Chithra Prabha Peachi Muthu

Login and Networking Services for Online Multiplayer Games

Helsinki Metropolia University of Applied Sciences

Master's Degree in Engineering

Information Technology

Master's Thesis in Information Technology

14 October 2015

Helsinki Metropolia University of Applied Sciences

PREFACE

The research started in 2012, exploring networking features for an online multiplayer game. Initially an understanding of the requirements of the login services required to involve a higher number of users online was explored. Multiplayer Game development using unity tool was experimented. Login and Networking services for a unity game were implemented. Also, various networking features supported in a unity game were studied. The possibility of using login and networking services supported in an IP Multimedia Subsystem (IMS) for online games was studied. IMS is one of the established frameworks available for communication and messaging applications for millions of users. A study was carried out to analyse the usage of authentication methods using the SIP protocol for games.

The main objective of the research was to study the networking needs in the multiplayer game, implement the secure login services, and check for the possibilities of using the IMS Platform Architecture with the SIP Protocol for login services in multiplayer games. In the study the Authentication and Messaging methods in SIP were compared with the login and networking services supported by the Azure Platform and Photon Gaming Platform, as these platforms are very actively being used in the multiplayer games developed using the Unity tool.

The research gives details of ways to implement login services and networking features using the unity game development tool, and compares it with the usage of the SIP protocol for login services. The study also indicated that some of the features in the IMS framework could provide better solutions for networking and communication in games, when compared to other protocols used in the game understudy. This would help the game team to look for a different perspective for authentication and networking features in future

This research would have not been possible without many individuals, whom I wish to give my sincere thanks, and to mention the most important ones, my mother who travelled to Finland from India when I needed her help to go for studies, my supervisor Ville Jääskeläinen and Vulpine Games.

Espoo, October 14, 2015
Chithra Prabha

Author(s) Title Number of Pages Date	Chithra Prabha Peachi Muthu Login and Networking Services for Online Multiplayer Games 62 pages + 5 Appendices 14 October 2015
Degree	Master's Degree in Engineering
Degree Programme	Information Technology
Specialisation option	
Instructor	Ville Jääskeläinen, Principal Lecturer

The aim of the thesis was to study login and networking services for online multiplayer games using the unity tool. In the thesis, the authentication, communication and networking protocols supported in the unity game development tool were studied and experimented. And an analysis of the possibility to use the SIP protocol for the login services was done. A game development team from a game company, Vulpine Games Oy, was Identified with the help of a lecturer in Metropolia. With the team the game design and architecture of the game development team was studied. The SIP protocol which provides the authentication and chat features was studied and compared with the authentication and messaging capabilities in the game understudy. The game understudy was designed to use the Azure Windows platform, with a photon server for networking, with authentication using the login of users with Facebook accounts. Hence in the research work with the game development team, the login with Facebook token and secure storage of the token was prototyped and studied. In the research many aspects of game design and development were studied, mainly the game development for multi platforms with the Unity 3D development tool. In the research the secure storage of the game data and the login credentials were prototyped.

In the research the advantage of various login and networking methods in Azure Platform, Photon Platform and the SIP protocol of IMS framework were experimented and compared. The results of the study include the implementations of login services for the game understudy and suggestions on the benefits of using IMS for multiplayer games. The discussion with the game team understudy led to better understanding of the architecture of the game, advantage and disadvantage of using the different methods of login services and in future how the game could be benefit from using IMS as well, along with other platforms, and be able to involve more users in their game. The outcome of the thesis also includes prototyping of functionalities such as authentication and secure data storage, SIP server and client setup.

Keywords	IMS ,Photon, SIP, Unity , Multiplayer Games

Table of Contents

1 Introduction 1

2 Research Approach 3

3 Technology Background 6

3.1 History and Basics of Game Design and Development 6
3.2 Multiplayer Game Architecture and Components 8
3.4 Web Sockets 16
3.5 Azure Platform 18
3.6 Photon Platform 22
3.6.1 Photon Server SDK 24
3.6.2 Photon Azure SDK (PASK) 24
3.6.3 Photon Unity Networking (PUN) 27
3.7 Unity Game Development Tool 27
3.8 IMS Architecture 28
3.9 SIP Protocol 32
3.9.1 SIP Components 36
3.10 IMS Gaming Platform 39

4 Implementations 43

4.1 SIP Setup 43
4.1.1 SIP Server and Database Setup 44
4.1.2 SIP Client SetUp 47
4.1.3 SIP Protocol in Games 47
4.2 Unity Game Development 47
4.2.1 Secure Storage of Game Data 48
4.2.2 Secure Login Using Facebook with Unity 50
4.2.3 Azure Mobile Services 53
4.3 Summary 56

5 Further Development 58

REFERENCES

APENDICES

Appendix 1. Instructions for SIP Setup in simple methods.

Appendix 2. SIP Client SetUp.

Appendix 3. Steps for Facebook SDK integration to Unity Game.

Appendix 4. Creating Azure Mobile Service

Appendix 5: Steps to try the Azure plugin (BitRave) sample with Unity

LIST OF FIGURES

Figure 1 Research Execution Flowchart 3
Figure 2 Internet Application Usage statistic graph in Mobile Phones [1] 7
Figure 3 Multiplayer Game Component [4] 9
Figure 4 Game Client Component and Connections [4] 10
Figure 5 Functional Description Components of a Sample Game [5] 11
Figure 6 Game Server Components [4] 12
Figure 7 Communication Framework using WebSocket [7] 17
Figure 8 Azure Authorization components [8] 19
Figure 9 Azure Access Control Services for Games [9] 21
Figure 10 Access Control Service for Azure Platform Services [10] 22
Figure 11 Photon High Level Architecture [11] 24
Figure 12 OSI Layer Model for IMS [14] 30
Figure 13 IMS Framework Layers Model [15] 31
Figure 14 Function View of IMS Framework [16] 32
Figure 15 Internet Multimedia Protocol Stack [18] 36
Figure 16 SIP Trapezoid [19] 37
Figure 17 Overview of Model Gaming Platform using IMS SIP protocol [21] 40
Figure 18 Model Session Signals in a Game using SIP protocol [21] 41
Figure 19 SIP Protocol Setup 43
Figure 20 Game Data Storage Components 50
Figure 21 Login Screen 51
Figure 22 Azure Mobile Services Authentication Overview [27] 53
Figure 23 Overview of Usage of Azure Mobile Service for Unity games 55

ABREVATION

ASR Advanced Speech Recognition

CSP Cryptographic Service Provider

DB Database

DES Data Encryption Standard

DTMF Dual-tone multi-frequency signalling

GPL General Public Licence

GSMA Group Special Mobile Association

HTTP Hyper Text Transfer Protocol

IETF Internet Engineering Taskforce

IMS Internet Multimedia Subsystem

IP Internet Protocol

ISDN Integrated Services Digital Network

ISUP ISDN User Part

Mbone Multicast backbone

NAT Network Address Translation

PSTN Public Switched Telephone Network

RFC Request for Comments

RUDP Reliable User Datagram Protocol

SCTP Stream Control Transmission Protocol

SDP Session Description Protocol

SDK Software Development Kit

SIP Session Initiation Protocol

TCP Transmission Control protocol

TLS Transport Layer Security**.**

TTS Text-to-Speech

UA User Agent

UAC User Agent Client

UAS User Agent Server

UDP User Datagram protocol

URI Uniform Resource Identifier

VoIP Voice over Internet Protocol

XCAP XML Configuration Access Protocol

XML Extensible Markup Language

1 Introduction

Internet has changed the ways how communication between communities takes place. With new technologies and frameworks over time, with the possibility to share the media over the internet, people stay connected to one another in different geographical locations very easily. Communication with the help of media has enabled people to get fascinated towards globalization, as it helps them to stay in touch with their communities, even when moving far away from their communities and families for work and other purposes. Also media communication has enabled people to find the virtual communities to interact with for the purpose of entertainment. Virtual communities are very active in the game industry. Game industry has found a huge change with the introduction of online gaming. Earlier the game industry was used to play video games on a computer or a machine, but now people play games online communicating with many people playing the same game. There are several new technologies and protocols which enable communication in games online, which changed the game industry.

Unity is a well-known game engine developed by Unity Technologies. Unity game development tool provides a good set of tools and software to develop a multiplayer game, it provides very good integration with many services platforms such as Azure platform and Photon platform. Unity also provides its own networking components for features like messaging. Most of the networking features in unity and its supported services platforms are implemented using websockets.

IP Multimedia Subsystem (IMS) framework is an architecture which provides services to deliver the Internet multimedia packets between networks easily. IMS framework is supported by various protocols to support the media communication over the internet.
IMS framework is used for various purposes like messaging, audio and video calls, file sharing and media streaming.

The aim of this thesis was to explore the current trends in the online multiplayer game development, study and find out the capabilities of the currently used login service protocols that are being used in the game development and compare them with the SIP protocol. The research Question is “How the scalable login service could be implemented

using the unity tool for an online multiplayer game? Can the IMS Framework's SIP protocol be best utilized for user authentication and networking in the game applications built using the unity tool?"

The outcome of the study includes an implementation of some of the features that are required by the game team understudy, and a SIP server and client setup to experiment authentication using SIP protocol. The login functionality of existing social networking identity providers e.g. Facebook and inviting friends, was mainly studied and prototyped. The discussions during the research were helpful in understanding the architecture of the game, better implementation and design of the game, and gave an insight into benefits of using the IMS platform in combination with the other existing gaming platforms in future.

The thesis report is structured with details of the research approach in section 2, in which the steps followed in the research are explained in detail. In section 3, the details of the background study about the technologies involved in the game understudy and IMS framework are documented. In section 4, the SIP server and client setup, the game features, implementation details of the game understudy, details of the Azure mobile service for authentication and other findings from the study are listed. At last in section 5, the details of the best ways to implement the authentication methods to securely login and network using the Azure platform, Photon Platform and the possibilities to use IMS platform with unity tool in future are discussed.

2 Research Approach

The study was started by creating a flow chart for the research, with the list of tasks to be done for experimenting and implementing the Login Services using various methods, such as Login using Facebook Authentication in the mobile device, networking using Photon Server in Azure platform for the game understudy and ways of using the SIP protocol setup for authentication (cf. Figure 1).The flowchart has steps for studying the architecture of a multiplayer game in the game team understudy, for studying the SIP protocol, implementation of some functionalities for the game understudy, then finally a discussion on comparison of various authentication methods using the photon and azure platforms and possibility to use SIP protocol for authentication.

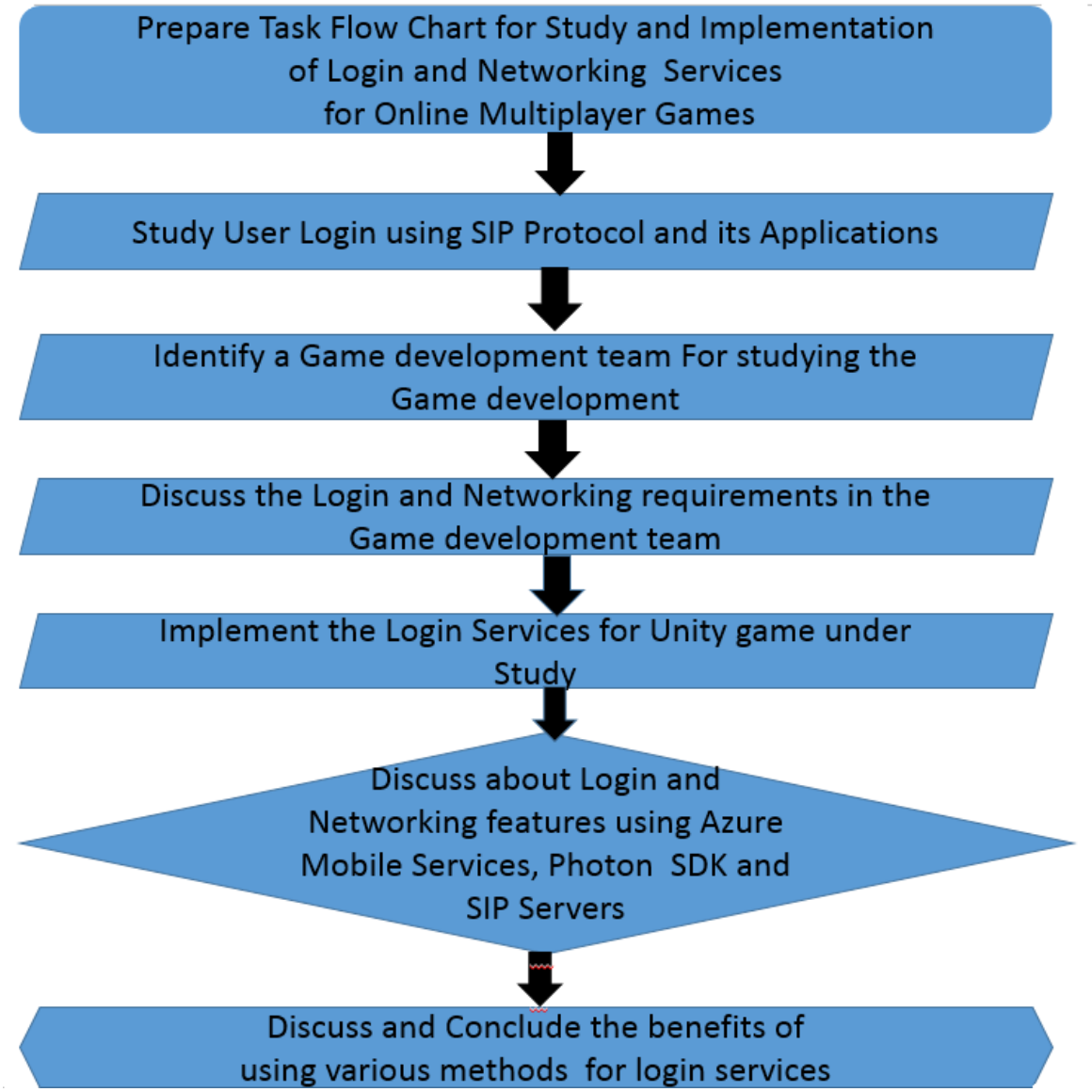


Figure 1 Research Execution Flowchart

As the first step a login using SIP protocol was experimented, in which the SIP server and clients were setup, a set of test users were created in the server database and using the created user credentials and logins the communication between two SIP clients were experimented and documented.

The next step was to study the game development, for this a game team was identified in Vulpine Games Oy organisation and the opportunity to work with the game team was explored. With the game team a discussion on how to get involved in their game development was done. As a result a plan for future meetings to discuss the further development needs of the game was done.

Next the requirements of the game understudy were gathered in discussion, details such as the concept of the game, tools and software used to develop the game and list of features that is required in the game were discussed. The main features that were required include technical design and implementation of a single shared game world for millions of players, security in massively multiplayer mobile games and scalable login service. Of these features the login service was initially required to be implemented, hence agreed to work on the scalable login service, and implementation of scalable login using the Facebook SDK was discussed.

Next prototyping the Facebook login and inviting friends feature was implemented, for this the Unity game development environment was setup and Facebook SDK authentication methods were experimented. Next, prototyping of the secure storage of the login credentials of the user was prototyped.

After that, further discussion to understand the networking needs of the game understudy was done. Features such as messaging and group communication were key needs for a multiplayer game communication and networking. The networking features for a game developed in unity could be implemented using unity networking APIs, the Photon SDK and the Azure platform's services, all these methods were studied. The practical implications in using the Azure platform and Photon SDK for features such as authentication and game data storage at the server side were discussed, then a study of a possibility to use authentication using the SIP protocol was discussed.

Finally the comparison of using the different login service methods of the Azure Platform, Photon Platform and SIP Protocol was done and the usage of various authentication methods of Photon Server on the Azure Virtual machine was detailed to the game team understudy.

In the research the data collection was done from data available in Internet in the form of articles and books. The researcher also used the knowledge and previous experience gained while working in the development of a SIP client *Nokia Communication Suite" which is a desktop client for communicating with the mobile clients, to discuss and compare the IMS capabilities with the other gaming protocols. Current trends in the development industry were studied with the Vulpine games Team in several discussions and face to face meetings with the CTO of the Vulpine Games Oy organisation.

3 Technology Background

In this section the details of the tools and software studied and used in the research are described. In the beginning the history and background of the game design development are described, next the key components of the multiplayer game are detailed, and then the details of the web socket protocol which is widely used in online games is described, next the Azure platform and Photon Platform components are described, then the IMS framework and SIP Protocol details are listed.

3.1 History and Basics of Game Design and Development

In the initial stage game industry had Arcades games. With latest inventions in the multimedia technology, the video games became more predominant among the game players, and with the invention of online games in 1970s, the game application had started to dominate among the internet users, with the deep penetration of the intrusive technologies into people's life and usage of mobile phones, the games are now the major players in the market shares of internet applications. As indicated in Figure 2 below, the game users contribute to high percentage of users of the internet applications [1]. Some Games are also integrated to social applications, to involve more communities to play games. Technologies used for games have also grown tremendously to address these needs.

Game development needs have led to the introduction of many tools and platforms making it easy to create and sell the games. Some of the key platforms that are currently providing services for games are the Azure Platform, Photon from Exit Games and Google Cloud platform. Also there are a few other platforms such as Nexus which are build using IMS framework, but those are not well recognised and used widely. This thesis is to mainly find out the best ways to use the available authentication methods to involve huge number of users in the game understudy, and find out ways to provide scalable login services using the social networks and also provide very good networking features for games.

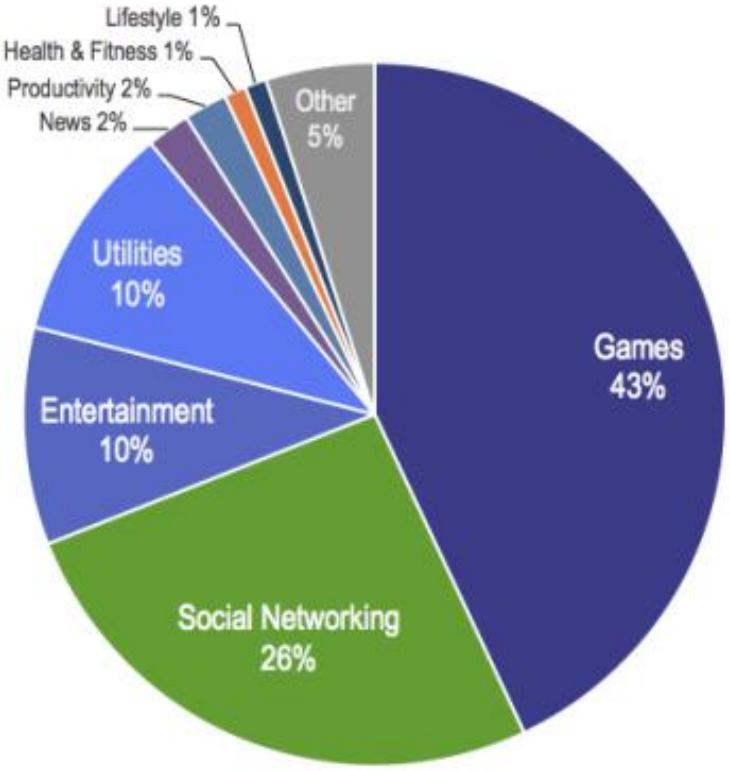


Figure 2 Internet Application Usage statistic graph in Mobile Phones [1]

To build a game is to create an imaginary world with characters that are fully computerized. This includes a lot of illustrations of real world situations, interactions, events and characters in digital form. The interactions between the objects and characters in the game would make the game player involved with an imaginary world and entertain or educate the game users. The objects and character interactions can be made using the controller of the objects, so the characters in the game are operated by controllers, the controllers could be a game program or another game player who is providing the instruction and act as the controller of the character, with the input devices like joystick, mouse and keys.

Game design is an art of creating characters and objects for the digital world. Game development starts with an idea to represent interesting events and happenings. The actual game design is done next. Game designing is done with the help of a lot of the maths and physics, game designers build and transform the computerized graphical objects and characters in an imaginary world in computerized space. With invention of latest graphics technologies and software, the game designing is made very easy, as there

are many tools available now, the tools provide readymade game objects and components to create games very fast and easily, also with a lot of support for game design and development.

Games are designed using the game theories which are well-defined mathematical objects such as trees and matrices [2]. A games need to have some basic elements specified, those elements are the players, the information and actions of each players at every decision point, and the payoffs of every outcome. Based on these elements of the game, a set of equilibrium strategies is designed for each player in a way that, when those strategies are employed in the game, no player can benefit from unilaterally deviating from the strategy. These strategies deployed in the game provide an equilibrium for the game, and thus the game could be played in a stable state in which the outcomes occur with a known profitability [3].

Game development tools such as Unity provide the support for creating the game arts and elements. With the help of these tools, the user can create a game world and elements, with a logic to interact between the elements and to create strategies of the outcome of the interactions. There are also game engines which provide a set of packages with designed elements and logics, by which the users can just design the art for their appearance and create new games.

3.2 Multiplayer Game Architecture and Components

With the growth of mobile technologies and internet facilities, the game elements can be distributed over internet and can involve more players in different location globally to have control of the game elements and play remotely. The multiplayer games with players located at various locations connected over internet has become the most growing trend in the game industry. In multi-player games, many players get involved in the action to produce an outcome, the strategies for outcome and their results can be based on the outcome they create together and individually based on the game logic. The connection and communication between the players are done using a set of servers and services. The picture below describes the basic blocks that make a multiplayer game based on flash technology which was very popular before a few years [4].The core concepts and components of a game is the same over years, hence reference [4] are studied and described in this session. Figure 3 illustrates a multiplayer game component.

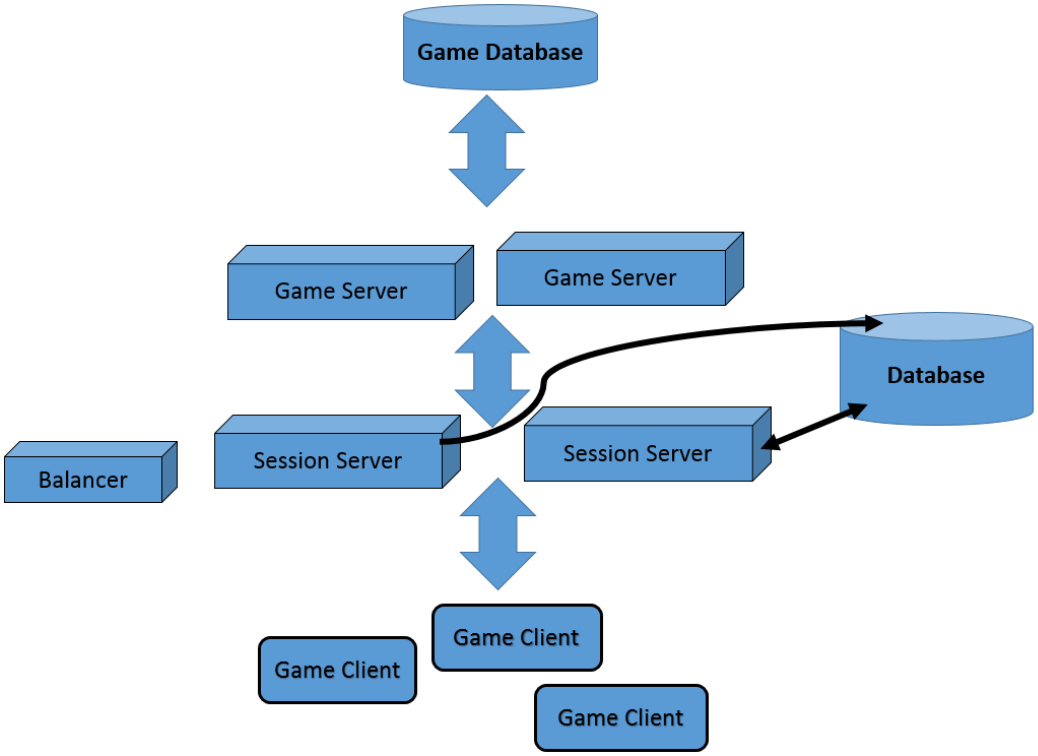


Figure 3 Multiplayer Game Component [4]

In multiplayer game all the data, such as the game points, that are needed to play the game and shared between users, is stored in the Game Database. The game server performs the tasks that are needed to store and retrieve the data from the database, it is also used for implementing the event logics for the game. There could be one of more data servers based on the game size, in the games that are played globally, there could be one game server for each geographical area. There can be many users playing together and as a result there can be several session initiated for each user, each session is maintained by a session server, and this server connects the user with the game server. When there are too many session at a point of time, the Balancer programs work to balance the load, the balancers work is to ensure that users session is maintained in a balanced way, that the users do not feel any latency when playing the game and there are not too many connection to the game .

