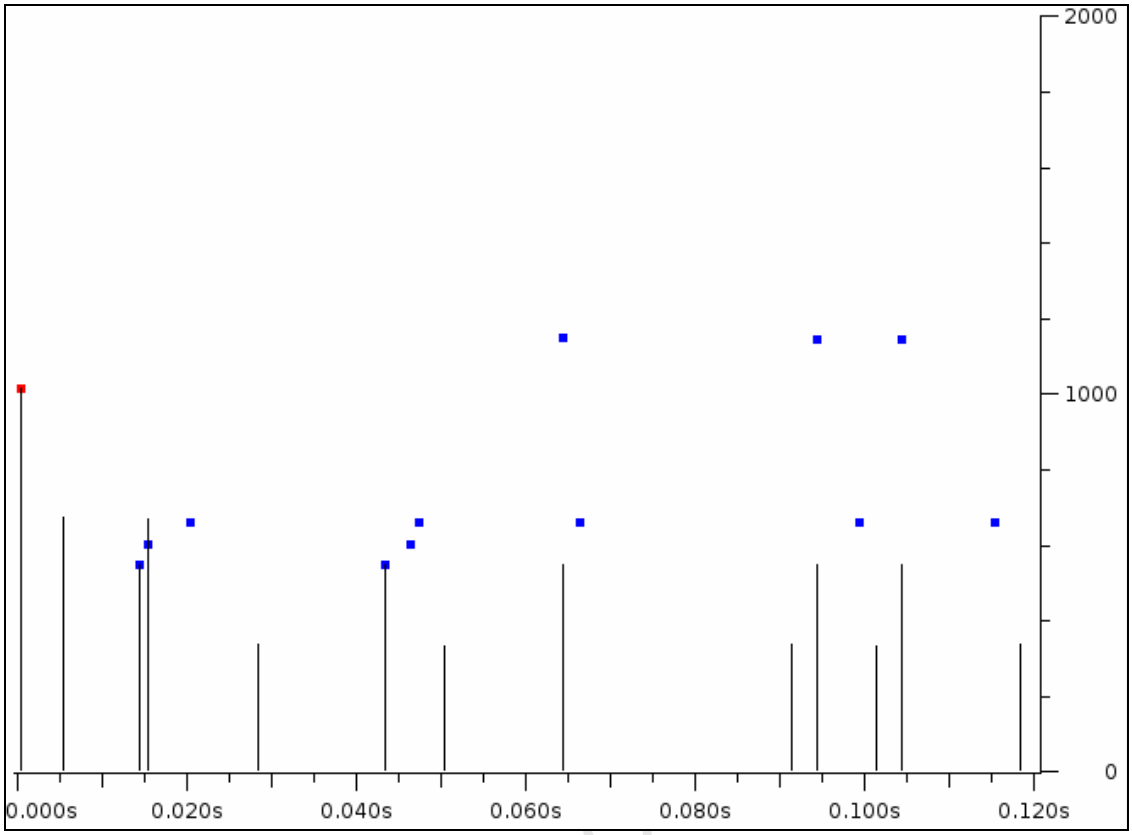

Figure 33: Dialog Establishment and game play

Chapter 6

Conclusions and Recommendations

As shown the IMS architecture is a great step forward in future telecommunications. The IMS is a service based platform and this feature is the reason for the IMS' increasing success. In order to demonstrate this feature, the research group at UCT requested that a gaming application become an addition to the UCT IMS Client.

The gaming service platform would increase wide range of services currently in use. It would also prove to form a base for services related to the gaming application. This chapter discusses the various conclusions reached during the creation of the interactive gaming application for the UCT IMS Network as well as the recommendations for future work on the IMS architecture.

6.1 Conclusions

In the process of expanding the UCT IMS research field the gaming application service was created. It was created to provide yet another platform in the IMS service development for the IMS research division. The gaming application which makes use of a wide range of standard IMS features hopes to create a basis onto which more service development may occur. By demonstrating its ease of development and implementation it hopes to bring about a wave of movement towards the IMS as a means of implementing future service based telecommunications.