Each session that the user creates when the game is played is created with the secure authentication methods, so that the users are playing the game with a secure connection between them. Each user creates and maintains the login for playing the game, and the login details are stored in the Authentication Database, so that those details could be used for communicating in secure way with the game server. A user plays the game using the Game client.

Game clients have components that are needed to get the inputs from the players for the game, game inputs like key board inputs, mouse movements and media inputs such as record the audio and video inputs, Game clients have User Interface Inputs and Display Components, to help the user view and interact with the game elements. Each game has a controller program which interacts with all the needed inputs for the game and processes the inputs according to the game logging and creates and outcome and then communicates it to the other components, such as game server and authentication server. The overview of typical game components in a multiplayer game is shown in Figure 4 below.

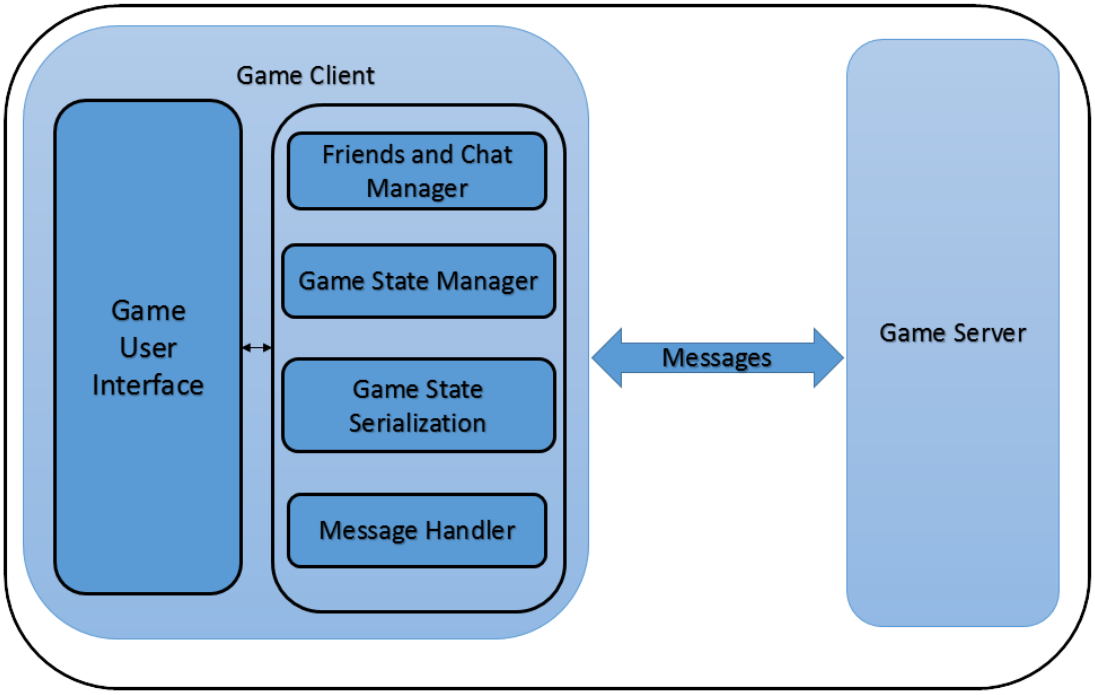


Figure 4 Game Client Component and Connections [4]

Below are the functional description components used in a sample 3D game, in the game the head movement is done based on the voice input, the game uses the Qt technology for user interface and Open GL Technology for graphics rendering and other speech

recognition technologies. Speech recognition is done using TTS, ASR, MRCP libraries and plugins, the speech interpreted to animate the face accordingly [5].

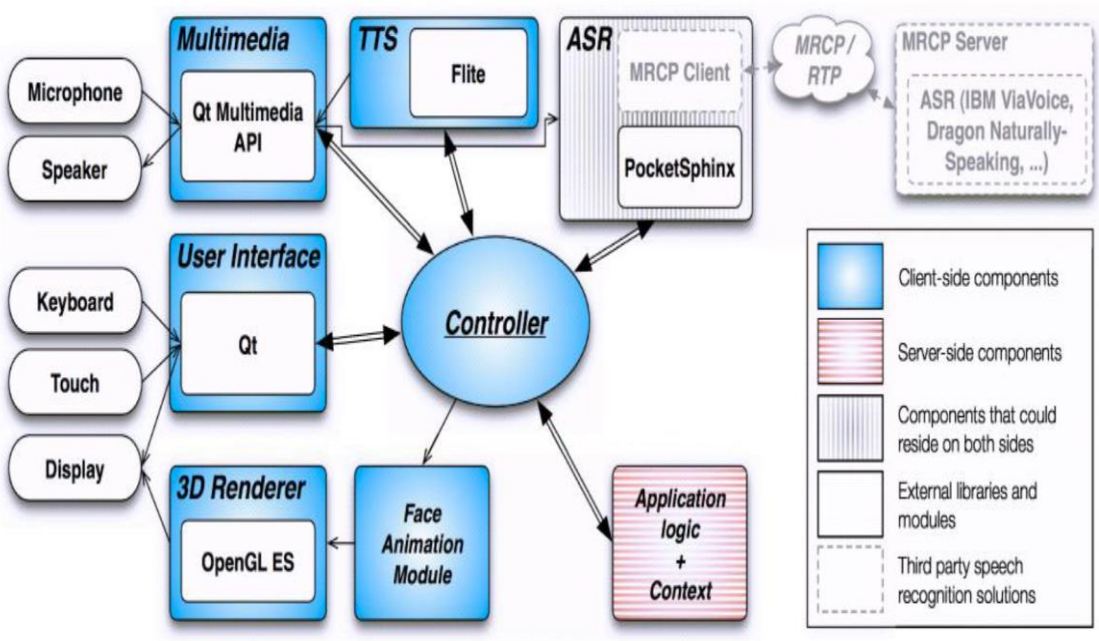


Figure 5 Functional Description Components of a Sample Game [5]

Game server components are most important for Multiplayer games, as they have a key role in involving multiple players across the network, hence the game server needs to be designed and build carefully. In most common games there are some components that need to co exists, those components are mentioned in Figure 6 [4]. The names of these components could vary in different platforms, but the basic characteristics of these components are needed for any multiplayer games and they need to exist in a game for a game to perform well.

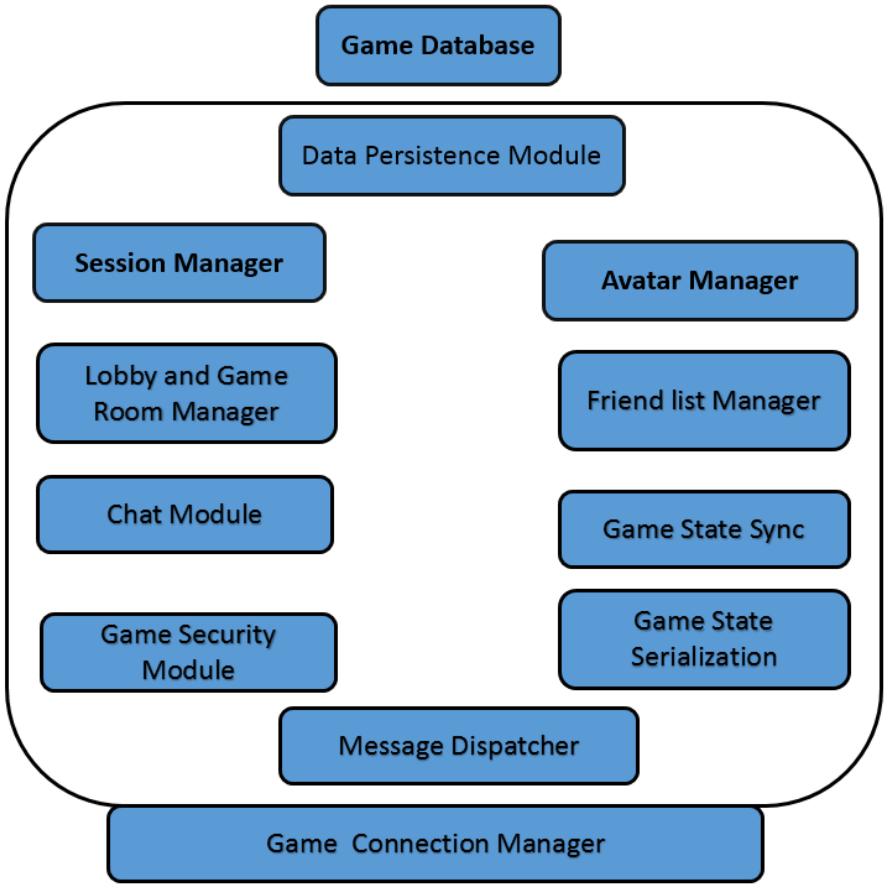


Figure 6 Game Server Components [4]

A typical game server must implement the following modules: Database Persistence, Session Manager, Avatar Manager, Lobby & Room Manager, Friends, Chat, Game State Sync Object, Game Security, Game State Serialization, Message Dispatcher and Game Client Connection Manager.

Persistence module is used in order to provide all the necessary features for the multi-player game client. The persistence module takes care of all the reads and writes to a relational database, such as MySQL. All other modules requiring reading or writing to the database will request this module to do the actual work.

Session Manager helps to maintain the sessions, a session is maintained for each player that is connected to the server. It does various bookkeeping tasks, keeps track of each session details, including things such as hours played, login time, client IP and port, etc. The session is cleaned after the player has logged out. Usually, there is one session for each client connection. The server may hold connections to several players at the same time, so one needs something called a session object to keep track of the data transfer

from different clients, so that the game does not mix up the data from other players. A session is created for every client that connects to the server. The session is where all the messages are processed. Each session holds its own temporary buffer to read the data coming in from the socket. Remember, data is received from a socket in small chunks. So until the whole message is received, the partial data received is stored in a holding buffer. Once the complete message is received, the message is then processed. The messages could be login, chat, or any game state sent by the client.

Avatar manager is responsible for keeping track of the properties specific to the player in the game. The avatar keeps track of virtual cash, hit points, life, etc. Usually, for each session, there exists only one active avatar.

Lobby contains many rooms and rooms contain many players. This module keeps track of where each player is and lets players know when a new room is created, updated or purged.

Friendship relations between avatars are persisted to DB by the Friends module. The relationships are established upon player requests, also keeping track of any pending friendship requests. A Chat module typically routes the messages from one player too many others or privately to one other player.

Game State synchronization module keeps track of all the current states of a game within a room. Any object modified by the client must be updated to all other clients within the same game instance within a room.

Game State serialization takes care of any objects needing synchronization across players that are transmitted as messages over the network. This module is responsible for converting a given object to a compact binary format, sometimes encrypting prior to sending over the network. The module also performs the complimentary function where the received binary messages are de-serialized when a message arrives at the destination over the network. A typical message format is sent between client and server. These message exchanges are the basic building blocks of any networked client-server implementation.

Security module ensures the client connections to any server it wants to connect are established in secure way. The server with which the client wants to connect to must

grant explicit permission to the client requesting it. When the client creates a connection object for example a socket ,and wants to connect to the server, security module should make requests first, then the security policy are exchanged ,when the request is received and the module sends back a policy data specifying that it is okay for the client to open a connection to the server.

Connection management and message dispatcher module is the workhouse for receiving messages from clients and routing them to different modules or services. The module is also responsible for dispatching messages generated from the server to the clients. Session object is created for the connection and adds the new session to the active list. For each session in the active list, calls read and write methods. Also detects any client termination and cleans up the corresponding session object.

3.3 Game Platforms

To develop a game which has high art content and elements and which will allow huge number of users, there are many components that need to be integrated together and working well integrated to create the complete game experience for the users. This requires more technical and design aspects e.g. good art design work, networking features, security feature, high quality graphics and rendering, and good processing of the game inputs. There are several tools and platforms available now for game design and development, a few of the most popular platforms are mentioned in Table 1 below[6],

Platform Name	Features and Cost
Adobe AIR	• Coding Language – Actionscript, Javascript HTML • Supported Platforms – iOS, Android, Blackberry, Kindle Fire, Nook Tablet, Windows, Mac • Cost – Opensource SDK
Unity	• Coding Language – Javascript, C#, Boo • Supported Platforms – iOS, Android, Blackberry, Windows Mobile, Mac, Linux, Windows, PS3, XBOX, Wii U • Cost – Initially Free,Charged based on the revenue from the game .Starts from $1500 or $75/m

Corona SDK	• Coding Language – Lua • Supported Platforms – iOS, Android, Kindle Fire, Nook • Cost – Starts from 16$/m
Cocos2D-X	• Coding Language - Applies C++, Javascript, and Lua • Supported Platforms - iOS, Android, Windows Phone, Black-Berry,Tizen etc. • Cost – Opensource and Free
Trigger.io	• Coding Language – HTML5, Javascript • Supported Platforms – iOS, Android, Blackberry, Windows Mobile, Phonegap • Cost – Starts from 79$/m
Marmalade	• Coding Language – C++(SDK), Lua(Quick), Obj-C(Juice), HTML Javascript(Hybrid) • Supported Platforms – iOS, Android, Windows Mobile, Tizen • Cost – Starts 15$/m
IntelXDK	• Coding Language – HTML5, Javascript • Supported Platforms – Android, iOS, BlackBerry, Windows Mobile, HTML5 • Cost – Free
Haxe	• Coding Language – Haxe, NME • Supported Platforms - Haxe can be compiled to all popular programming platforms with its fast compiler – JavaScript, Flash, NekoVM, PHP, C++, C# and Java – which means your apps will support all popular mobile devices, such as iOS, Android, BlackBerry and more • Cost – Free and Opensource
Sencha Touch	• Coding Language – Javascript, HTML5 • Supported Platforms – iOS, Android, Windows Mobile, HTML5 • Cost – Starts from 995$

JQueryMobile	• Coding Language – Javascript, HTML5 • Supported Platforms – iOS, Android, Windows Mobile, HTML5 • Cost – Free
Monocross	• Coding Language – C#.Net • Supported Platforms – iOS, Android, Windows 7 Mobile • Cost – Opensource

Table 1 List of Tools and Platforms for Game Development

Among the listed platforms and tools in Table 1, Unity is one of the most popular tool that is used by game developers to build their game. Also Unity offers developers to try the too for free and also publish the games and earn money with no charge up to certain limit.

3.4 Web Sockets

In 2010 web sockets were introduced, to facilitate better communication in the web browser clients, which also facilitated the games to network easily. Figure 7 gives overview of the messaging in a browser client over internet happens [7] using websockets. Services such as Application services, Data storage and Web Services could be used in the game application.

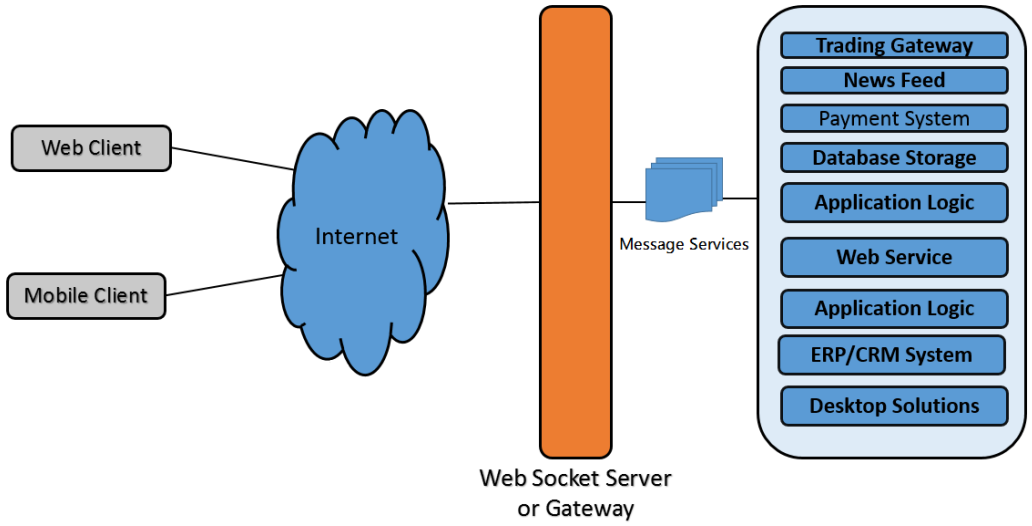


Figure 7 Communication Framework using WebSocket [7]

The WebSocket Protocol enables two-way communication between clients running untrusted code in a controlled environment to a remote host that has opted into communications from that code. The WebSocket Protocol was initially designed for communication in browsers, but it is also used for communication in server applications and clients. The WebSocket communication is started with opening handshake by framed messages over TCP.The goal of the WebSocket technology was to enable two way communication with servers in browsers, without relying on the multiple HTTP connections. TCP Port number 80 is used for communications using websocket, as this is of good benefit in environments that block non-web connection in the internet using a firewall. Websockets is unaware of the firewalls and the proxy servers, it supports the HTTP-Compatible handshake with the help of which the HTTP servers are able to share their default HTTP and HTTPS ports (80 and 443) with a WebSocket gateway or server [7].The WebSocket protocol defines a ws:// and wss:// prefix to indicate a WebSocket and a WebSocket Secure connection, respectively.

The benefits of Websockets over HTTP protocol, is that the Websocket protocol is optimized for speed, it utilizes less bandwidth unlike HTTP which uses more bandwidth for every request with various headers and frames, sent with every request. Websockets do not require many requests, as after the handshake is done, the data transfer goes un intercepted until the connection is closed. Websockets provide less latency when using

Firewalls and proxies, with better polling and streaming solutions. Websockets is the latest technology with features like minimum data used for communication, better utilisation of bandwidth, and as still it is operating in the HTTP Protocol it works with load balancers and it supports usage of binary data. Web Sockets are best suited for Games, Chat applications, Real time data transfer and News tickers.

Though Websocket is still a TCP based protocol, which is based on handshake and signalling ,but as the signalling is done only for opening the connection in a particular port, and then the data transfer happened continuously, websocket provide a very good alternative in TCP equivalent as using the UDP and more secure as well. With many benefits the websocket is being used by most of the game developers.

3.5 Azure Platform

Azure is a services and internet computing platform. Microsoft manages and supports many datacentres to provide the Azure platform services. The platform supports features that are available to be used separately and also as package and all these features are also supported with developer services as per the requirement of each features. Azure platform provides support for various features like compute, Data management, Networking, IT services, Developer Tools and services, Authentication services for Identity and access, Mobile Services and Notification hubs, Backup services, Messaging Services, Compute assistance with scheduler, Data Storage, Media and Commerce. Service to buy and sell the application, with advertising support.

Azure platform ensures the secure access control to its services components with authentication and Authorization methods. Figure 8 represents the methods of using the access token from the identity providers such as Facebook [8] .The mobile client applications use the access token provided by the identity providers such as Facebook, to access various services that are available in the Azure platform like Game Server services and Data Storage services. Using the Azure cache services speed up the authentication process, this services stores the authentication tokens in the cache, so that every time the user login or access from a client the authentication tokens are not request, instead the token from the cache is used and only if the login fails the token is requested again and stored in the cache for further authentication, this saves a lot of time for authentication.

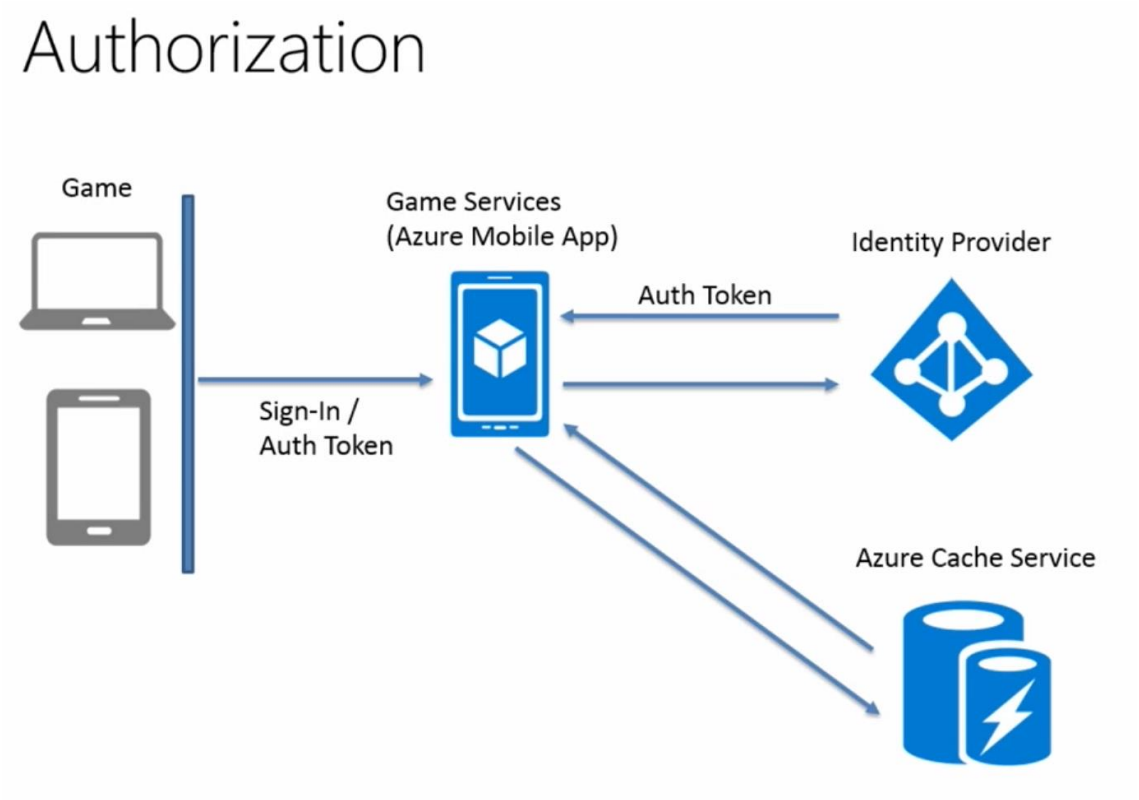


Figure 8 Azure Authorization components [8]

Azure platforms' Data storage component is one of the most important for the applications like games. Azure Storage Service manages the storage of blobs, queues, and tables, it is needed that this data be kept secure, so that there is no unauthorized access to it. Azure Data storage provides access to the data with storage account, which has an account name and an access key which are used to authenticate access to the storage service. The management of these access keys is important. The storage service provides two access keys for each storage account, so that when one key needs to regenerated, other access key can be used, and regenerating the other key is done in background, and keep the service running without delay. The storage service supports hash-based message authentication (HMAC), in which a storage operation request is hashed with the access key. On receiving the request, the storage service validates it and either accepts or denies it. The Windows Azure Storage Client library provides several classes that support various ways of creating an HMAC, and which hide the complexity of creating and using one. Blobs are ideal for storing static content for web roles in the form of files, so the storage service provides several authentication methods for access to containers and blobs. But in practice, a container can be configured to allow anonymous access to the blobs in it. Blobs in such a container can be downloaded without any authentication.

Possession of the storage account name and access key is sufficient to provide full control over the data managed by the storage account, it is needed that the access keys be kept secure. In particular, access keys should never be downloaded to a client, such as a Smartphone, as that exposes them to potential abuse. Azure makes sure that the keys are accessed securely by using primary and secondary keys, in which the secondary key is being used, when the primary key is being generated, and the primary key are usually encrypted for safety purpose.

Some Windows Azure services use a different authentication scheme, based on a service namespace and authentication token. In practice, these are similar to the account name and access key used to authenticate against the storage service, although the implementation is different. These Windows Azure services use the Windows Azure Access Control Service (ACS) to perform authentication. However, this is abstracted away in the various SDKs provided for the services.

The Azure platform in addition to the data storage features, it has support for many other functionalities. The Windows Azure Service Management REST API is a RESTful API that provides programmatic access to most of the functionality available on the Windows Azure Portal. This API uses X.509 certificates for authentication. Prior to use, the certificate must be uploaded, as a management certificate, to the Windows Azure Portal. The certificate must then be added as a certificate to each request made against the Service Management API.

Figure 9 below explains the details of the architecture of how the Azure platform meets the demand of the gaming platform's backend needs. It consists of the components for Authentication, Notification and other services, which are accessed by the game client using the Access control service Authentication methods. The services are used by the Platform as s Service or Software as Service components which are needed for building the multiplayer games. And the data storage, with the component for Analytic services for analyzing the component performance and emulating the services.

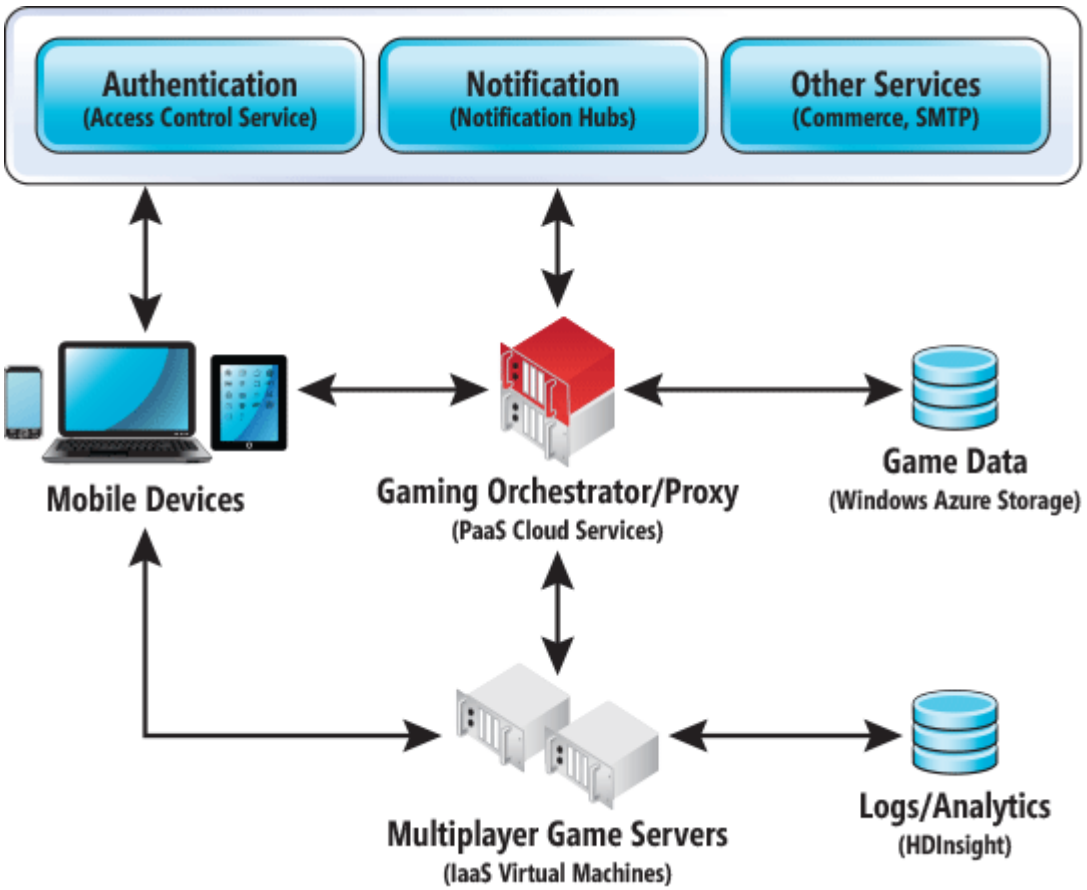


Figure 9 Azure Access Control Services for Games [9]

The Access control services provide a secure way to access the server application such as Photon servers, which are components that act as Software as a Service (Saas).this service allows the users to use their social networking accounts for authentication in games. Figure 10 gives the overview of how the ACS authentication works, it uses the authentication token from the social networks to generate the ACS token and is used for authenticating the game clients, but currently ACS is well supported for windows store applications mainly.

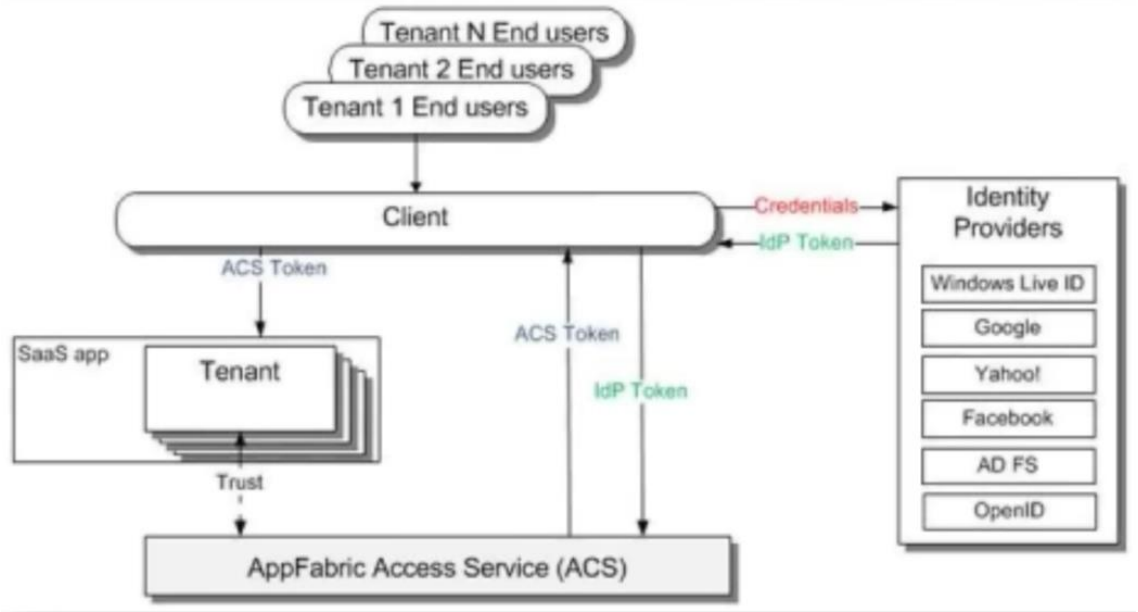


Figure 10 Access Control Service for Azure Platform Services [10]

Another authentication method Azure provides for all mobile devices is Azure Mobile Service, this is experimented in the study and more details are available in the implementation section. Azure Mobile Services SDK provides the support for using the cloud backend services for the mobile applications and games, the latest approach provided by the Azure platform for games using Azure Mobile services, supports services that could be used for games, but for that one need to design the game logic and run the server applications using the Azure mobile services, it also supports the Authentication using the Facebook networking accounts. Appendix 5 explains the steps that are needed to use the Azure mobile services database with the Unity client.

3.6 Photon Platform

Photon Framework offers a variety of services for game development, Photon is created and owned by Exitgames, a company from Germany. Photon has complete support for any time hosting and operations management for online games, with scalability to tens of thousands of concurrent users. Photon provides unique solutions for game development and it delivers affordable and easily usable middleware services. Photon provides advanced features for the development of games that let players compete against one another in real time, by providing already developed features which are ready to use set of features like, lobbies, matchmaking, buddy lists, leaderboards, and more. It supports

In-game credit for subscriptions and item sales. Photon is more reliable multiplayer game development engine for computers and mobile devices.

Photon has support for many platforms such as Windows, Mac, flash, Unity 3D, iPhone and Android. Photon is a purely C-based core engine which is quite simple, delivers the best performance while also being easily expandable using C#/.NET. Photon fits well with Unity, as it supports C# class to handle server logic for pear to pear on the client, and could be used as just plugin into the game client, thus lets the developers to abstract away from pear to pear (P2P/Server networking and their networked physics and game sync code, hence if P2P fails, player can pause the simulation, have a player transfer the world-state to the server, connect everyone to the server, and just keep on playing.

Photon offers different ways to use its services with other gaming platforms. Photon could be used as server , as networking tool integrated with the unity tool using PUN , and it also supports the using its services integrated with Azure platform.in the bellow session the details of using photon framework in various ways is described.

Photon supports the protocols UDP and RUDP TCP, HTTP and WebSockets, and is totally abstracted, Photon uses Self-certifying File System (SFS) .Photon uses Self-certifying File System (SFS), SFS provides a secure and decentralized file system that is available for global networking.SFS allows its users to access their files from any part of the world and share the files with anyone.SFS uses strong encryption and decryption methods to provide security over untrusted networks. Photon has very good capability to scale, provide support for cloud features and clustering. The Figure 11 gives the high level architectural view of the Photon [11].

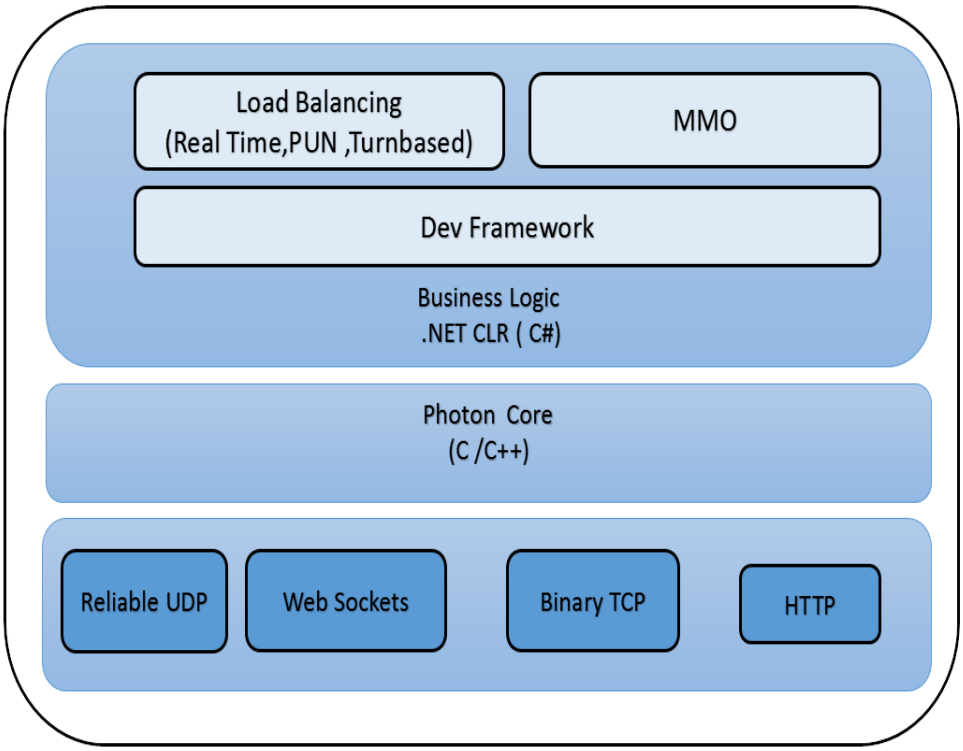


Figure 11 Photon High Level Architecture [11]

3.6.1 Photon Server SDK

Photon supports a high performance C++ server application, which takes care of the protocol part, it has good support for common features of the clients and hosts C# applications that take care of the game logic. Photon Server is server application that could be run or hosted in any machines of user's choice, this helps users to use the cloud virtual machines as server machines. Photon lets the user to customize the applications as per their needs, and lets them have complete authoritative control, with this facility the developers are able to configure and set up their own infrastructure in applications which involves multiple players. The Photon Server SDK enables users to develop their own multiplayer game backend using the Photon socket server. It Implements authoritative logic, persistency, cheat prevention or prediction in a central instance between many Photon clients. Photon has good feature applications such as Counter Publisher, Lite and Lite Lobby, MMO (Massively Multiplayer Online) and Lobby Manager.

3.6.2 Photon Azure SDK (PASK)

Azure platform offers two types of hosting methods to use the photon server features. The first method is using the Azure Cloud services, by using the Web or Worker roles,

using the PASK SDK. Second method is to use the Azure virtual machines and install the Photon Server SDK in the Virtual machines.

In the first method there are objects that act in certain roles, one of such objects the Azure worker role instance is a windows server, this server associates with itself an agent that starts a worker role application and monitors it as well. A worker role application, also could be called as Photon role is a bootstrap program to start Photon and its applications. Worker role application is a managed application that is developed in visual studio using c# language. This worker role applications is packaged and uploaded to the Azure platform, Azure fabric then setups the windows server instance from it. The windows server instance is setup in several drives in the Azure, just by unpacking the uploaded package into several drives with standard naming formats as “C: Logs”,”D: OS”,”E: (or F :).Each time the package is updated next time, these drives remain unchanged and just the contents gets updated in same folder.

This setup also provides separate location in the drives for the logs, which helps to monitor the application, and folders to deploy the applicaitons.Azure also restarts the server instance in the same physical box. For publishing the Photon role applications, the files are split into different Zip files, the Photon related files are packackaged into one zip file, the photon role package is split into three archives. These files are then uploaded to the Azure data storage blobs. The files are named as photon-server.zip, photonapps.zip, photon-bintools.zip and the photon-role, of these files the photon-role file does the task of downloading the archives as a part of the bootstrap process. This file structure for the photon applications is same for any platforms, this allows users to use the same application files in any platforms and so it provides access to Azure features like blob and table storage from the photon applicaitons.But from the photon application files, it is not possible to access the worker role-application specific API, such as APIs for service configuration update event. Photon role application installs the components in the folder locations as follows: “C:\photon-logs:” for logging, “E:\Photon\Deploy: “folder contains the win64 Photon binaries and resembling the structure known from the path {photon-sdk}\deploy}.

PASK components include a batch file, project file, AzureLoadbalancing, AzureLite and Tools like AzureFileTransfer. These components are installed in the location "{photon-sdk}/src-server/Azure”. The batch file named “azure.build.prompter.cmd" used to build the zip files that contains the Photon and its application and uploads it to the local storage or to the cloud. The project file "azure.build.proj" is used by build prompter as a build file.

AzureLite Component consists of two parts, the AzureWorkerRole for bootstrap and monitoring the Photon Server and a Photon Server for hosting the Photon Appli-acitons.AzureLite component inherits from Lite of the Photon and other application configuration files of Photon. The components includes the AzureDeploy which is the visual studio cloud service project ,with the description and configurations that are used to publish the AzureWorkerRole, which is specific for the AzureLite. The WorkerRole inherits from WorkerRoleBase. AzureLoadbalancing components has a Photon application which inherits from the Load balancing configuration of Photon and its

Application configurations. It also consists of a specific worker role called "Azure.Load-balancing.WorkerRole" for configuring the Load balancing configuration files. Az-ureShared component is a WorkerRoleBase and it is the core implementation of the Photon Role .AzureFileTransfer component is used to upload the files to the Azure Blob storage, and it sued by the build-prompter. Networking features in the PASK are provided by the Azure worker role instances, these instances are deployed as cloud services. Only one IP address is assigned for each Cloud Services, the instances function behind the load balancer. Azure allows configuration of a port for the load balancer probes, which help to check the health of the instance of the worker roles, if one instance fails, the load balancer probe is taken out of rotation and no traffic is routed to that instance any more, this is done by a concept called custom probes.

To work with the Azure PASK following tools are needed, Visual Studio 2010 or 2012, the latest Azure SDK need be installed with storage and compute emulator, CloudX-plorer or [Azure Storage Explorer] (http://azurestorageexplorer.codeplex.com/) for ex-plorer like access to the blob storage, the latest Photon SDK and a Windows Azure ac-count .PASK is not used more currently due to its complexity.

Second method of running the photon server SDK on the Azure virtual machine is con-sidered as the best solution for using the photon for games. As the Azure worker role has more complications than the first method, the second method of using virtual ma-chines was introduced as replacement for using PASK.

3.6.3 Photon Unity Networking (PUN)

Photon provide the APIs that are compatible with the Unity's networking services. PUN features can easily be integrated to the unity project by installing the PUN package from the assert store. PUN provided scripts that could be used to create the multiplayer games with unity. The simple steps to create the multiplayer online game using PUN is detailed in the reference [12].

PUN APIs are used to create a game server, join a game server to find active hosts, create player or object over the network, spawning the network players which allows the players to connect to one another and play , implement network communication by creating rooms, and interpolate the game data ,as there might be delay in the send rate in each player, so the send rate for packets in each player could be set to certain time for example every 5 seconds and the data is synchronized on that time ,so that it is easy to predict synchronized game data.

The PUN package for unity provide namespaces Photon.NetworkPlayer and Photon.MonoBehaviour. The PUN also supports serialization of the game players and the game data according to how the users want to serialize e the data of the game.

3.7 Unity Game Development Tool

Game development requires usage of methods designed for interaction called Game Mechanics, which is supported by a good design of Game Objects and Game World. Game engines usually provide an Integrated Development Environment, where all activities and tasks related to game development are seamlessly integrated using coding, objects, and environment creation. Unity is a game engine, which is very popular among the game developers, hobbyists, and developers who are new to programming as well, it provides a good tool to design and develop games, focusing essentially on the game mechanics, rather than the underlying layers necessary to build a game. Unity tool makes it possible for designers to control the logic of their game using high level programming or scripting languages, including C #, JavaScript or Boo Hence decreasing the learning curve and improving the workflow. There are many game engines available, but Unity is one of the very few that provides a significant number of tools and techniques

that simplify the development process, help to produce high-quality games, and addresses many aspects of game development, including an Integrated Development Environment , Artificial Intelligence , animations, or lighting [13].

Game Engines are mostly used for production of video games, but with evolution of technology and provision to export objects to different formats to be supported in web and mobile devices, they are now used for a wide range of applications, with purposes other than gaming. As an example game engines are now employed for simulation, teaching, and training, as they often make it possible to create and manage very realistic environments easily. Moreover, such tools provide means for the creation of universally accessible environments. Having got more user needs, the gaming technologies are now supporting distributed objects over internet and communicate online.

Unity is built with simplicity and effectiveness in mind to allow both novice and advanced developers to maximize their game creation experience. It makes it possible to develop games of different genres such as platformers, role playing games, first-person shooters, massive multiplayer online role playing games, simulations, or strategy games, and for a comprehensive number of platforms, such as Android, iOS, Windows Phone 8, PC, Mac, Linux, PS3, or XBOX 360.and Web Browsers.

Unity has various build in capabilities, including to this it offers the possibility to employ third-party plugins that greatly enhance the workflow and add some very interesting effects and functionalities. Unity includes a built-in access to the assets store, an online store that provides material for our Unity projects, such as example, textures, characters, GUI systems, and scripts. While the majority of these items have to be purchased, some of them can be imported in user's project for free.

3.8 IMS Architecture

IP Multimedia Subsystem (IMS) is service bundling technology. It has created a change from usage of the voice only communication to usage of advanced communication with multimedia, like allowing video calls and media files sharing. IMS provides service driven business model. It has enabled transition from network centric, single services to subscriber centric offering, which offers low cost service economics and high number of users.

IP Multimedia Subsystem is the service engine for full convergence, for combining the usage of different kind of devices on one platform. It is IP based and web integrated platform. It provides application mobility, enabling media communication from mobile to several devices like TV, PC and home phones .It provides network mobility and access mobility. IMS services is limited only by the demand and the ability to sell them.

IMS is an application driven architecture, it enables convergence by service creation and deployment across various devices. It is a framework of many combined services, it provides services like voice and video messaging and conferencing, presence and multiplayer gaming. Service providers have more benefit by moving towards using equipment that fits in the IMS standards.

IMS benefits are that it enables resources sharing with the help of media servers, usage of common user data, and usage of single user data across applications. With the advances session control support like using SIP protocol for session control, it eliminates, multiple service solutions, gives consistence services, single presence status no matter what ever is the access type and network transparency.

The OSI, Open System Interconnection reference model developed by the International Organization for Standardization, ISO, provides a reference model for communication between devices the model defines seven layers stack through which the communication occurs, each layer provides a distinguished and well defined services to the adjacent layer in the stack. The Layer 1-4 has most common protocols for many frameworks, and the layers 5-7 has some difference based on the service the framework provides, for IMS the Layer 5 has the SIP, RTP and RTCP protocols, and the Layer 6 and 7 together provide the Application servers. The OSI model for IMS is explained in the Figure 12.

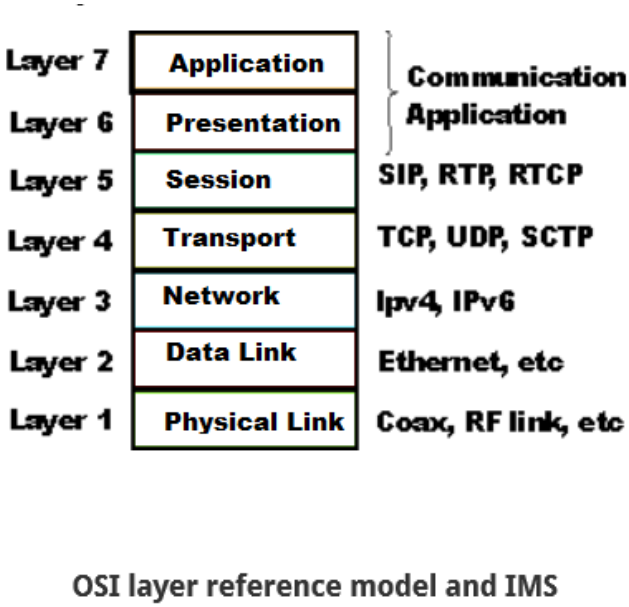


Figure 12 OSI Layer Model for IMS [14]

IMS Framework layer Model gives the overview of the key components in each layer of OSI model that make the IMS protocol. The Key components in each layer for IMS protocol is explained in the Figure 13. It consists of the Device Layer, Transport Layer, Session Control Layer and the Application Services Layer.

In IMS the transport layer is mainly functioning to initiate and terminate the SIP signalling that are needed for setting up sessions. Transport Layer takes care of providing the gateways that are needed to communicate with the different types of networks, such as sessions between the circuit switched network and the packet switched networks, this is facilitated by the means of media gateways. The media gateways are used to convert VOIP media Streams into the format that could be used to transmit through the circuit switched networks. This layer provides services to convert the analogue or digital formats to packets. This layer also help to provide media related services like speech recognition, speech synthesis, conferencing, collecting signalling tones and playing announcements.