The IMS platform offers a wide range of service capabilities. By manipulating the basic SIP messages, basic services can be readily created. The research conducted shows that the IMS can provide numerous benefits to operators and vendors alike. Each can readily implement these services by utilizing all the various IMS software packages available. While the IMS is still relatively new various companies have already begun the process of integrating its features into our modern telecommunications.

The gaming application created was based on the card game blackjack. The game was chosen due to its ease of implementation and its ability to be readily adapted to a majority of networked user terminal devices. The devices would need only to contain colour displays and input devices an example of this would be a 3G mobile device. Currently the UCT IMS client has yet to be fully implemented in a mobile domain. More specifically the hand held devices domain. But once this has been done the gaming service should prove to be a great asset. Due to time constraints only basic features were implemented to the game play of the gaming application. Although room was left for future service providers and operators alike to readily apply more features without drastically changing the fundamental structure of the game play.

The gaming application was implemented on the UCT IMS test bed due to the test bed housing on a broadband local area network the tests conducted were not true representations of the real world scenarios where factors such as bandwidth, latency, packet error and losses become more applicable. Baring this in mind the testing still proved to be significant. It demonstrated that a client could run multiple independent service sessions via the IMS without drastically reducing performance.

6.2 Recommendations and Future work

As the IMS is still relatively new there are numerous tasks that could be explored in the area of gaming services and other entertainment based services. The following is a list of areas that could be studied or implemented in improving the overall gaming experience in the

- Implementation of data base to which clients may store records of their game play.
- More games should be added to the server to allow the clients more variety in during a gaming service
- Once a robust client has been developed for mobile devices the gaming application should be installed on to them and thoroughly tested.
- Future gaming could also try to incorporate other IMS services within their game play. An example could be a game in which players usually view their opponents' physical features, to conclude on what course of action to take, a video conferencing session could be linked up simultaneously with the game session.

References

[1] 3rd Generation Partnership Project (3GPP), "*IP Multimedia Subsystem (IMS),"* TS 23.228.

[2] Massively Multiplayer Online Gaming; http://www.mmogchart.com: Accessed Feb 2008

[3] G. Camarillo and M.A Garcia-Martin: *The 3G IP Multimedia Subsystem (IMS) Merging the internet and the cellular worlds* 2nd Ed: Wiley 2006

[4] www.wimax.com/eduacation/wimax/ims: Accessed Apr 2008

[5] www.hauwei.com/publications: Accessed May 2007

[6] M. Poikselka, G. Mayer, H. Khartabil and A. Niemi: *The IMS IP Multimedia Concepts and Services in the Mobile Domain.* Wiley 2006

[7] Wikipedia: "IP Multimedia Subsystem" [Online] http:www.en.wikipedia.org//wiki/IP_Multimedia_Subsystem: Accessed Dec 2007

[8] S. Chakraborty, T. Frankkila, J. Peisa, P. Synnergren. *IMS Multimedia Telephony over Cellular Systems*: Wiley 2007

[9] http://www.juniper.net/techpubs/software/management/src/ src10x/sw-sdx-solutions/html/ims-integration4.html: Accessed Apr 2008

[10] University of Cape Town *IP Multimedia Subsystem Workshop*, D. Waiting and R. Good 2007

[11] IETF, "*SIP: Session Initiation Protocol*," RFC 3261

[12] J.F. Kurose, K. W. Ross, *Computer Networking Top-down Approach Featuring the Internet 3rd Ed*: Pearson Addison-Wesley 2005

[13] http://wirelesscafe.wordpress.com/2008/03/29/what-is-sip-session-initiation-protocol/ Accessed Apr 2008

[14] http://en.wikipedia.org/wiki/Session_Initiation_Protocol: Accessed Apr 2008

[15] http://www.antisip.com/documentation/osip2/ Accessed May 2008

[16] http://sip.antisip.com/docu/eXosip2/index.html Accessed May 2008

[17] Calhoun et al, "Diameter Base Protocol", RFC 3588

[18] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, July 2003. RFC 3550

[19] Kent, S. and R. Atkinson, Security Architecture for the Internet Protocol, November 1998. RFC 2401

[20] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and P. Kocher, The TLS Protocol Version 1.0, January 1999. RFC 2246

[21] E. Adams, A. Rollings,: *Fundamentals of Game Design*, 2006 Prentice Hall.