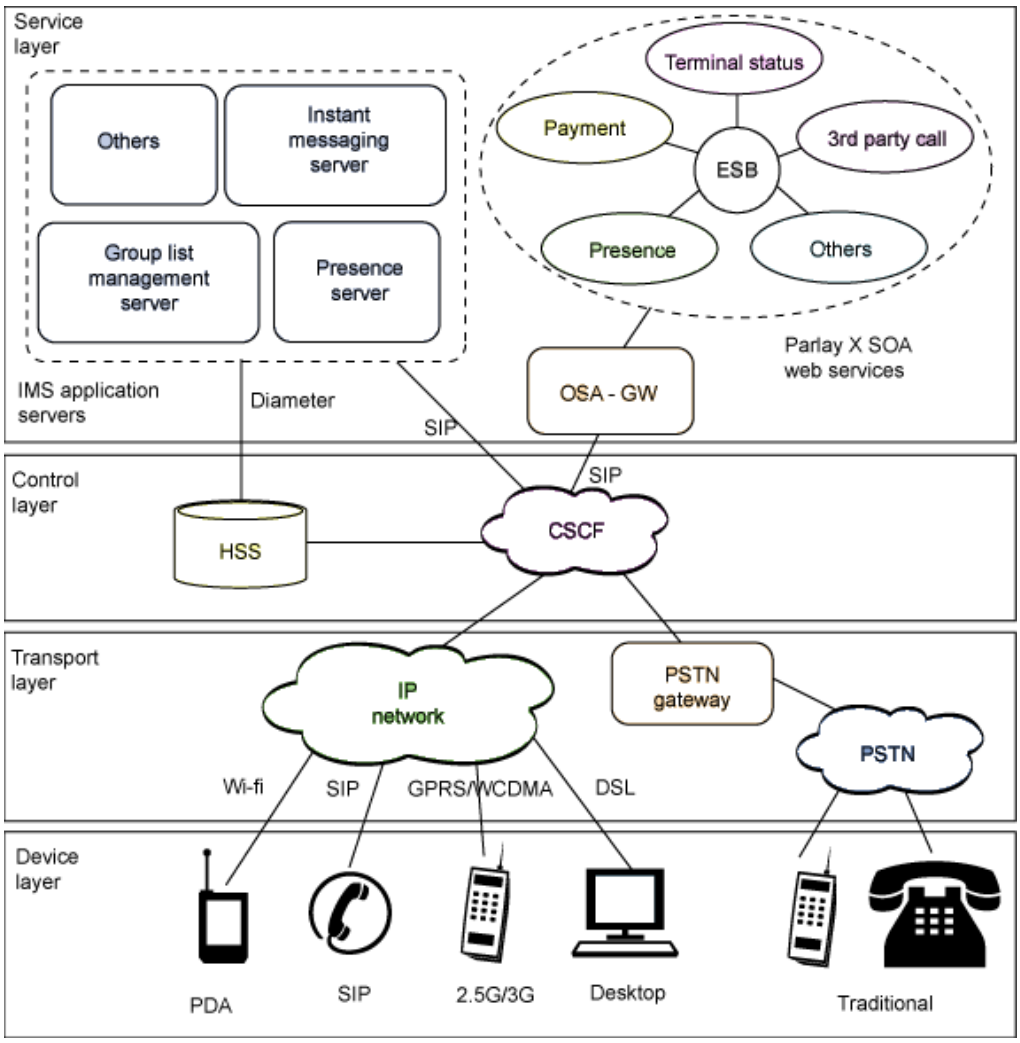


Figure 13 IMS Framework Layers Model [15]

The session control layer contains the components Call Session Control Function (CSCF) and Home Subscriber Server (HSS).The CSCF takes care of registration and routing of the SIP signalling messages between the endpoints, which helps to route the signalling messages to the correct application servers.CSCF also provides the QoS (Quality of Service), for ensuring the quality of services, but communicating transport layer and the devices layer. The HSS component is used for maintaining the user profiles, with the details of user regisration,preferences and other similar data.HSS also includes the presence server, which is needed for interactive applicaitons,for example Push to Talk over Cellular(PoC).In addition to the CSCF and HSS, the session control layer also includes the Media Gateway Control.

The Application Service Layer manages the end services that are required by the users. In this layer various servers are supported, some of such servers include the IP Multi-media - Service Switching Function (IM-SSF), Supplemental Telephony Application Server, Open Service Access - Gateway (OSA-GW), Web Services, Enterprise Services Bus (ESB) and other similar services. This layer .With the help of this layer, IMS layers are designed to be more flexible and making it possible to provide different kind of te-lephony and non-telephony services at the same time.

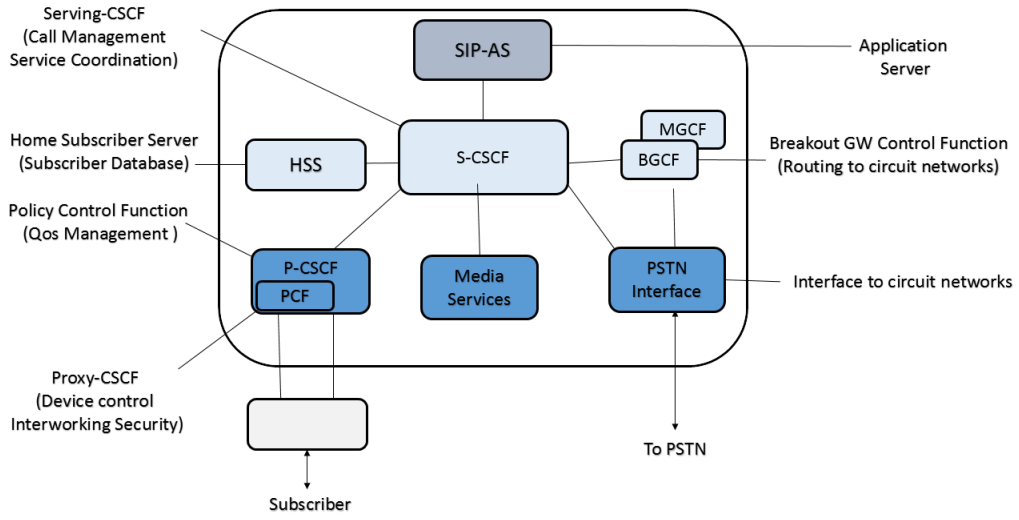


Figure 14 Function View of IMS Framework [16]

The Functional View of IMS Architecture is described in Figure 14.The core entities of the IMS architecture are the Serving-CSCF, HSS, Policy Control Function, Proxy-CSCF, Application Server and the Breakout Gateway Control Function. The CSCF components are responsible for the registration of endpoints in the sessions, they also do routing of the SIP Messaging Signals, and interlinking the network and transport layers to provide Quality of Services. The HSS component provides support to store the subscriber data for home networks. THE MGCF and BGCF components helps in signalling across differ-ent kind of networks and media endpoints. The Media Services component manages the media resources. The Application Server is the component in which the services for end user are deployed

3.9 SIP Protocol

SIP Protocol is used for creating multimedia sessions in an IP network, the protocol is used to establish, modify and terminate the multimedia sessions. Its applications include, but are not limited to, voice, video, gaming, messaging, call control and presence [17].SIP is a part of multimedia architecture, the SIP protocol itself, along with other related protocols are monitored and standardized by Internet Engineering Task Force (IETF).An idea to have a session signaling protocol over the IP network was initially designed during 1992, mainly planned for multicast conferencing, four years after that SIP protocol originated as a component of IETF multicast backbone (Mbone).SIP was initially an experimental multicast network on top of the internet. SIP protocol was mainly used by IETF for sharing the multimedia contents, which includes the IETF meetings, conferences and seminars. SIP was then adopted as VoIP signaling protocol, and it became the IETF proposed standard as [RFC2543]. SIP protocol was enhanced further to address the interoperability issues, with better design and new features. The actual protocol document was rewritten completely for clarity, the newly created document became the proposed standard as [RFC3261], but still the protocol remains backward compatible with the old document [RFC2543],though it made [RFC2543] obsolete.

Table 2 Main RFCs related to SIP Protocol [17]

RFC	Title
2046	MIME part two, media type
2246	TLS protocol version 1.0
2617	HTTP Authentication: Basic and Digest Access Authentication
2778	A Model for Presence and Instant Messaging
2779	Instant Messaging/Presence protocol requirements
2806	URLs for telephone access (deprecated by RFC 3966)
2848	PINT
2915	NAPTR DNS Resource Record
2916	E164 number and DNS
2976	SIP INFO Method
3087	Control of service context using Request-URI
3204	MIME Types for ISUP and QSIG
3261	SIP : Session Initiation Protocol
3262	Reliability of provisional responses in SIP (PRACK)
3263	SIP : Locating SIP servers
3264	Offer-Answer model

3265	Event packages
3266	Support for IPv6 in SDP
3310	HTTP digest authentication using authentication and key agreement (AKA)
3311	UPDATE method
3312	Integration of resource management and SIP
3313	Media authorization
3320	Signaling Compression (SigComp)
3321	SigComp extended operations
3323	'A Privacy Mechanism for the Session Initiation Protocol (SIP)'.
3324	Short Term Requirements for network asserted Identity
3325	Private extensions for SIP for asserted identity within trusted networks Reason Header
3326	SIP extension header field for registering non adjacent contacts (Path) Security Mechanism Agreement
3327	SIP-T
3329	Grouping of media lines in SDP
3372	ISDN ISUP in SIP
3388	Internet media type message/sipfrag
3398	Change process for the Session Initiation Protocol
3420	SIP extensions for instant messaging
3427	Private Header (P-Header) Extensions to SIP for the 3rd-Generation Partnership Project (3GPP)
3428	SIP and SDP static dictionary for Signaling Compression
3486	compression of SIP messages
3515	SIP REFER method
3520	Session Authorization Policy Element
3550	RTP: A Transport Protocol for Real-Time Applications (updates RFC 1889)
3578	Mapping of ISDN ISUP Overlap to SIP
3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing
3608	Extension Header field for service route discovery during registration
3665	SIP Basic Call Flow Examples
3666	SIP PSTN Call Flows
3680	'A Session Initiation Protocol (SIP) Event Package for Registrations'
3725	Best Current Practices for 3 PCC calls flows

3761	ENUM
3824	Using E.164 with SIP
3840	Indicating User Agent Capabilities in SIP
3842	Message Summary & Waiting Indication Event for SIP
3856	A presence Event Package for the SIP
3857	SIP Event Template Package for Watcher Information
3858	An extensible markup language (XML) based format for Watcher Information
3862	Common Presence and Instant Messaging : Message Format
3863	Presence Information Data Format (PIDF)
3891	SIP Replaces header
3892	SIP Referred-By mechanism
3903	SIP extension for event state publication
3966	Subscriber Number Scheme
4028	Session Timers
4032	Update to the Session Initiation Protocol (SIP) Preconditions Framework
4040	RTP Payload Format for a 64 kbit/s Transparent Call
4353	A Framework for Conferencing with the Session Initiation Protocol (SIP)
4538	Request Authorization through Dialog Identification in the Session Initiation Protocol (SIP)
4579	Call Control— Conferencing for User Agents
4575	A Session Initiation Protocol (SIP) Event Package for Conference State
4733	RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals (updates RFC 2833)
5629	A Framework for Application Interaction in the Session Initiation Protocol

Table 2 provides the list of the main RFC related to the SIP Protocol.

SIP Protocol is based on the Hyper Text Transfer Protocol (HTTP) and the Simple Network Management Protocol (SNMP). Figure 15 shows where SIP fits into an Internet Multimedia protocol stack [18].

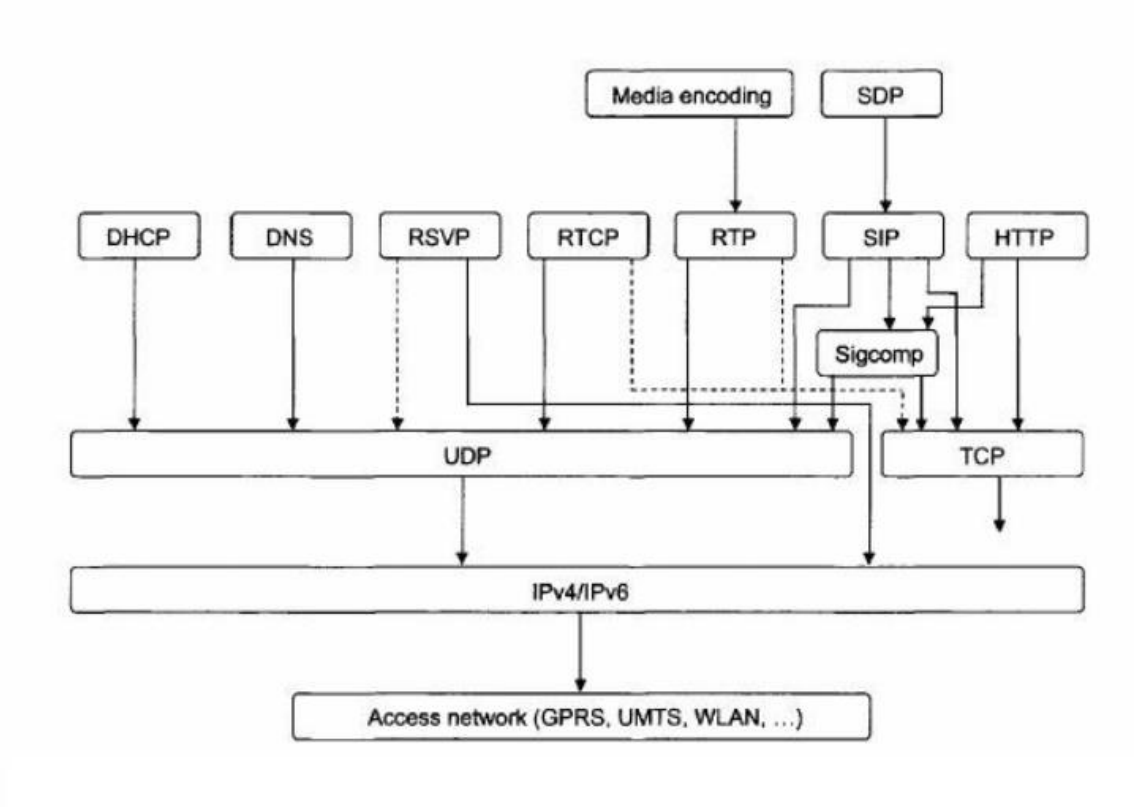


Figure 15 Internet Multimedia Protocol Stack [18]

SIP was initially designed for multimedia Conferencing. SIP was initially designed for multimedia Conferencing. The key design goals of the protocol are to have Transport Protocol neutrality, Request routing, to have new application of media and to have more extensibility and personal mobility. Hence the protocol supports the transport protocol to run over reliable and unreliable protocols. It provides good performance, good control of packets, and also provide a clear separation of media and signal description with help of proxy routing.

3.9.1 SIP Components

SIP components are the elements that are involved in a multimedia session using the SIP protocol, these elements can be classified into user agents (UA) and intermediaries (servers). The communication between the endpoints (UA) in a session can happen without any intermediaries, but in many cases the service providers and the network administrators would wish to track the traffic in their network for this the intermediaries are needed. Figure 16 represents a typical network set-up using SIP protocol, referred to as

the “SIP trapezoid” [19].Hence the components of SIP can be represented using a trapezoid consisting of User Agents, Proxy Server, DNS server, Location server and Inbound Proxy servers.

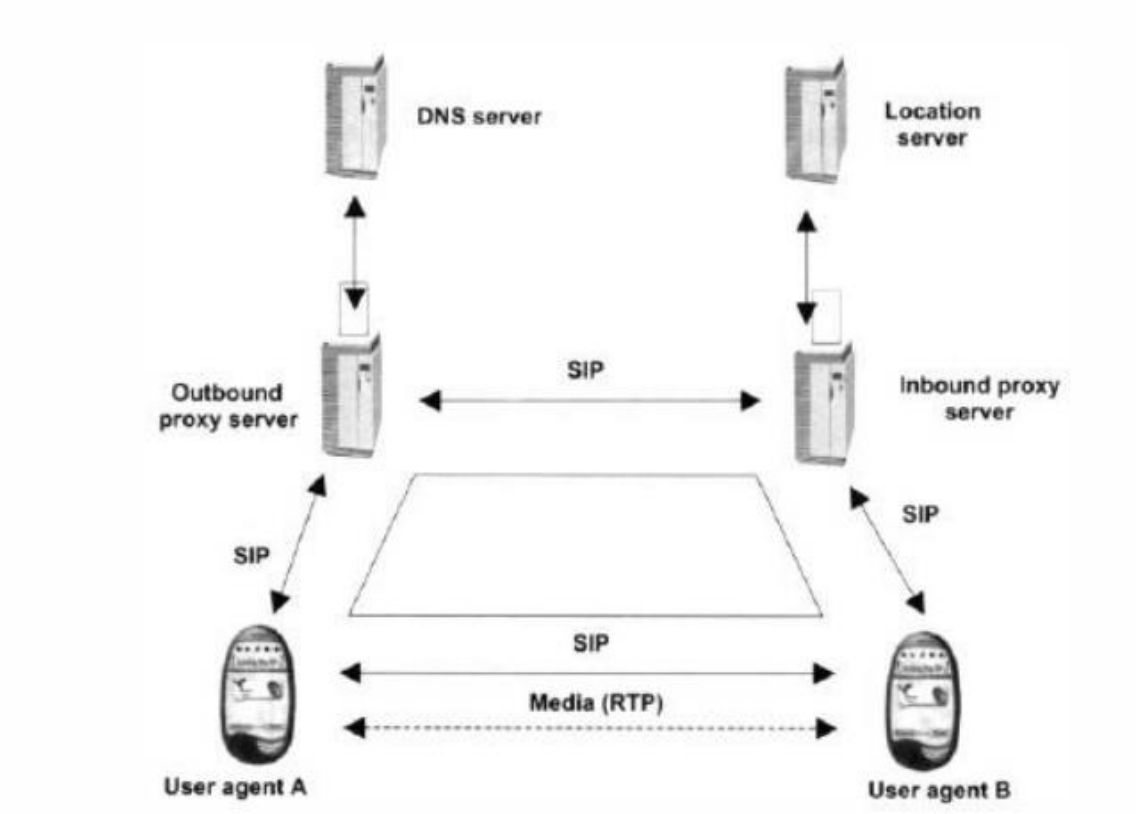


Figure 16 SIP Trapezoid [19]

User Agent (UA) is the end point of a session, it sends and receives the SIP requests and receives the SIP responses, it is usually the user's device a dedicated hardware appliance or an application in a terminal. The User agents are of two types, the one is the User Agent Client (UAC) and the other is the User Agent Server (UAS).UAC is the caller application that sends the request to initiate a session. UAS is the end point which accepts, redirects or rejects the requests. UAS sends responses on behalf of the user.

SIP intermediaries are the components where the SIP message passes through before reaching the final destination. Intermediaries route and redirect requests. The different types of intermediaries entities are Proxy server (Inbound Servers), Redirect servers (Outbound Servers), SIP gateways and Application Servers.

A SIP proxy server receives a SIP request from a user agent or another intermediaries, also it acts on behalf of the user agent to forward or respond to the request. A SIP proxy forwards IP messages at the application layer, in the same way the router forwards the IP packets to the IP Layer. A proxy server could only modify requests and responses according to the rules set in RFC 3261. These RFCs makes sure that the SIP signalling is handled transparently, but still allow the proxy server to give a valuable performance and services. A proxy server also could send request to a number of locations at the same time. Such proxies which send request to multiple end points are called Forking Proxies, these proxies can work parallel or sequential. There are three proxy server types: Dialog statefull proxy, is a proxy is dialog statefull if it retains the state for a dialog from the initiating request (INVITE request) to terminating request (BYE re-quest).Trans-action statefull proxy maintains client and server transaction state machines during the processing of a request. Stateless proxy is the server machine that forwards every re-quest it receives downstream and every response it receives upstream [19].

Redirect servers (Outbound Servers) maps and converts the address of the SIP requests into new addresses. It acts to redirect the request and does not participate in any trans-actions. Location server is used to keep track of the location of users. Registrars server accepts REGISTER requests.

Registrar servers are used to store explicit binding between a user's SIP address and the address of the host machine where the user is at the receiving end.

Application server (AS) is an element that provides end users services in the network, such as such servers for presence and conferencing.

Back to back user agent (B2BUA) is where a UAS and a UAC are glued together. The UAS terminates the request as a normal UAS. The UAC initiates a new request that is somehow related to requests received at the UAS side, but not in any protocol-specific link. The main purpose of this entity is that it acts like proxy, but it breaks the rules of a proxy and it can modify the requests.
SIP gateways play key role in establishing the session across user agents connected in different type of networks and in different locations. SIP Gateways are applications which provide the support to interface a SIP network to other networks which use differ-ent protocols or technologies. A SIP gateway is an application that interfaces a SIP net-work to a network utilizing another signaling protocol for example PSTN network. In

terms of the SIP protocol, a gateway is just a special type of user agent, where the user agent acts on behalf of another protocol rather than a human.

SIP uses Uniform Resource Indicators or URIs for most addresses. For SIP, the URI scheme is either sip for a normal SIP URI or sips for a Secure SIP URI. Secure SIP means that a SIP message sent using this URI will be protected using TLS across each hop. SIP URIs must contain either a host name or an IP address. They usually contain a user part, although they do not have to. For example, a URI for a proxy server typically will not have a user part. URIs also may contain parameters such as passwords. SIP URIs and SIP names could be used as login credentials to establish the session and authentication. SIP protocol can be combined with other protocols like Diameter protocol, which provides complete authentication services, SIP services along with other protocols could be utilised together to take into use the services such as messaging, voice call, videos calls and other data transfers.

3.10 IMS Gaming Platform

The gaming platforms using IMS architecture and that are available in the market for the game developers are very few, there are only one or two such platforms that are designed using IMS platform such as Nexos. Nexos is a closed platform and not available for free, it was not possible to experiment it. However, the architectural details available at Nexos website [20] give a very good picture of the capabilities of the IMS framework used as a gaming platform. Nexos platform uses the Rich Communication Services to build the rich communication features for the gaming platform. Rich Communication Services is a program for the creation of inter operator communication services based on IMS, developed by Summit Tech.

Rich Communication Services by Summit Tech, provides IMS SDK and clients like joyn and Rich communication suite. Joyn is the first IMS client to get the GSMA accreditation RCS also now provide the plugins for using the IMS features in the Unity Games. The game developers could use the RCS plugin for the chat and other media communications in their game, this reduced the need for them to concentrate on developing communication software and they could concentrate on developing the games instead.

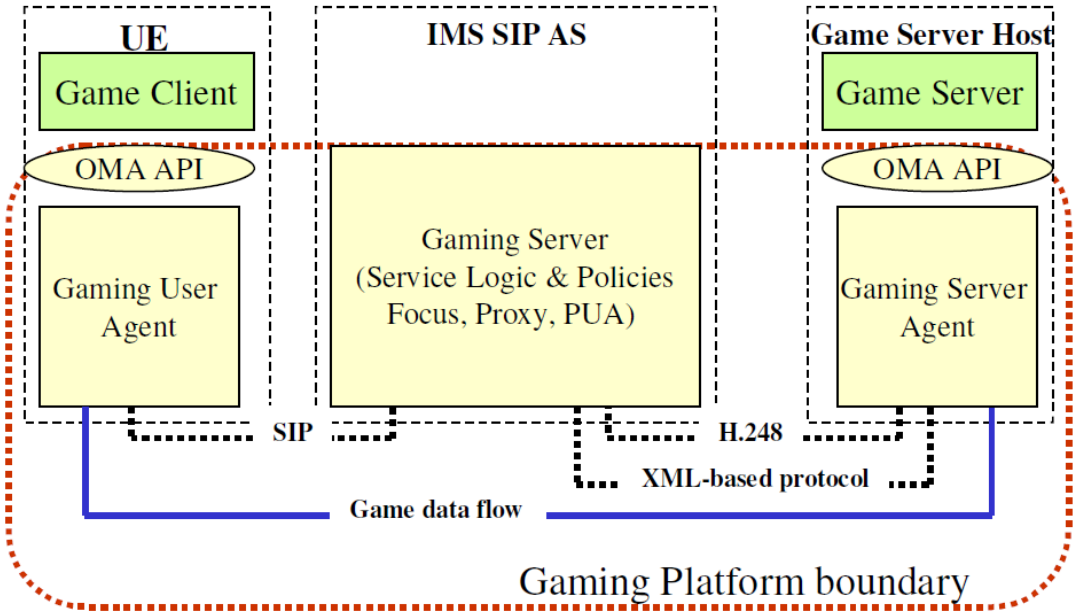


Figure 17 Overview of Model Gaming Platform using IMS SIP protocol [21]

Other that the RCS, on exploring the history for any projects or games developed using the IMS frame work, there are not many papers or projects implemented to build a game using IMS, one of such project is explained in the reference [21]. In this white paper IMS platform is experimented for using in game propose, using SIP Protocol, combined with the H.248 Gateway protocol .The game client and Servers could be developed using the Open Mobile Applications APIs.The communication between the game client and the server is done with the aid of the IMS Application server which implements the Service logics and game server logics.The overview of the IMS gaming platform model is shown in Figure 17[21].

In Figure 18 [21] an overview of signal flow for a game using the IMS SIP protocol, the signal flow starts by sending a SIP INVITE message containing the SIP URI of the game session to the Gaming Server (Game Focus). The Game server receives the SIP INVITE message and performs tasks such as authorization and authentication of the user, and evaluates rules for policies, then sends an ADD message to the Gaming Server Agent (Game Server). The Game Server receives the new user information as the session description in the INVITE message, then it allocates the resources and adds the new player to the game, then the Gaming Server sends SIP OK message as the response to the user, along with this message the Game Server's session description is also send. Afterwards the user establishes the game data connection with the game server depending on the received session description of the game server. Now the players can start playing the game. The game users are interested to obtain the updated game related information, for example the top score in the game, to obtain this information the user sends a SIP SUBSCRIBE message to the Gaming Server (Game Focus).On receiving the user subscription message the Game server responds with a SIP OK message. After some time, when the Game Server modifies some game information. It notifies the user about the change immediately, by sending a SIP NOTIFY message. Then the user would responds with a SIP OK message.

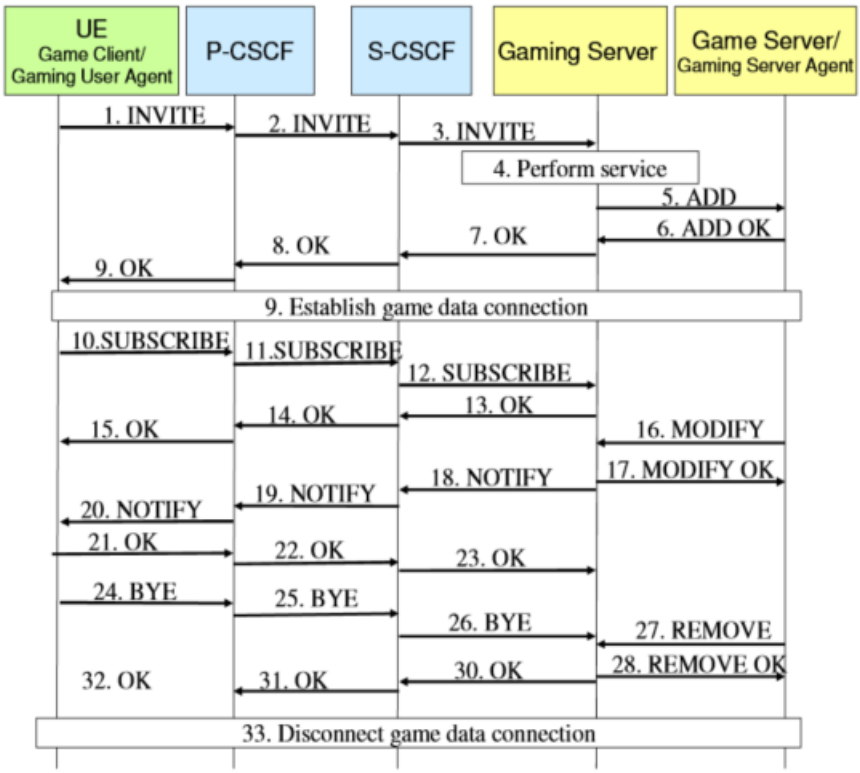


Figure 18 Model Session Signals in a Game using SIP protocol [21]

The IMS Presence Service could be used to know the status of the other players, such as which player is active or inactive, this service may be involved in the subscription and notification service for the user. Game Services then publishes the information of the games users to the Game server. When the user wants to leave then the user sends the SIP BYE message to the Gaming Service to end the session. Then the Gaming Server removes the user from the game session and sends REMOVE message to the Game server. After that the resources allocated for the user is removed and the data connection is closed. In the same way similar session can be added to establish new message and media communication further.

4 Implementations

In this section the details of the implementation to understand the different methods for providing login services for a game application are explained. First the SIP protocol setup is explained, in which the creation of SIP user's credentials for login services is detailed. Next the login service implementation for the unity game understudy is detailed. After that the details of the Photon Server for the game understudy is described and then the details of the usage of Azure mobile services for game development studied is described. Finally in the summary, merits and demerits of various authentication methods that could be used for login services in a multiplayer game is discussed. Also best ways to implement the suitable authentication services in the game understudy is suggested.

4.1 SIP Setup

For studying Login services using the SIP protocols a SIP server was setup and along with it two computer devices that have SIP client installed were setup. Overview of the setup is given in Figure 19.

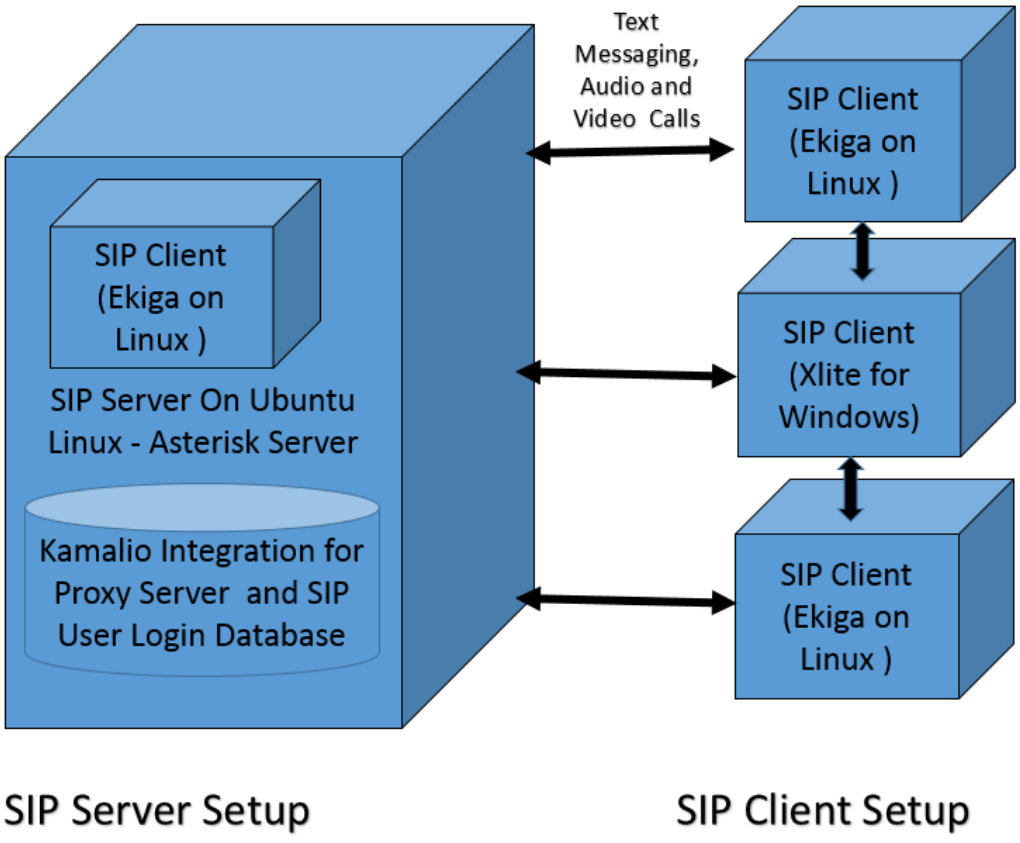


Figure 19 SIP Protocol Setup

The Server setup consists of the Asterisk SIP Server installation along with Kamalio integration, installed in a Ubuntu Operating System PC , in this PC a SIP client was aslo intalled for testing the setup. The Client Setup consists of PCs with Ekiga clients installed in it or a Xlite for Windows client installed. Using these setups ,the login, text messaging ,audio and vidoe calls between the clients were experimented.

4.1.1 SIP Server and Database Setup

For setting up a SIP server, a study on the requirements to setup a SIP server was studied and details of available open source SIP servers were studied. The SIP server requirement was that the server needed to support easy installation, and supporting integration for other services such as proxy server and data base for user login credential management. Asterisk was found to be the best server for this purpose, to serve as a proxy server and also support My SQL database setup with it. Hence, in this study the setting up of Asterisk with Kamailio was done. Asterisk is an open source framework for creating feature rich telephony systems, it is powerful and flexible framework. Asterisk offers advanced VoIP features, which could also be extended to be used in the multiplayer games for communication. Asterisk supports SIP Protocol and lets users to setup the SIP server for creating the media sessions between the users in the same network. For establishing the session between users who are located in different network or location, the SIP server need to support the NAT protocol, this could be done by integration of the other frameworks such as Kamailio which provide such services.

Earlier the OpenSER was the framework which provided the Open Source SIP server with the NAT supports, which acts as a Proxy server, the successor of this server is named as Kamailio, these servers are available under GPL license.Kamilio has high capability to handle huge number of call setups per second. Kamailio is used in large platforms for VoIP, presence, instant messaging, real time communication of media, and other applications. It is easily scalable to be used as SIP to PSTN gateways, for PBX systems or for media servers like Asterisk. The important aspects of the Kamailio server supports secure connection using TLS for VoIP protocol. It supports Web Sockets for IPv4, IPv6 and WebRTC, SIMPLE instant messaging and presence with embedded XCAP server and MSRP relay asynchronous operations. It supports various backend systems such as LDAP, Oracle, MySQL, JSON, SNMP monitoring [22].

Ubuntu Operating systems is a well know Linux operating system for servers, hence in this study the Asterisk and the Kamailio servers were setup in a computer with a Ubuntu operating system running on it. Initially the Asterisk server setup with SIP extension and without SIP user account was tried and details of this simple setup of server is detailed in Appendix 1.

Table 3. Steps to Setup Asterisk Server with Kamailio Integration [23].

Steps	Description of Step	Instruction
Step 1	• Build and Install Asterisk Server	• Command logs to build and install asterisk ○ *apt-get install mysql-server* ○ *apt-get install libmysqlclient-dev* ○ *apt-get install unixodbc-dev* ○ *apt-get install libmyodbc* ○ *cd /usr/local/src* ○ *wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-1.6.2.13.tar.gz* ○ *tar xvfz asterisk-1.6.2.13.tar.gz* ○ *cd asterisk-1.6.2.13* ○ *sudo apt-get install git-core build-essential autoconf automake libtool libncurses5 libncurses5-dev make libjpeg-dev pkg-config unixodbc unixodbc-dev zlib1g-dev* ○ *apt-get cache search libxml2* ○ *apt-cache search libxml2* ○ *apt-get install libxml2-dev* ○ *./configure*
Step 2	• Build and Setup Kamalio for database integration with Asterisk.	• Follow the setes in the link http://kb.asipto.com/asterisk:realtime:kamailio-3.3.x-asterisk-10.7.0-astdb ○ apt-get install gcc ○ apt-get install flex ○ apt-get install bison ○ apt-get install libmysqlclient15-dev ○ apt-get install make ○ apt-get install libcurl4-openssl-dev ○ apt-get install libxml2-dev ○ apt-get install libpcre3-dev ○ cd /usr/local/src ○ wget http://www.kamailio.org/pub/kamailio/3.1.2/src/kamailio-3.1.2_src.tar.gz

		○ tar xvfz kamailio-3.1.2_src.tar.gz ○ cd kamailio-3.1.2 ○ make include_modules="db_mysql dialplan" cfg ○ make all ○ make install ○ root@chithraPC:/usr/local/kamailio-3.1/share# cd kamailio/ ○ root@chithraPC:/usr/local/kamailio-3.1/share/kamailio# ls ○ dbtext mysql ○ root@chithraPC:/usr/local/kamailio-3.1/share/kamailio# cd .. ○ root@chithraPC:/usr/local/kamailio-3.1/share# cd .. ○ root@chithraPC:/usr/local/kamailio-3.1# find . -name kamailio.cfg ○ ./etc/kamailio/kamailio.cfg ○ root@chithraPC:/usr/local/kamailio-3.1# vi /etc/kamailio/kamailio.cfg ○ root@chithraPC:/usr/local/kamailio-3.1# ls ○ etc lib sbin share ○ root@chithraPC:/usr/local/kamailio-3.1# vi ./etc/kamailio/kamailio.cfg ○ root@chithraPC:/usr/local/kamailio-3.1# vi /etc/kamailio/kamctlrc ○ root@chithraPC:/usr/local/kamailio-3.1# /usr/local/kamailio-3.1/sbin/kamdbctl create ○ ERROR: database engine not specified, please setup one in the config script ○ root@chithraPC:/usr/local/kamailio-3.1# vi ./etc/kamailio/kamctlrc ○ root@chithraPC:/usr/local/kamailio-3.1# /usr/local/kamailio-3.1/sbin/kamdbctl create ○ root@chithraPC:/usr/local/kamailio-3.1#adduser --quiet --system --group --disabled-password --shell /bin/false --gecos "Kamailio" --home /var/run/kamailio kamailio • The above steps creates a setup which has creates 3 users with user name 1001, 1002, and 1003 and password for all as 1234, so now the SIP server and database with the user credentials is ready to be tested with the client.
Step 3	• Verify the SIP setup using the Xlite or Ekiga sip client.	• Install Xlite in a windows PC and try logging in with the use name and password created in the above steps.

Table 3 gives the steps to setup the Asterisk and integration with Kamailio as the Proxy server and MySQL database integration, for creating the sample SIP user account and storing them in the MySQL database.

4.1.2 SIP Client SetUp

To study the authentication and communication setup using the SIP protocol, the clients need to be installed and setup. There are several SIP clients which are available for free in the Internet, hence a study of the different SIP clients for different operating systems like Windows, Linux and Mobile devices was done, of which several clients like Xlite, Ekiga, Joyn were explored. The Ekiga client was selected for experimenting a detailed call flow study and it was installed in the same server machine, running Ubuntu operating system, and was used for testing the server and client communication. A more detailed description of the study is explained in the Appendix 1 and 2. Ekiga is an open source Softphone for VoIP applications .It supports high quality audio and video calls. It is used by many service providers as it is interoperable with many other standard compliant software, hardware and used by many service providers. It supports both SIP and H.323 telephony standards. The steps to setup the Ekiga client and SIP account in the clients and establishing the calls is explained in details in Appendix 2.

4.1.3 SIP Protocol in Games

SIP protocol could be used for Login and networking in the game applications as well. But currently there is no in-built support for this in Unity. Hence, it could be more tedious for game developers to concentrate on using the SIP protocol for their needs. There are other servers and packages available for login and networking services, such as Photon and Azure services, and they provide easily usable packages and are ready to use for login and networking services. Hence game developers prefer to use the readily available services, instead of using the SIP protocol, which requires comparatively more work for configuration of server, understanding and extending the protocol to be used for gaming purposes. But using SIP and other protocols in IMS framework, one could build a ready to use packages similar to the ones available currently like Photon, by using the vast features of IMS framework. In addition to login and networking, features like chat, lobby, voice communication and shared world could also be implemented using the IMS framework.

4.2 Unity Game Development

In this section the implementations done for the game client development of the game understudy are detailed. Unity Editor was used as the development environment. The

two main features that were prototyped were the login with the Facebook account and storing the login account credentials in a secure way, the implementation details of these two features are detailed in the sections below.

4.2.1 Secure Storage of Game Data

Unity supports client development using the .NET and JavaScript libraries, which provide support for developing games with massive graphics and assets. Unity tool provides various classes and dynamic libraries to address the needs for designing the user interfacing, data input, data storage and processing. Integrated Asset Store in the Unity development environment lets the users to use the readily available game objects, libraries and plugins for purchase or for free.

Unity tools support various APIs for developing games, it provides good support for designing graphics, built-in physics engine, advanced scripting, and networking system. Unity provides platform independent way of saving the data using the PlayerPrefs Scripting APIs [24]. These APIs work as a hash table, and lets user to store data in key-value pairs. The main problem with the built-in PlayerPrefs class is that the performance is very poor and is slow on iOS and even slower on Android. A test was done and based on that to store the data in and android device showed that only 6.25 key-value pairs could be stored per second [25]. If the number of records in a game is too high it could take way too long. Most likely, the PlayerPref APIs does file input and output operations for every modification. The current PlayerPrefs class in Unity is not designed well to handle large amount of data. Hence a custom storage class need to be programmed for faster saving.

A prototype storage class was implemented for the game understudy, this uses the classes from .NET framework, the encryption and decryption of the data is done using the code below, the methods are named as "*EncryptData*" and DecryptData

```
private string EncryptData (string inputString)
{
 DESCryptoServiceProvider enCryptoProvider = new DESCryptoServiceProvider ();
MemoryStream mStream = new MemoryStream ();
CryptoStream enCcryptoStream = new CryptoStream (mStream, enCryptoProvider.CreateEncryptor (bytes, bytes), CryptoStreamMode.Write);
StreamWriter sWriter = new StreamWriter (cryptoStream);
sWriter.Write (inputString);
sWriter.Flush ();
```

```
cryptoStream.FlushFinalBlock ();
return Convert.ToBase64String (mStream.GetBuffer (), 0, (int) mStream.Length);
}

private static string DecryptData (string encryptedString)
{
DESCryptoServiceProvider cProvider = new DESCryptoServiceProvider ();
MemoryStream mStream = new MemoryStream (Convert.FromBase64String (encryptedString));
CryptoStream cStream = new CryptoStream (mStream, cProvider.CreateDecryptor (bytes, bytes), CryptoStreamMode.Read);
StreamReader reader = new StreamReader (cStream);
return reader.ReadToEnd ();
}
```

"System.Security.Cryptography" Namespace provides cryptographic services, it is used to provide secure encoding and decoding of data, as well as many other operations, such as hashing, random number generation, and message authentication. DESCryptoServiceProvider Class defines a wrapper object to access the cryptographic service provider (CSP) version of the Data Encryption Standard (DES) algorithm. The public method named “CreateDecryptor“, creates a symmetric Data Encryption Standard (DES) decrypt object with the specified key and initialization vector. And the method "Public method “CreateEncryptor“, creates a symmetric Data Encryption Standard encryptor object with the specified key (Key) and initialization vector (IV). Using the above mentioned methods game data and login credentials can be encrypted in secure way for transmitting over network for authentication or for storing in the mobile device's applications storage area. This data could be send to the game database and stored remotely as well. All kind of data such as integer, string and game objects can be stored using the code in this encryption methods.

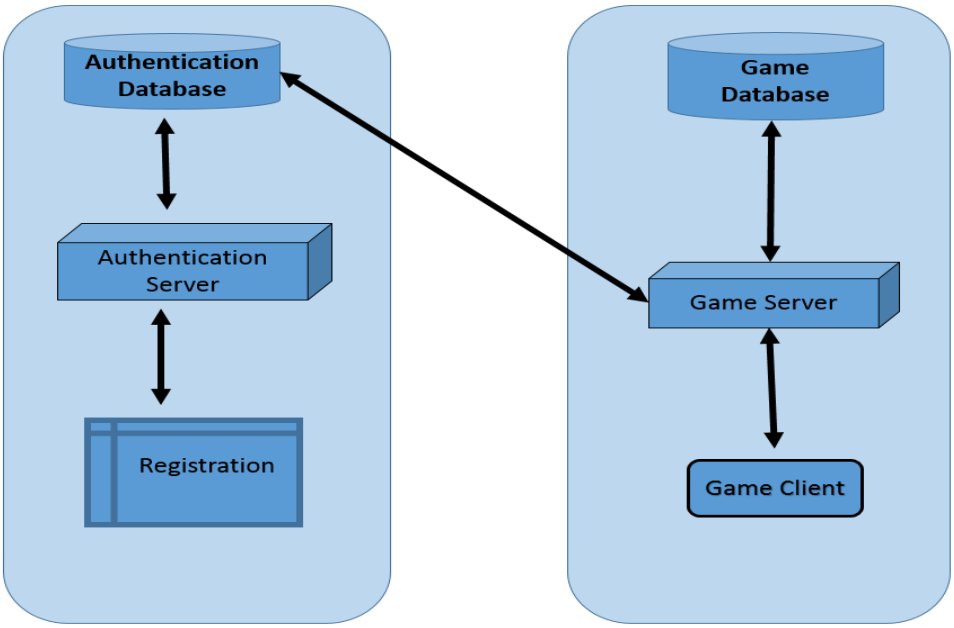


Figure 20 Game Data Storage Components

Figure 20 gives the details of the components that needs to be stored and accessed the secure data for authentication and gaming purpose. The authentication database is the data base where login and user specific details are stored, the game database is where the game data like game score are stored. Game server uses both the authentication database and game database to access the user's login and game data. In the implementation of prototype the encrypted login credentials were stored in a local file in the mobile device in which the game client is running.

4.2.2 Secure Login Using Facebook with Unity

A multiplayer game involves as many users as possible for its success, currently the social network that is very popular in the Internet and that supports gaming as well is Facebook. Hence the Vulpine games team has planned to have the Facebook login integration for their game understudy, and the Facebook integration with the unity game project was prototyped in the study.

The Facebook SDK for Unity comes with a comprehensive set of Facebook's social features, using which the players of Unity game could share content with their friends and lets the users to create a personal, social gaming experience. It provides the user interface for login and also provides ways to access the login token in a secure way which could be used for authentication with other services platforms like Azure.

Implementation of Facebook login services is done using the Facebook SDK package that is available for the Unity tool. Facebook SDK is available as a custom package and available for download freely [26]. User needs to download the SDK and install it to the Unity game project. The steps that are needed to integrate the Facebook authentication method that was experimented in the study is shown in Appendix 3. Facebook SDK provided the login UI screen that is built in in the Facebook SDK. Below are the User interfaces that are available for the login steps in the SDK. The below screenshots (Figure 21) are from the prototype game, that was developed for Android mobile device.

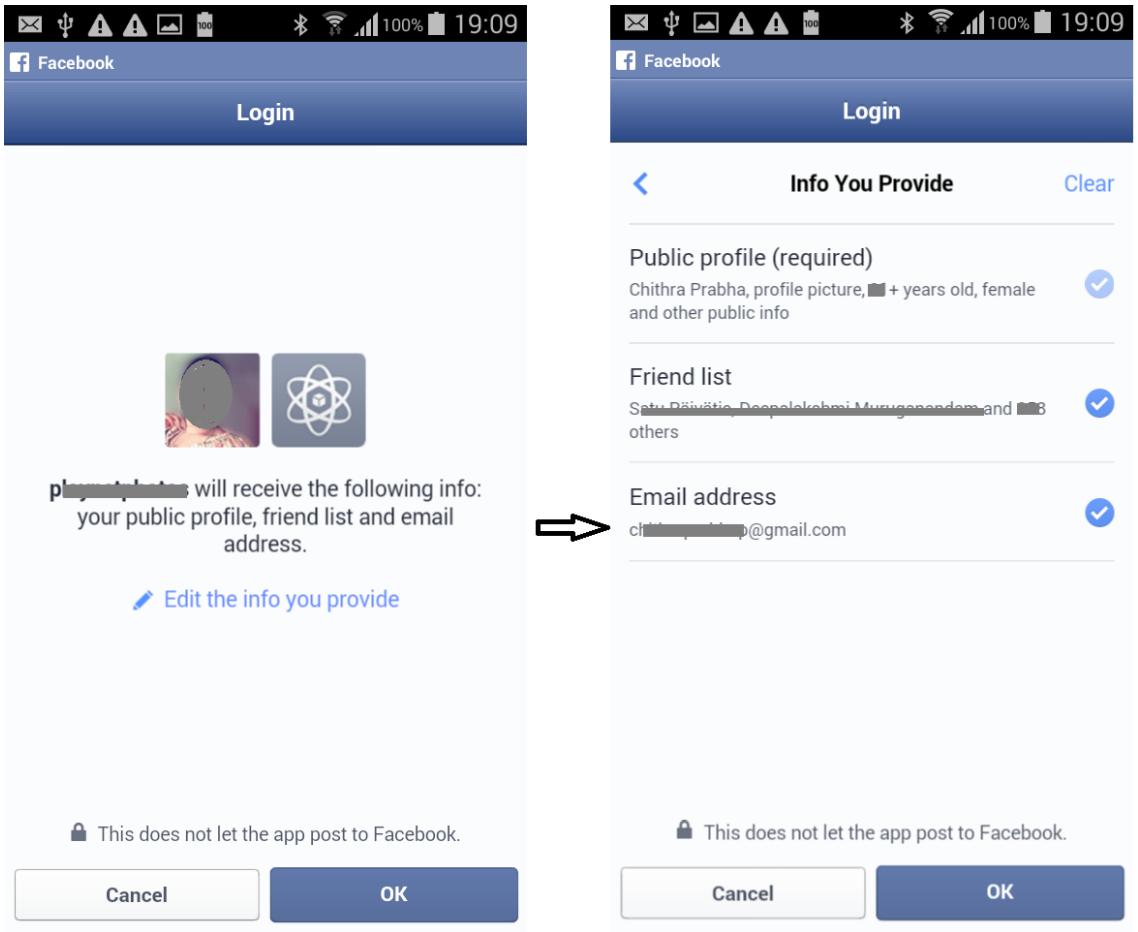


Figure 21 Login Screen

The code for the login is below, the Facebook SDK provided the APIs called FB.Login (...) for login which gets the public profile of the user with name and profile picture and then asks permission to get further details. When the login is successful and the user accepts to proceed further, using the FB.API (...) call with the proper query strings for the details that are required from and to the Facebook account, the game application is able to communicate with the Facebook account. This is implemented using the HTTP

requests, with the required framed query string as input. The sample code for implementing the Facebook login in unity game is detailed below.

```
public void Login2Facebook ()
        {
                FB.Login("public_profile,user_friends, mail,publish_actions",
                 Login Callback);
        }

void LoginCallback(FBResult result)
        {
                if (FB.IsLoggedIn)
                        OnLoggedInOK ();
        }

void OnLoggedInOK()
        {
                string meRequestString =
        "/v2.0/me?fields=id,first_name,friends.limit(105).fields(first_name,id,pic-
        ture.width(148).height(148)),invita-
        ble_friends.limit(105).fields(first_name,id,picture.width(148).height(148))";

        FB.API (meRequestString, Facebook.HttpMethod.GET, scriptAPI-
        Callback);
        }

void scriptAPICallback (FBResult result)
        {
                 if (result. Error != null)
                {
                // just try again until a successful result from Facebook
                        FB.API (meQueryString, Facebook.Http-
                Method.GET, scriptAPICallback);
                         String resultData = result.Text;
                        return;
```

```
        }
    }
```

4.2.3 Azure Mobile Services

Azure mobile services are designed to get users up and going with storing data, push notifications and authentication for the components that the user needs for the game. Figure 22 gives an overview of how the Azure Mobile services can be used to authenticate the devices using the different social networking sites such as google, Facebook, twitter and Microsoft Account. The client in the device needs to provide the credentials for the social networking sites for login and when the credentials are verified, the access token is validated by the azure mobile services and then the access to the azure service for the devices whose identity is verified with the token is made available.