[22] A. Akkawi, S. Schaller, O. Wellnitz, L. Wolf: "*Networked Mobile Gaming for 3G-Networks"*: ICEC 2004

[23] A. Akkawi, S. Schaller, O. Wellnitz, L. Wolf: *"A Mobile Gaming Platform for the IMS":* NetGames 04, Month 1-2, 2004

[24] D. Wisniewski, D, Morton, International Game Developers Association "2005 Mobile Games White Paper" IGDA Online Games SIG

[25] A. Jarett, J. Estanislao, International Online Game Developers Association "*IGDA Online Games White Paper*" 2003.

[26] M. Balakrishnan, M. Sadasivan, *White Paper Mobile Interactive Gaming Interworking in IMS,* Nov 2007 *www.infosys.com/engineering-services/product-engineering/**white-papers**/default.asp - 29k* Accessed 3 Nov 2008

[27] Fraunhofer FOKUS, "Open Source IMS Core." http://openimscore.org/.

[28] T. Magedanz, D. Witaszek, K. Knuettel , "*The IMS playground @ FOKUS – an open testbed for next generation network multimedia services",* Proc. IEEE Conf TridentCom, 2005.

[29] University of Cape Town, "UCT IMS Client."http://uctimsclient.berlios.de/.

[30] Extending IMS QoS Provisioning to the Access Network, R. Good 2007.

[31] *White Paper Siemens IP Multimedia Subsystem (IMS),* Siemens Communications, Mobile Networks, St.-Martin-Str. 76, D-81541 Munich Siemens AG 2006

[32] http://www.gamblingcommission.gov.uk

[33] Nokia Siemens, "IMS/SIP Frequently Asked Questions," Siemens AG, pp.1-7, 2005

Appendix

Application Programming Interface

Game Server

The game server was written in the C programming language the following is the set of functions and methods that make up its execution.

Add_Game_Participant(eXosip_event_t *je)

This method takes in a subscription request and adds the client to a list of participants. It then calls on various other functions to initialise the game play. These are the shuffle(), and the Send_Game_Notify().

End_Game_Session(eXosip_event_t *je)

The method takes in a subscription end request and removes a client from the list of players.

Send_Game_Notify()

This method generates a notify request and also contains instructions on what cards need to be sent with the request based on the game being played.

Shuffle()

This function shuffles the card indexes and places them in an array of 52 integers.

Publish_received(eXosip_event_t *je)

Receives publish messages from the client and forward the message to other functions based on the choices made before the publish was sent by the client.

Get_exosip_events()

This functions retrieves eXosip events from the port number it listens too and thereafter forwards them to the relevant functions

Main()

Initialises the game server and kills it if the server if the command is given.

Gaming Client

NewGame()

Creates a new game window and sets all the variables back to their initial state.

Lookupsimages()

This function loads up the bitmaps of all the cards upon the start of a game.

Playerfunc()

This function deals with what happens when the user clicks the player button. The decisions made in this function are dependent on the state of the game at the time.

Dealerfunc()

This function deals with what happens when the user clicks the dealer button. The decisions made in this function are dependent on the state of the game at the time.

GameQuit()

This function handles the closing of a game session when the player clicks the Game Exit button.

Winner()

This determines the winner of the game based on the game being played and the various permutations applicable to the game.

Game_subscribe()

This function sends the subscription message to the Game Server. The subscribe also has details about the type of game the user would like to play.

Game_Subscribe_received(eXosip_event_t *je)

This establishes the subscription dialog when the subscription has been acknowledged by the Game Server.

Game_process_notify(eXosip_event_t *je)

This handles all the notify messages from the Gaming Server.

Game_send_publish()

This handles publish messages sent to the Game server from the client.