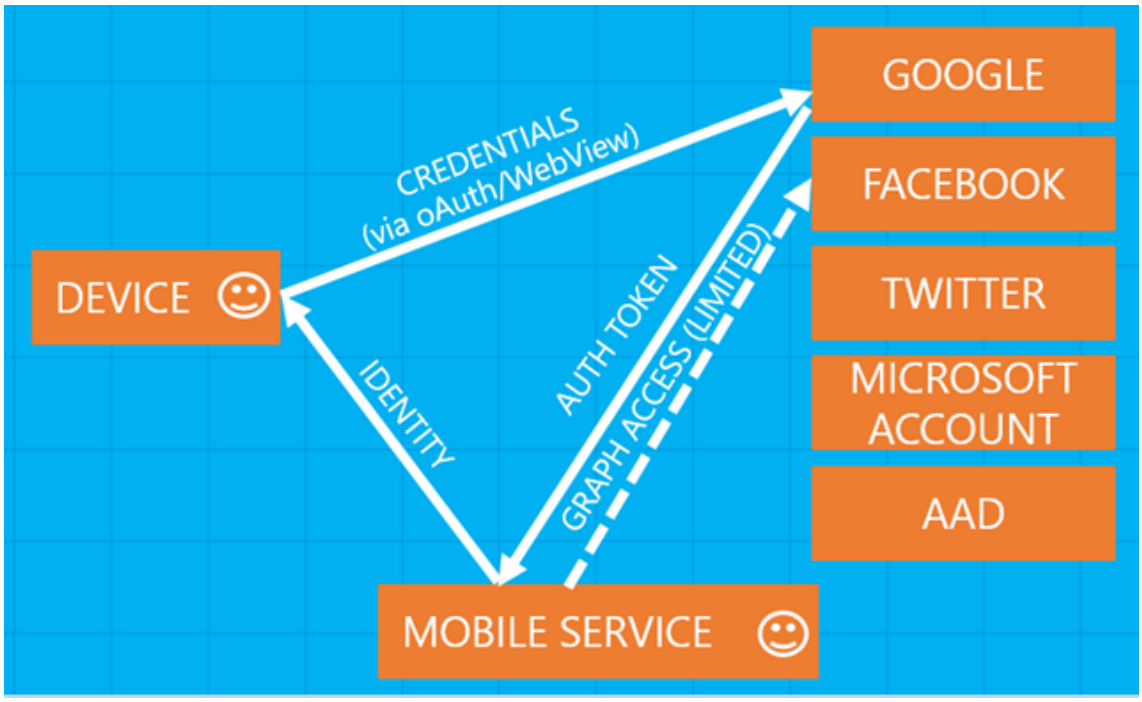


Figure 22 Azure Mobile Services Authentication Overview [27]

In the game understudy the game server was supposed to be implemented on the Azure Virtual machine, hence a study on how the authentication to the server application in the azure machine was done. For this the steps to create the Azure Mobile services and Setting up of the Facebook access tokens for login in the Azure mobile services portal were experimented, the steps for this are detailed in Appendix 5. For integrating the

Azure mobile services to the Unity game projects, it is required to write the Azure plugin for Unity. BitRave is a sample plugin that is available for the users under Massachusetts Institute of Technology (MIT) License for free. BitRave plugin provides sample unity scripts and assets to integrate the Azure data storage services, with Facebook authentication. The steps for trying this are explained in Appendix 5.The plugin implements the authentication requests from the unity scripts to the Azure services using the RestSharp APIs. The sample code from the plugin for login request is below [28].

```
public void LoginAsync(AuthenticationProvider provider, string token,
            Action<AzureResponse<MobileServiceUser>> callback)
    {
        AuthenticationToken authToken = CreateToken(provider, token);
        _LoginAsyncCallback = callback;

        var path = "/login/" + provider.ToString().ToLower();
        var baseClient = new RestClient(_baseEndPoint);
        var request = new RestRequest(path, Method.POST);
        var json = SerializeObject(authToken);

        request.RequestFormat = DataFormat.Json;
        request.AddHeader("Content-Type", "application/json");
        request.AddParameter("application/json", json, ParameterType.RequestBody);

        var handle = baseClient.ExecuteAsync<MobileServiceUser>(request, LoginAsyncHan-
        dler);
    }

private void LoginAsyncHandler(IRestResponse<MobileServiceUser> restResponse,
            *RestRequestAsyncHandle handle)
    {
        var response = DeserialiseObject<MobileServiceUser>(restResponse);
        response.handle = handle;
        _LoginAsyncCallback(response);
    }

 private static AuthenticationToken CreateToken(AuthenticationProvider provider,
            string token)
    {
        AuthenticationToken authToken = new AuthenticationToken();
```

```
switch (provider)
{
                    case AuthenticationProvider.Facebook:
                    case AuthenticationProvider.Google:
                    case AuthenticationProvider.Twitter:
                      {
                        authToken = new FacebookGoogleAuthenticationToken()
                        { access_token = token };
                        break;
                      }
                    case AuthenticationProvider.MicrosoftAccount:
                      {
                        authToken = new MicrosoftAuthenticationToken()
                                    {authenticationToken = token };
                        break;
                      }
    }
    return authToken;
}
```

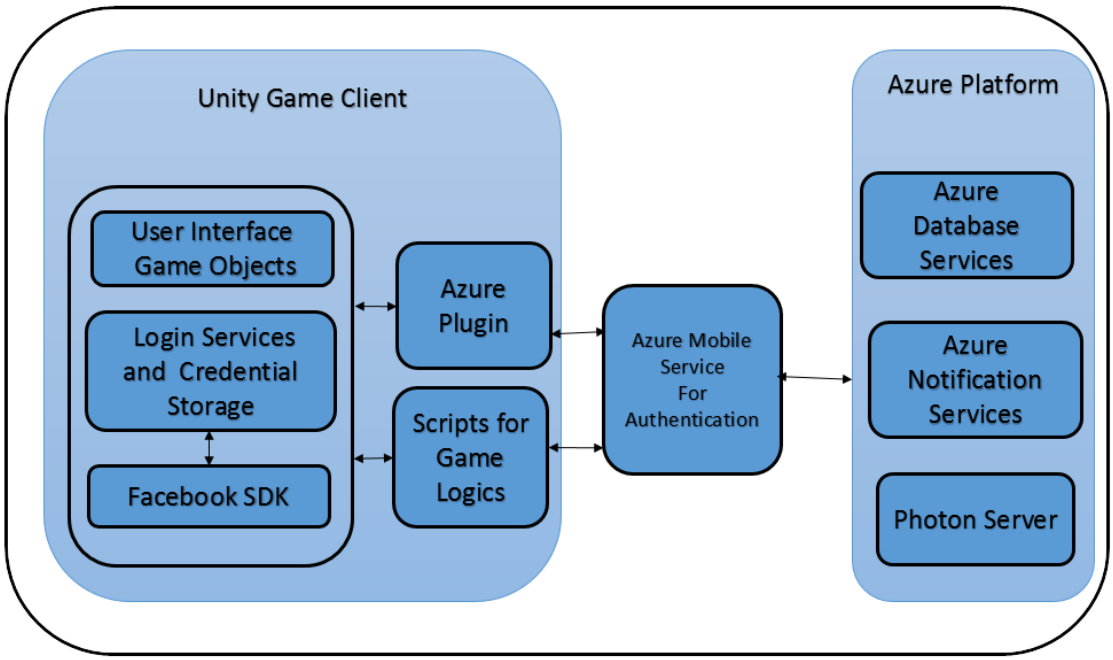


Figure 23 Overview of Usage of Azure Mobile Service for Unity games

In Figure 23 the overview of the game understudy is detailed. Game client is built using the Unity Technologies and the Server side is implemented using the Photon platform, The Server applications are run in Azure platform, the communication between the game client and the game server needs authentication, which is done using the Azure Mobile service. The game uses the Unity editor for development and it uses the Facebook SDK for the login and using the Facebook token the authentication to the azure platform could be implemented as a plugin, similar to the one explained in the session above with BI-trave sample.

Since the server side implementation was not under the scope of this study, the implementations needed for authentication to the server applications used in the Azure platform were implemented as prototype samples and demonstrated to the game team understudy, the team would make use of the prototypes to further implemented authentication to the server applications and integrate it to the game understudy.

4.3 Summary

Of the various methods discussed in the previous sub sections, the first method of using SIP server for authentication is not viable as there is no good support for integrating it into the Unity tool. The complete implementation of SIP login and networking should be implemented from scratch which would mean a huge amount of work for the study. The other method is based on the Photon support for Unity gaming, which is very a good and proven solution.

There are several ways to use the photon framework. Photon's PUN could be used for implementing a massive multiplayer online game, but it does not have an authentication module integrated, even though it offers very good features for Massive Multiplayer Online games. Currently, as the social networking sites are the driving factors for game applications, login services integrated to the social networking account are preferred to get more users to be involved in the game understudy. To use the authentication with social networking accounts could be implemented using the authentication services provided in the Azure platform. Azure mobile services explained in the previous section is very well suited for a secure login and it also includes needed networking features. It would serve as a good means to the requirements of the game understudy. The Azure platforms PASK methods could also be used with ACS authentication services. For this ACS service package needs to be developed using visual studio and made available for

using it with the Unity tool. Game developers also needs to write scripts in Unity for implementing the ACS configuration and certificates and then probably access it from the game client scripts using the RestSharp services. This need to be explored in future.

5 Further Development

In the study the implementation for getting the Facebook token and storing the token in a secure way was experimented. This token needs to be passed on to the Azure platform's Authentication system to get access to the server applications. The ways for passing the token to the server application in the Azure platform were also explored in the study by using the prototype samples. The study gives the details of various authentication methods used in Azure platform and in particular gives details of how the Azure mobile services could be used to authenticate the game users in the game understudy.

The study also analysed other Authentication methods for a multiplayer online game. This includes the usage of IMS framework. The study shows that IMS protocols have not been experimented before with the Unity games development framework. Unity has a very good editor and tool for designing graphics for the game, and it has inbuilt networking facilities that are easily usable for a multiplayer game. Using IMS is not recommended or is of no interest among the game developers in the game team understudy. Since photon framework already supports the networking features such as Chat, Load balancing, Lobby and MMO, which are more compatible with the Unity Networking components, developing the IMS related login and Networking is a completely new approach, which would take a long time. The study shows, however, that the IMS framework has several features and benefits.

Currently the missing feature in the platforms that are prominently used in game development is that to measure the data usage only for playing a game. The game data usage cannot be measured by the service providers in many of the platforms currently popular among game developers. So using the IMS would enable the data usage measurements, restricting and billing for the usage of data network charges.

From the study in the Vulpine Games team it is understood that Unity with the Azure and the Photon platforms is very well suited for the multiplayer game development, they provide very secure means of authentication and login services, also they provide easy to use solutions at affordable cost for creating games, and also provide support to run the game in multiple platforms. Also Unity, Photon and Azure services provide very good support for game developers and are constantly improving their services, but these platforms do not provide the convergence for the subscription data and media functions, which is possible with IMS. It would be good if these platforms also supported the IMS

functionalities, so that billing and data usage measurements could be possible for gaming as well. Hence this research could be extended to study how the IMS platform features could be integrated to tools like Unity, to further research on how the IMS features and capabilities could be best utilised for better monitoring the gaming time and data usage. Hence designing a server using IMS framework for Unity and making it available as a simple package to be used in a Unity game project would be an interesting project to be development in future. Currently Photon Framework had very stable and advanced services for server side requirements for games developed using Unity, so the Vulpine Gams team would implement their game using the Azure and Photon platforms, by making use of the this study.

REFERENCES

[1] WWW document http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[2] WWW document http://plato.stanford.edu/entries/game-theory/

[3] WWW document http://www.thelearningpoint.net/home/mathematics/a-totorial-on-extensive-games-with-problems-and-solutions

[4] Hirematada, Prashanth. Flash 10 Multiplayer Game Essentials: Create Exciting Real-Time Multiplayer Games Using Flash. Olton, Birmingham, GBR: Pack t Publishing, 2010.

[5] WWW document http://research.microsoft.com/en-us/um/redmond/events/cloud-futures2012/monday/Interactive_Services_Interactive3D_Lucas_Kencl.pdf. “Interactive 3D Services over Windows Azure” Danihelka, Kencl, Czech Technical University in Prague

[6] WWW document http://www.riaxe.com/blog/top-cross-platform-mobile-development-tools/

[7] WWW document https://www.websocket.org/aboutwebsocket.html

[8] WWW document http://channel9.msdn.com/Series/Microsoft-Azure-Back-End-for-Gaming

[9] WWW document https://msdn.microsoft.com/en-us/magazine/dn532200.aspx

[10] WWW document https://technet.microsoft.com/en-us/video/access-control-service-acs-and-the-cloud

[11] WWW document https://doc.photonengine.com/en/onpremise/current/getting-started/photon-server-intro

[12] WWW document http://www.paladinstudios.com/2013/07/10/how-to-create-an-online-multiplayer-game-with-unity/

[13] WWW document http://docs.unity3d.com/Manual/net-HighLevelOverview.html

[14] WWW document http://www.radio-electronics.com/info/telecommunications_networks/ims-ip-multimedia-subsystem/ims-layers-stack.php

[15] WWW document http://www.ims-way.com/about/what-is-ims/architecture-ims/

[16] WWW document http://www.radio-electronics.com/info/telecommunications_networks/ims-ip-multimedia-subsystem/ims-architecture.php

[17] Hersent, Olivier. IP Telephony: Deploying VoIP Protocols and IMS Infrastructure (2nd Edition). Hoboken, NJ, USA: John Wiley & Sons, 2010.

[18] Khartabil, Hisham, Mayer, George, and Niemi, Aki. IMS : IP Multimedia Concepts and Services in the Mobile Domain. Hoboken, NJ, USA: Wiley, 2004.

[19] Johnston, Alan B., and Schutzer, Daniel. SIP : Understanding the Session Initiation Protocol (3rd Edition). Norwood, MA, USA: Artech House, 2009.

[20] WWW document http://nexosims.com/nexos.pdf

[21] Amjad Akkawi, Sibylle Schaller, Oliver Wellnitz, and Lars Wolf. 2004. A mobile gaming platform for the IMS. In Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games (NetGames '04). Pages 77-84 ACM New York, NY, USA ©2004 ISBN: 1-58113-942-X

[22] WWW document http://www.kamailio.org

[23] WWW document http://www.kamailio.org/dokuwiki/doku.php/install:kamailio-3.1.x-from-git#install_and_maintain_kamailio_openser_v31x_from_git

[24] WWW document http://docs.unity3d.com/ScriptReference/PlayerPrefs.html

[25] WWW document http://www.previewlabs.com/writing-playerprefs-fast/

[26] WWW document http://www.paladinstudios.com/2014/05/08/how-to-create-an-online-multiplayer-game-with-photon-unity-networking/

[27] WWW document http://blogs.msdn.com/b/cdndevs/archive/2014/12/11/part-4-azure-mobile-services-what-you-need-to-know-about-authentication-and-authorization.aspx

[28] WWW document http://www.deadlyfingers.net/azure/unity3d-game-dev-with-azure-mobile-services-using-bitrave-plugin/

[29] WWW document http://ubuntuserverhelp.com/asterisk-sip-conf/

[30] Felicia, Patrick. Getting Started with Unity. Olton, Birmingham, GBR: Packt Publishing, 2013.

[31] WWW document https://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/

[32] WWW document http://doc.exitgames.com/en/onpremise/current/reference/photon-azure-starter-kit-pask

[33] Dempster, Barrie Gomillion, David Merel, David, "Build Feature-Rich Telephony Systems with Asterisk", Packt Publishing Ltd: Olton Birmingham, 2009.

[34] WWW document http://docs.unity3d.com/Manual/PluginsForDesktop.html

[35] Mohov, Sergey. Practical Game Design with Unity and Playmaker. Olton, Birmingham, GBR: Packt Publishing Ltd, 2012.

[36] WWW document https://developers.facebook.com/docs/unity

[37] WWW document http://www.richcommunicationsuite.com/index.htm

APPENDICES

Appendix 1. Instructions for SIP Setup in simple methods [29].

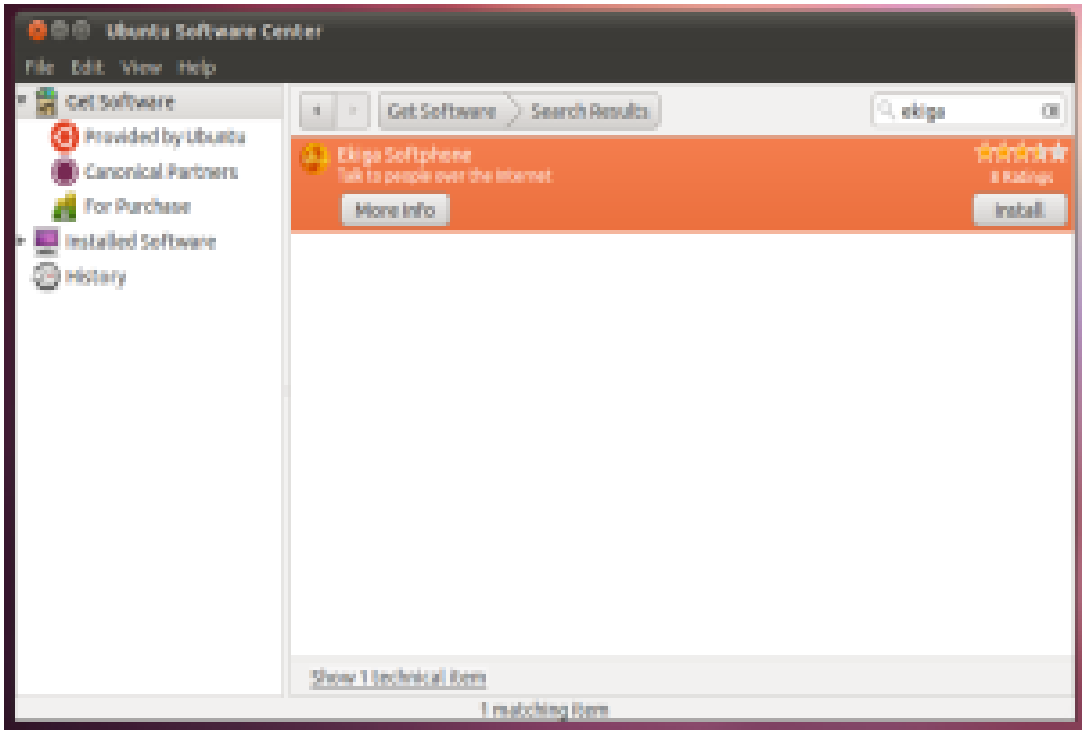
Steps	Description of Step	Instruction
Step 1	• Install server	• *apt-get install asterisk*
Step 2	• Setup Server Configuration File (sip.conf) , the details of variables in the file contents and their value are as below **type:** Defines the role the extension will have, and we have three options: - peer: Where the device will only be able to receive calls from the PBX - user: Where the device will only be able to make calls to the PBX - friend: Where the device will be able to both make and receive calls through the PBX **nat:** Defines, if the connecting device that will use that profile, is or is not behind nat. **username:** Defines the username to be used to connect to the PBX **secret:** Is the password to be able to connect to the PBX **host:** The IP that the device will have, if it is set to dynamic, that device will be able to log into the PBX from any IP **default:** This is very important and defines the group at which that profile belongs, and it is used in the /etc/asterisk/extensions.conf file.	• Add the below contents to /etc/asterisk/sip.conf *[5010]* *type=friend* *username=5010* *secret=987123* *host=dynamic* *context=default* *[5020]* *type=friend* *username=5020* *secret=987123* *host=dynamic* *context=default*
Step 4	• Setup the extension details in configuration file extensions.conf	• Add the below contents to /etc/asterisk/extenstions.conf *[default]* *exten => 5010,1,Dial(SIP/5010)* *exten => 5020,1,Dial(SIP/5020)*
Step 5	• Install SIP Client in two PCs .	• Install Ekiga using apt-get command *apt-get install ekiga*
Step 6	• Verify Server Setup	• Start the asterisk server using the CLI command *asterisk –start* • Make a SIP Call using the extension (5010) from one of the clients • Verify that the call is established

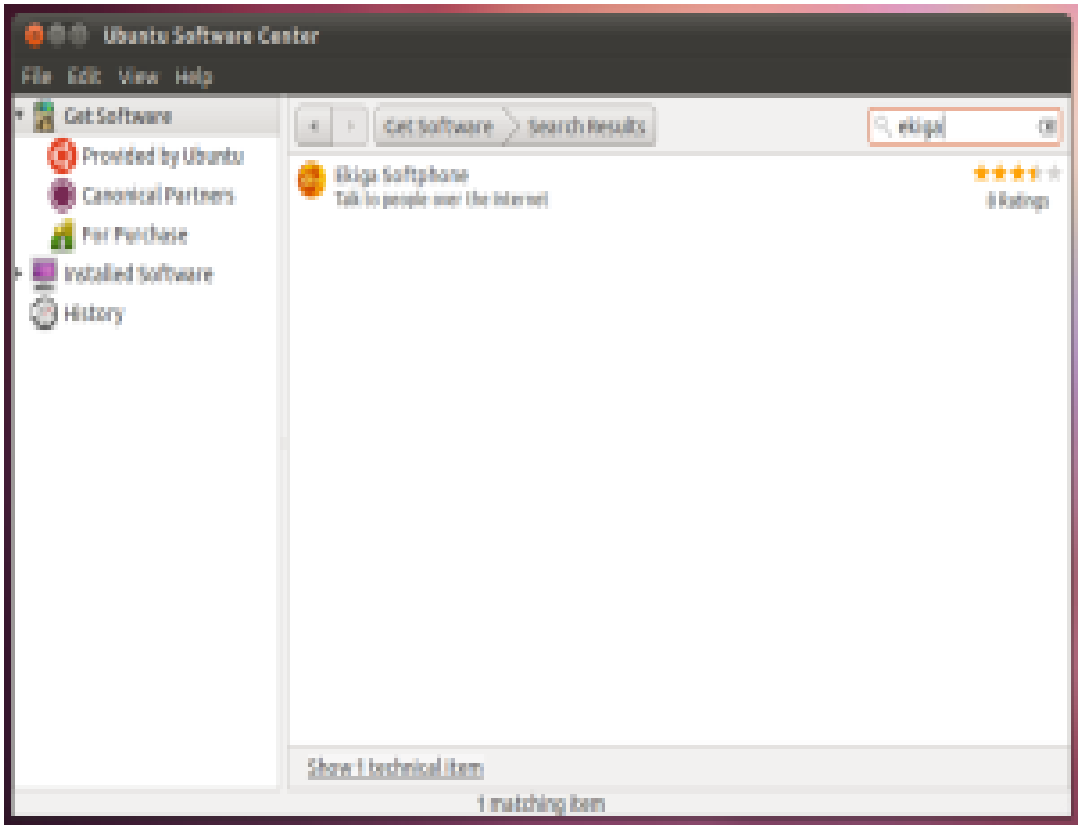
		• Check the server logs of the sessions the server hosts currently , using the command *asterisk –rvvvv* • Stop the server using the CLI command *asterisk –shutdown* • Check the server logs of the sessions using the command belwo , there should not be any logs *asterisk –rvvvv*
Step 7	• Verify Client Setup	• Make call between the clients ,using the extension numbers instead of SIP URIs,Try messaging between the two clients.

Appendix 2. SIP Client SetUp.

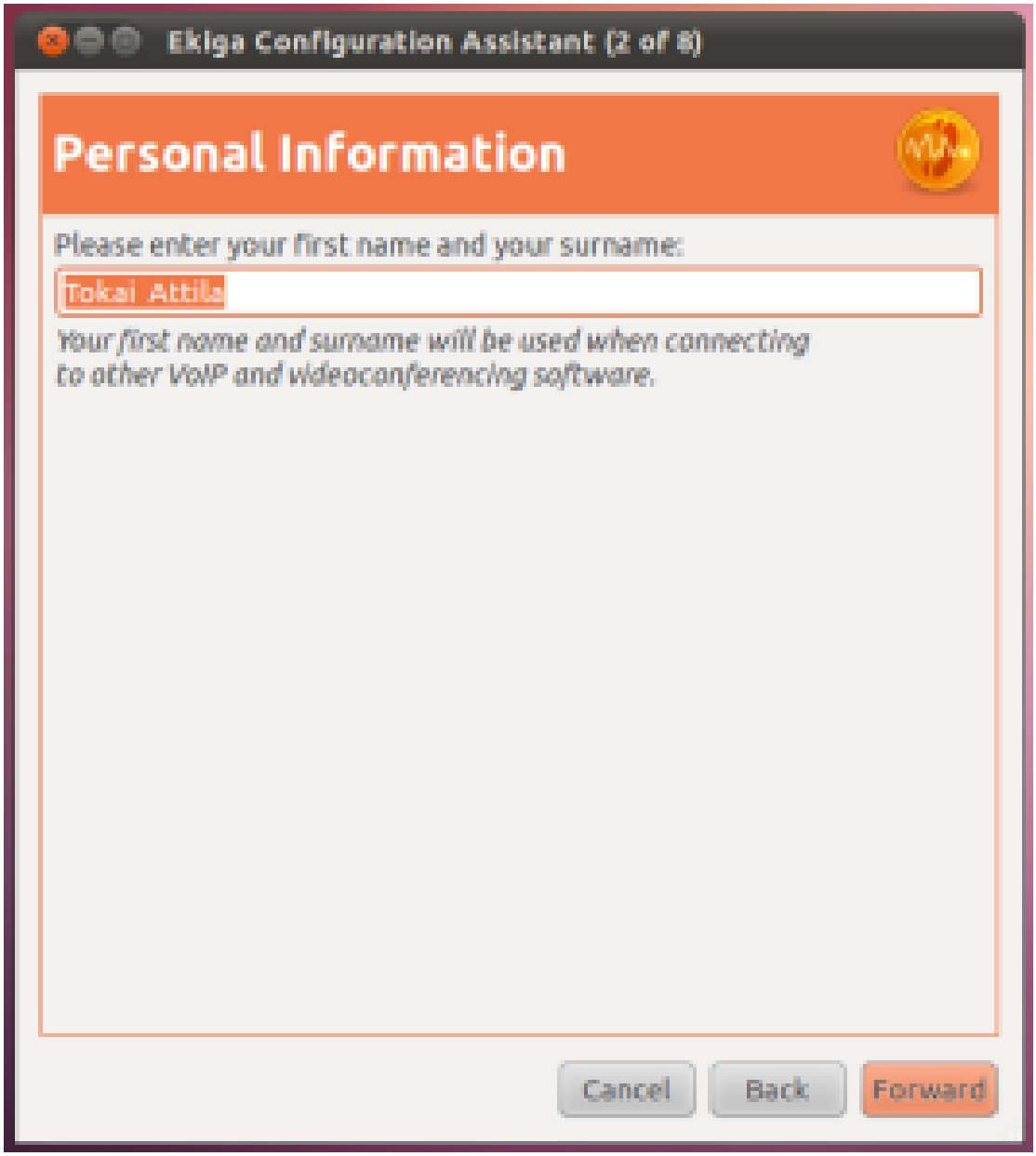
Step 1. Install the Ekiga client software from Ubuntu Software Center.

Open the Ubuntu Software Center Application form the System File menu and search for Ekiga. Then click the Ekiga software installer option and give the Password for authentication to install the software, the screen shots for these steps are below.


Ubuntu Software Center
File Edit View Help
Get Software
Provided by Ubuntu
Canonical Partners
For Purchase
Installed Software
History
Get Software
Search Results
ekiga
Ekiga Softphone
More Info
Install
Show 1 technical item
1 matching item

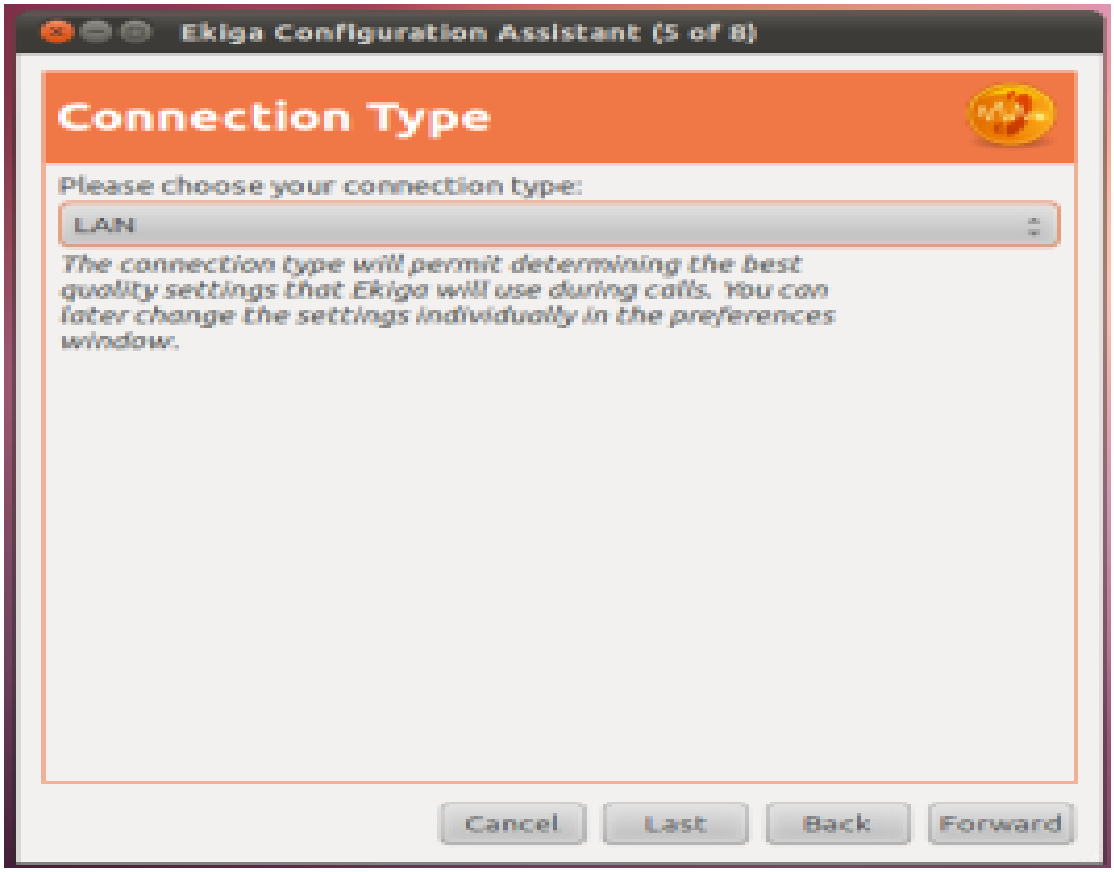

Ubuntu Software Center
File Edit View Help
Get Software
Provided by Ubuntu
Canonical Partners
For Purchase
Installed Software
History
Get Software
Search Results
ekiga
Ekiga Softphone
Show 1 technical item
1 matching item

Step 2. Launch Ekiga from the Applications -> Internet Menu and follow the initial setup dialogs, just by clicking the forward button.


Ekiga Configuration Assistant (2 of 8)
Personal Information
Please enter your first name and your surname:
Tokai Attila
Your first name and surname will be used when connecting
to other VoIP and videoconferencing software.
Cancel
Back
Forward

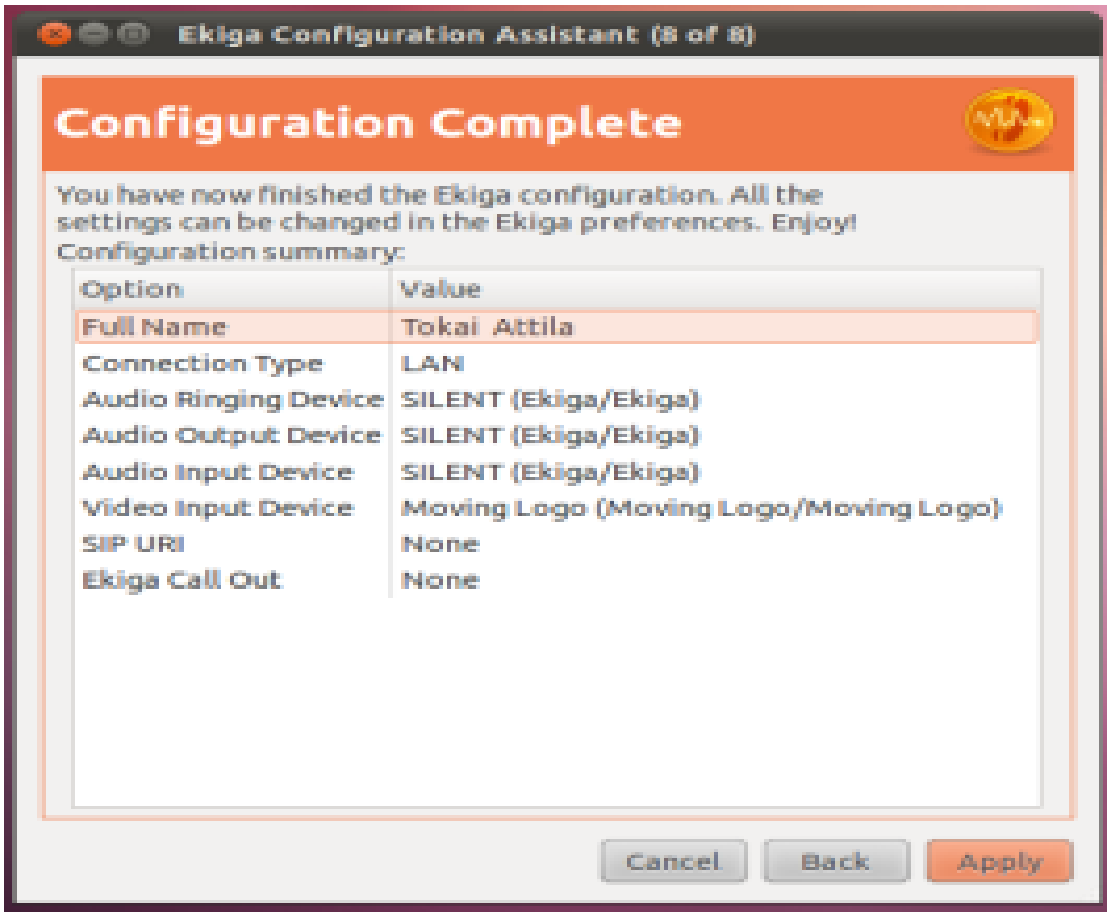

Ekiga Configuration Assistant (3 of 8)
Ekiga.net Account
Please enter your username:
Please enter your password:
The username and password are used to login to your existing account at the ekiga.net free SIP service. If you do not have an ekiga.net SIP address yet, you may first create an account below. This will provide a SIP address that allows people to call you.
You may skip this step if you use an alternative SIP service, or if you would prefer to specify the login details later.
Get an Ekiga.net SIP account
I do not want to sign up for the ekiga.net free service
Cancel
Back
Forward

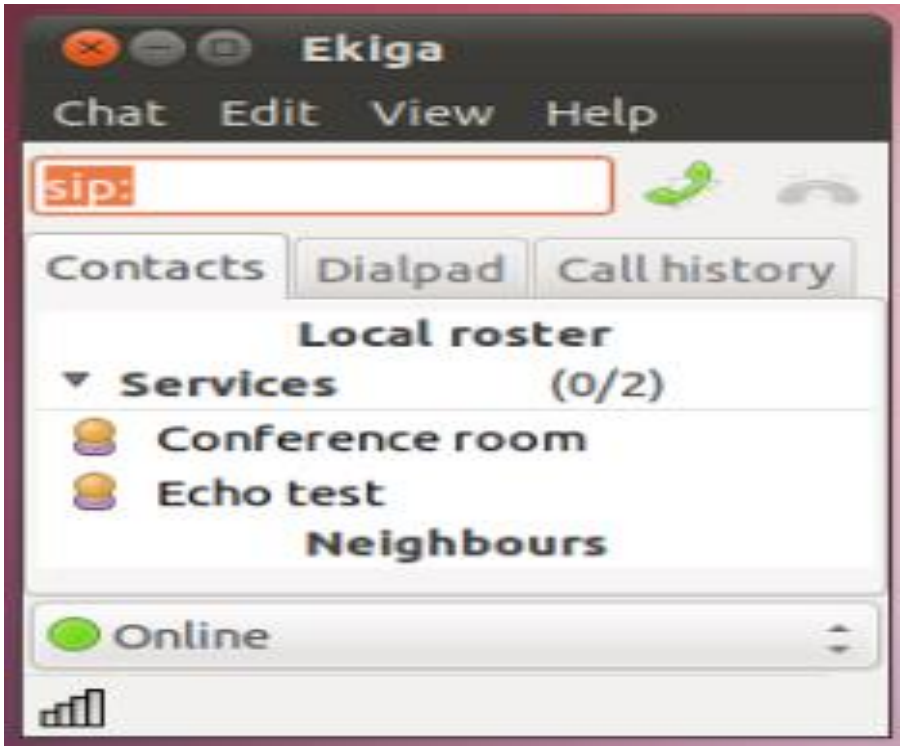

Ekiga Configuration Assistant (4 of 8)
Ekiga Call Out Account
Please enter your account ID:
Please enter your PIN code:
You can make calls to regular phones and cell numbers worldwide using Ekiga.
To enable this, you need to do two things:
- First buy an account at the URL below.
- Then enter your account ID and PIN code.
The service will work only if your account is created using the URL in this dialog.
Get an Ekiga Call Out account
Recharge the account
Consult the balance history
Consult the calls history
I do not want to sign up for the Ekiga Call Out service
Cancel
Last
Back
Forward


Ekiga Configuration Assistant (5 of 8)
Connection Type
Please choose your connection type:
LAN
The connection type will permit determining the best quality settings that Ekiga will use during calls. You can later change the settings individually in the preferences window.
Cancel
Last
Back
Forward

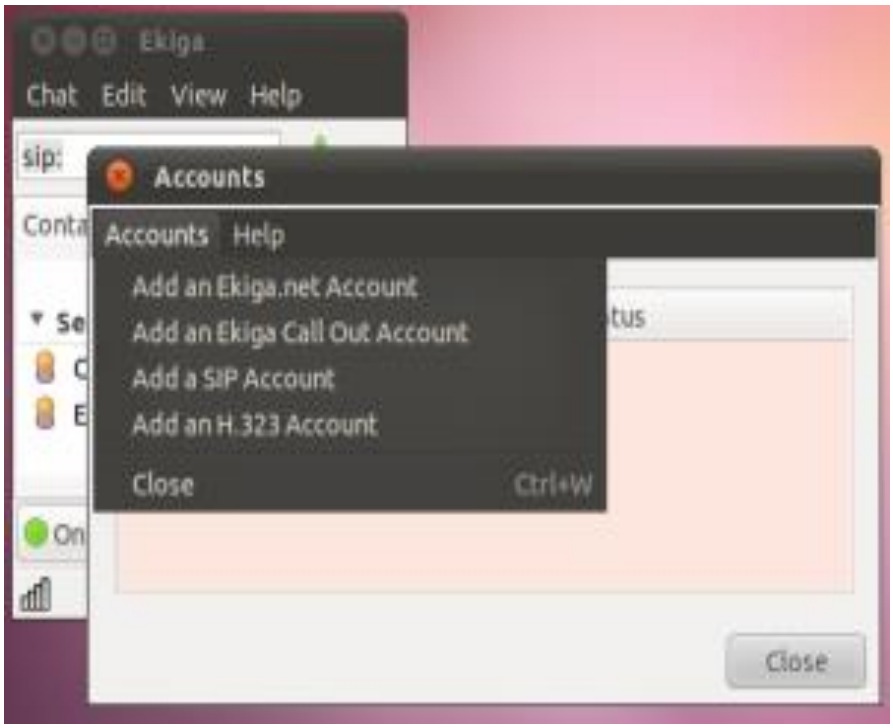

Ekiga Configuration Assistant (6 of 8)
Audio Devices
Please choose the audio ringing device:
SILENT (Ekiga/Ekiga)
The audio ringing device is the device that will be used to play the ringing sound on incoming calls.
Please choose the audio output device:
SILENT (Ekiga/Ekiga)
The audio output device is the device that will be used to play audio during calls.
Please choose the audio input device:
SILENT (Ekiga/Ekiga)
The audio input device is the device that will be used to record your voice during calls.
Cancel
Last
Back
Forward

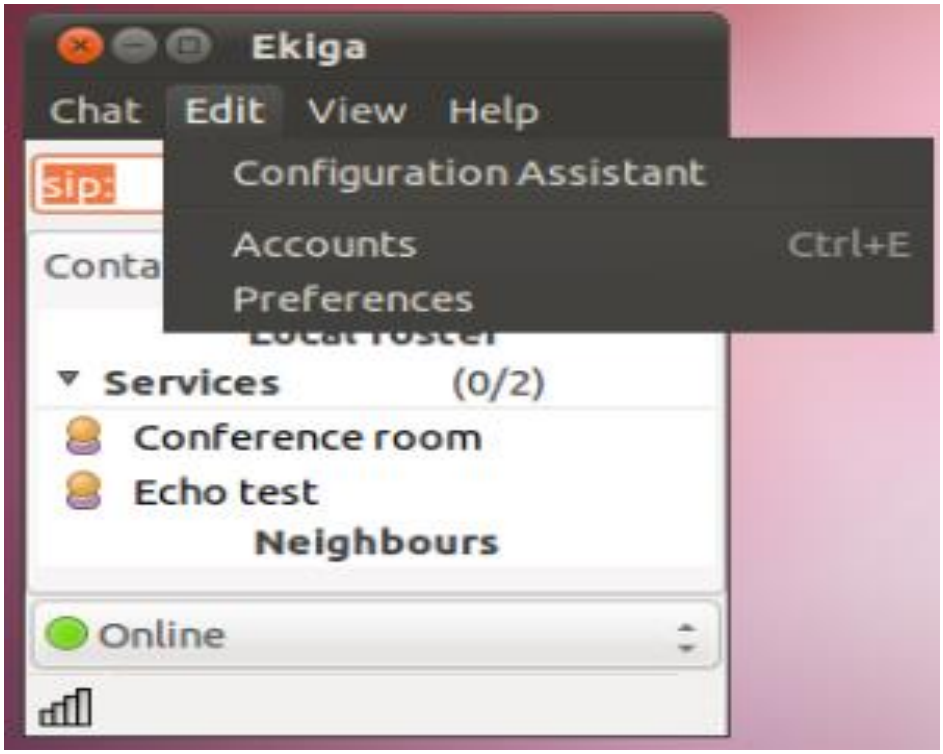

Ekiga Configuration Assistant (7 of 8)
Video input device
Please choose your video input device:
Moving Logo (Moving Logo/Moving Logo)
The video input device is the device that will be used to capture video during calls.
Cancel
Back
Forward

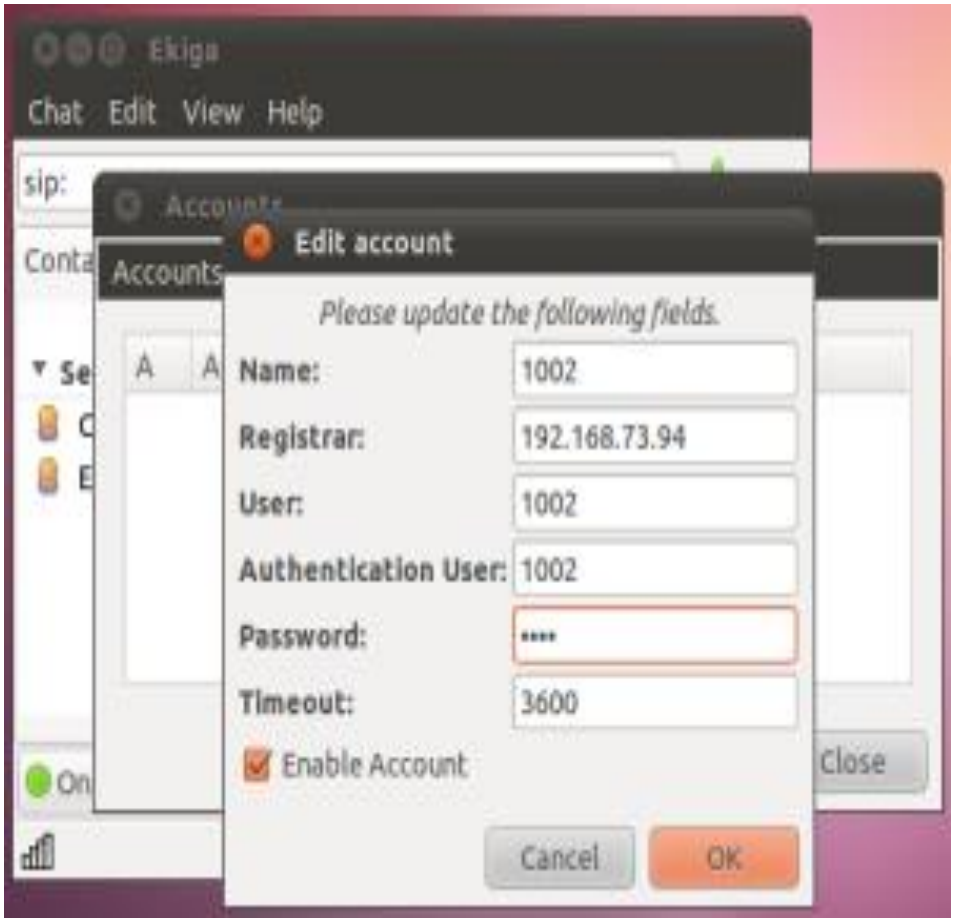

Ekiga Configuration Assistant (8 of 8)
Configuration Complete
You have now finished the Ekiga configuration. All the settings can be changed in the Ekiga preferences. Enjoy!
Configuration summary:
Option
Value
Full Name
Tokai Attila
Connection Type
LAN
Audio Ringing Device
SILENT (Ekiga/Ekiga)
Audio Output Device
SILENT (Ekiga/Ekiga)
Audio Input Device
SILENT (Ekiga/Ekiga)
Video Input Device
Moving Logo (Moving Logo/Moving Logo)
SIP URI
None
Ekiga Call Out
None
Cancel
Back
Apply



Step 3: Setup the SIP account details as mentioned in the screen shots below.

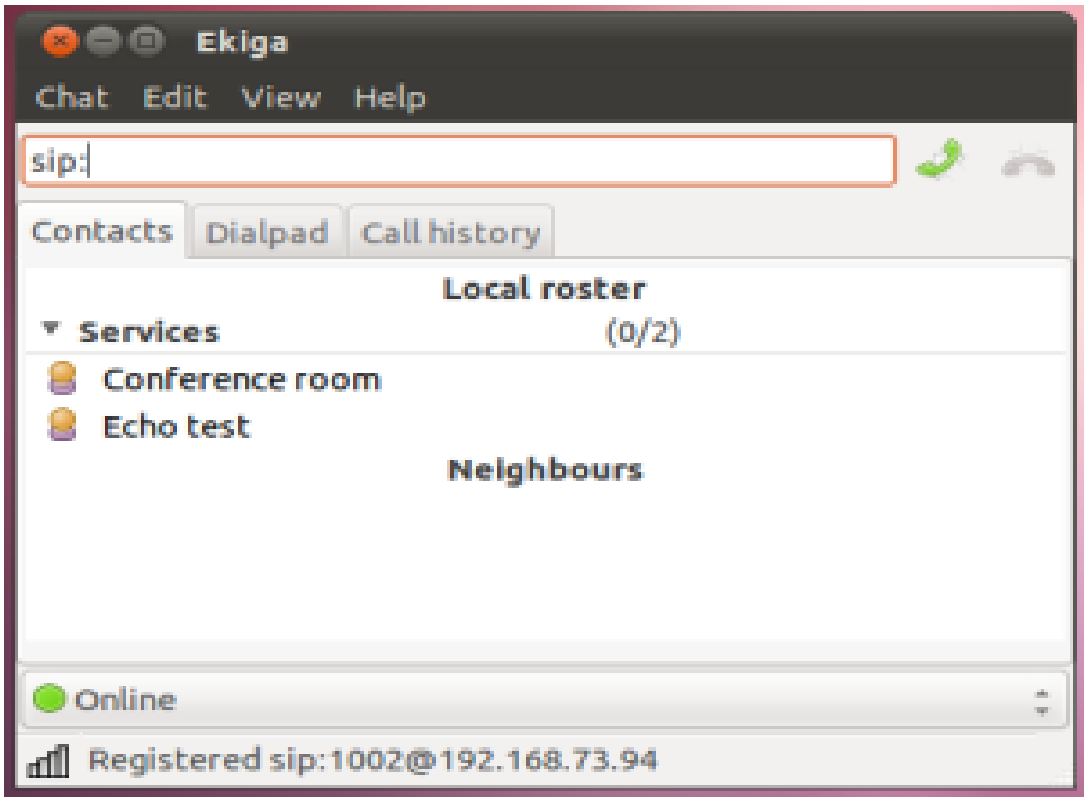


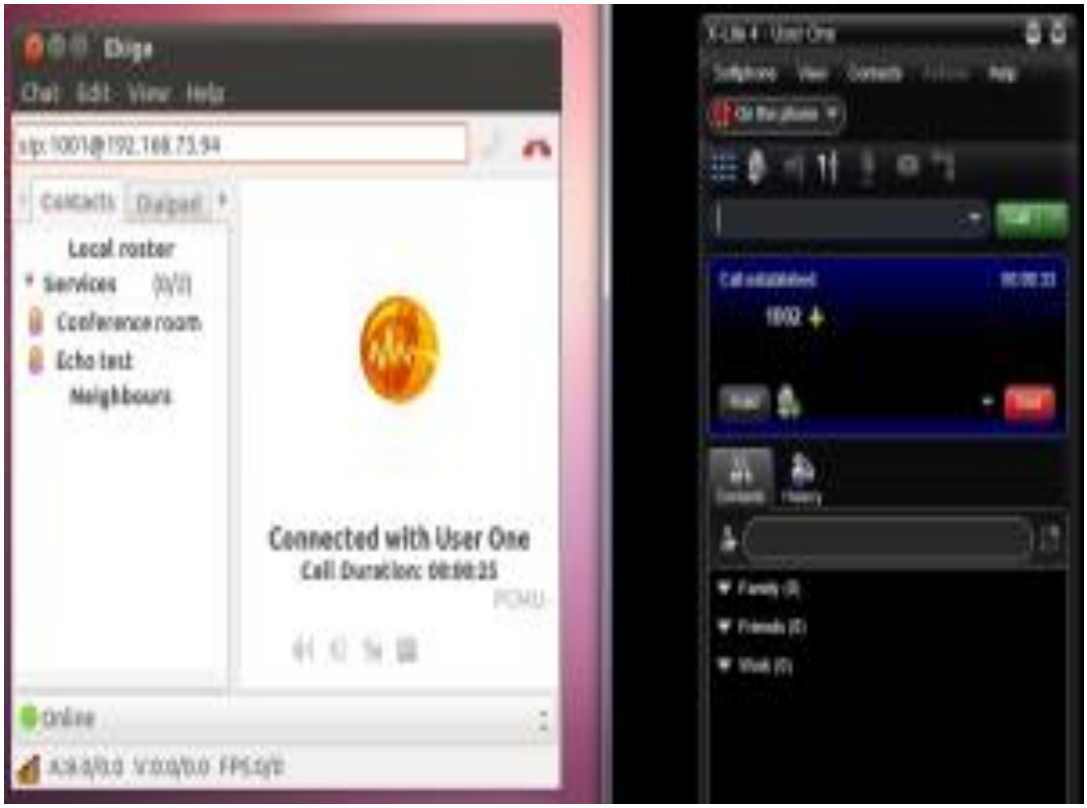




Step 4. Type the user's SIP URI created in the Asterisk server ,in the format sip:username@(IP Adddress of Asterisk Server) ,example sip:1002@192.168.73.94 .

Repeat step 2-3 for In another client machine and set it up with the with different sip address sip:1001@192.168.73.94 ,now the SIP call could be extablished between the two ekiga cleints.





Step 5: Check the setup using the asterisk –rvv command and the view the connection details as below.

```
    -- Registered SIP '1002' at 192.168.73.143:5060
       > Saved useragent "Ekiga/3.2.7" for peer 1002
pbx*CLI> sip show peers
Name/username              Host                                    Dyn Forcerpor
t ACL Port     Status
1001                       (Unspecified)                            D          0
       Unmonitored
1002/1002                  192.168.73.143                           D          5
060    Unmonitored
OCS_SIP_TRUNK              192.168.73.95                                       5
060    UNREACHABLE
3 sip peers [Monitored: 0 online, 1 offline Unmonitored: 1 online, 1 offline]
pbx*CLI> _
```

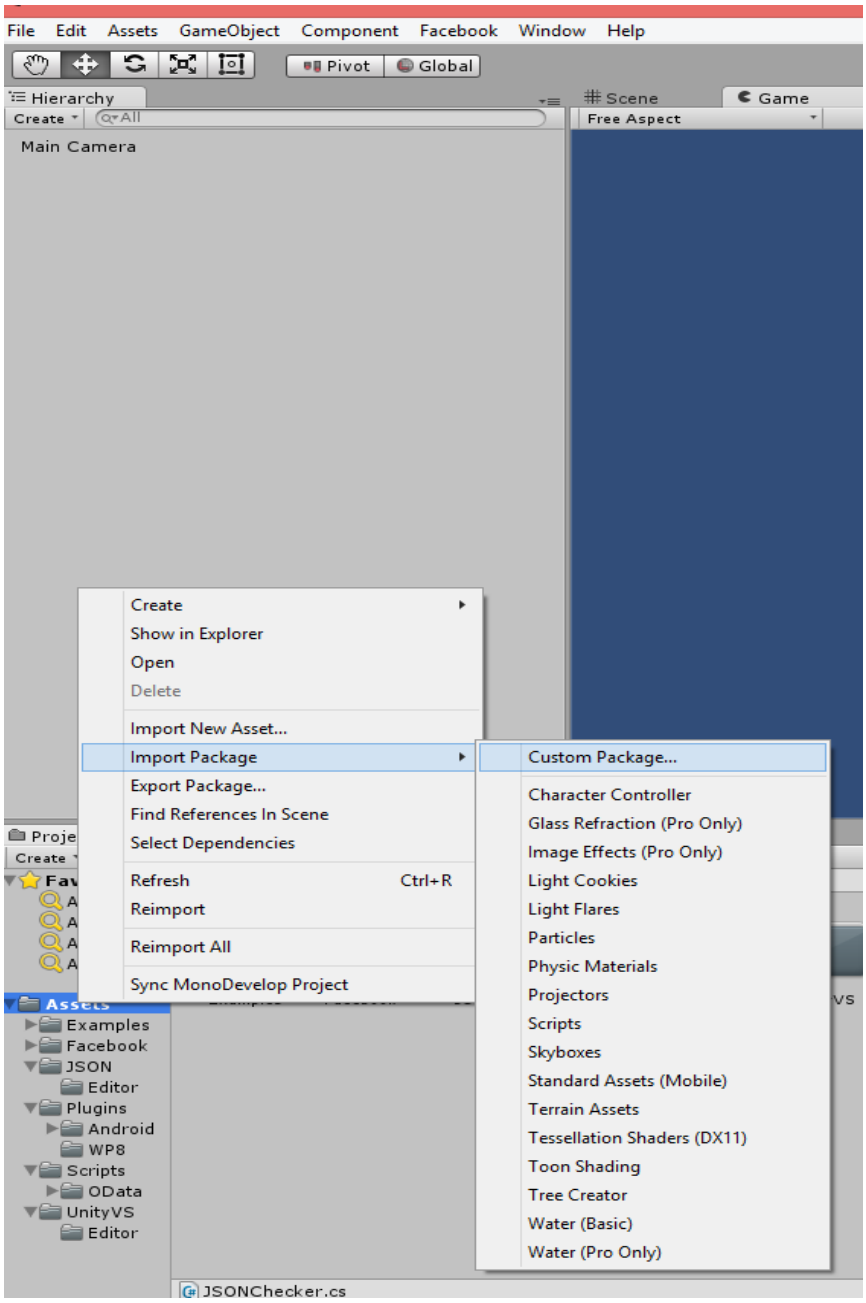
Appendix 3. Steps for Facebook SDK integration to Unity Game.

Step 1.Login to thefacebook developer's website and download latest version of Facebook SDK . https://developers.facebook.com/docs/unity/downloads

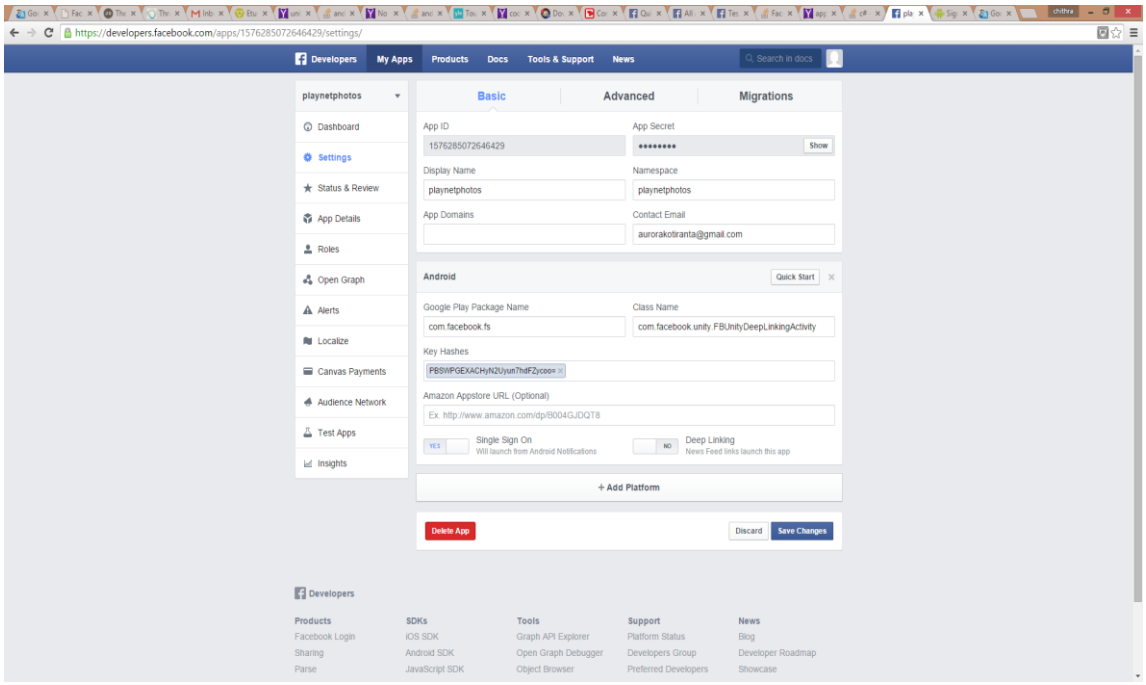
Step 2.Create the game project in Unity

Step 3.Install latest version of Facebook SDK to the project in the unity project ,with steps below.Use the file you downloaded in step 1

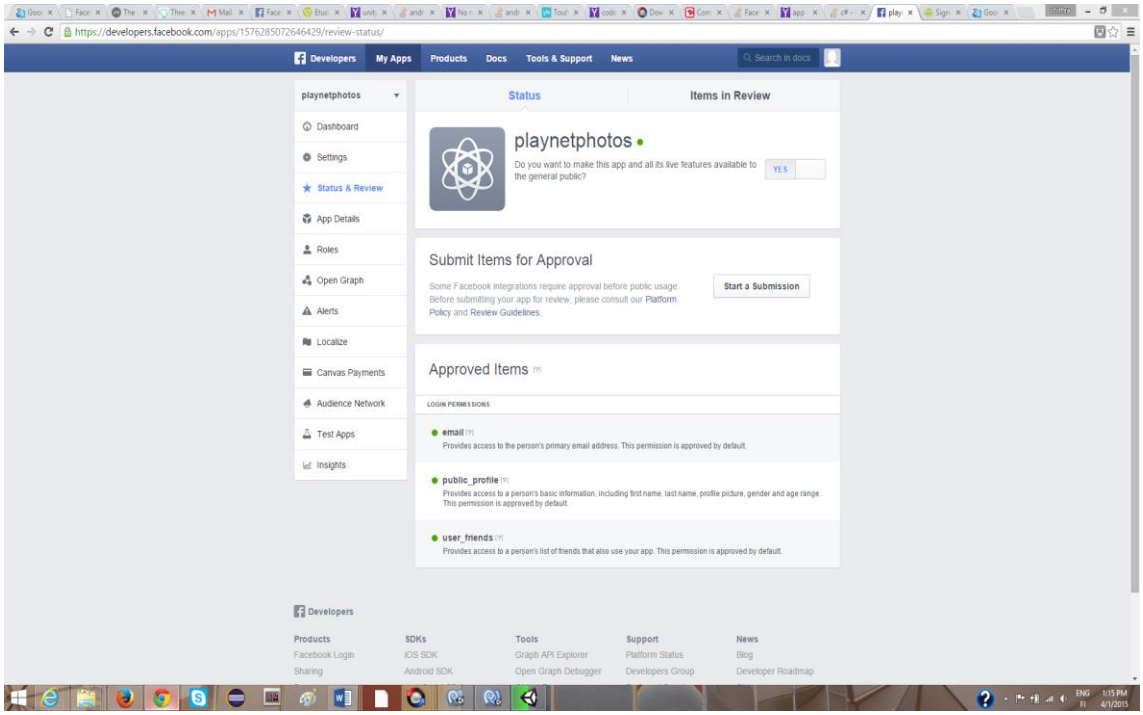
Install Facebook Unity Custom Packge , from Assets →(right Click) -> Custom Package



Step 4. Create and Setup the Facebook Applicaiton, in Facebook developer website

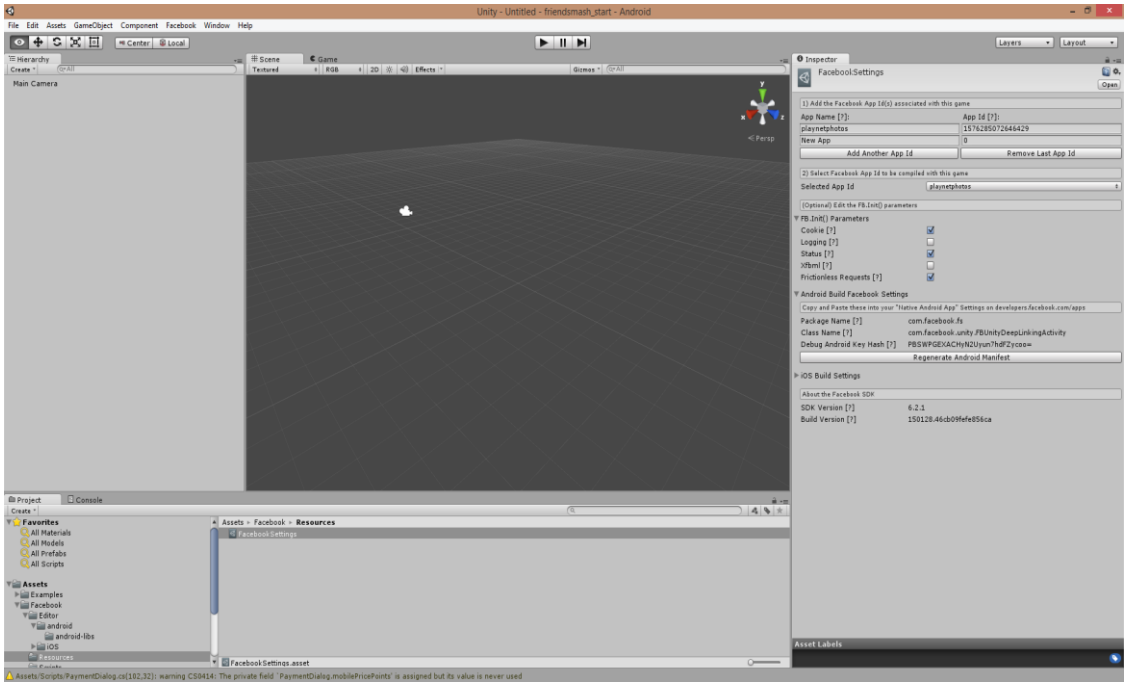


Step 5. Make the status of the App to be available public .



Step 5. Setup Unity project settings for facebook app .

Select Facebook → Edit Settings, from the menu bar,copy the facebook app ID and other details required for setup from the above steps 3 and 4 ,and fill the details in the Edit tab fields .



Step 4.

Make sure that the Open SSL and JDK are installed on the PC and the path these libraries setupare correctly in the environment varailbles setting as “path” in control panels.If not installed, download and install the OpenSSL and JDK libraries and set the installation path in the control panel.

Step 5.

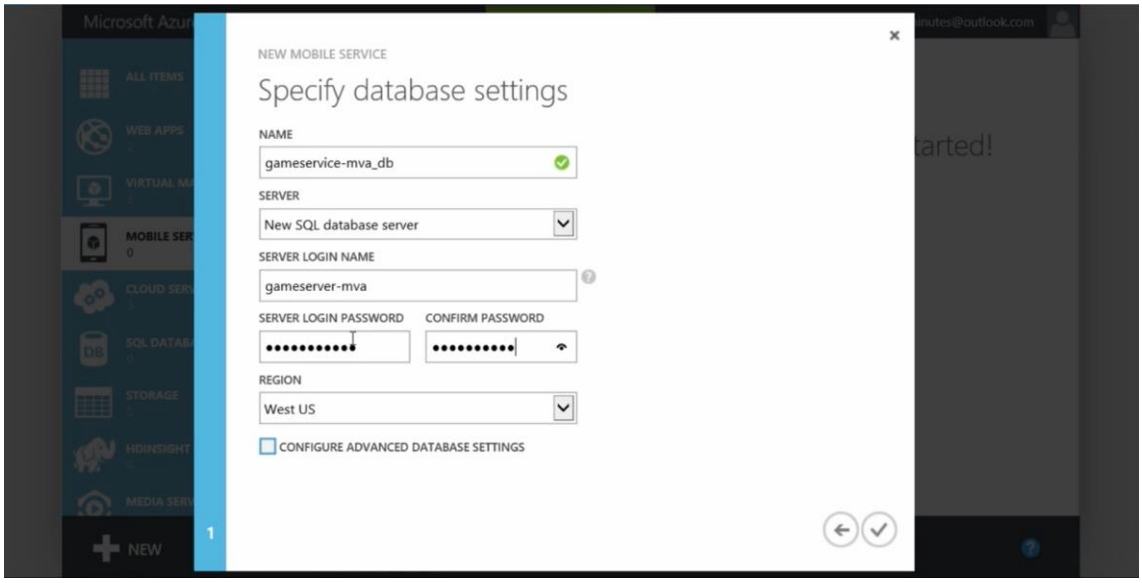
Create a Login Button and Game Object, in the game object script, add reference to the Facebook SDK and make calls for the Facebook Login API, and link the game object with the login button.

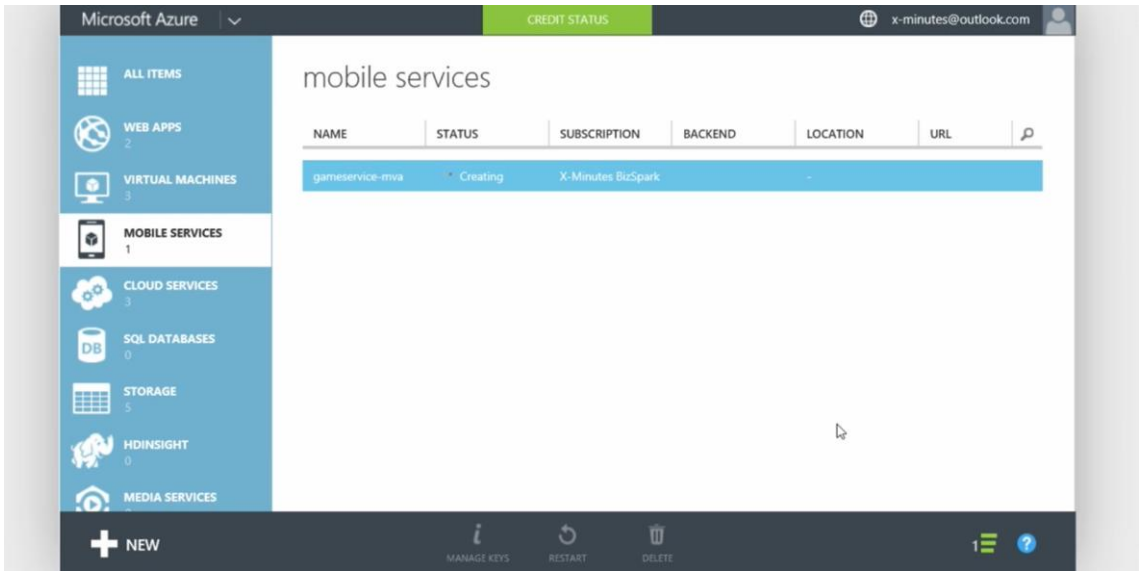
Step.6 Build and Run the project from the File menu. Select the Android OS option to try the sample project in the Android device. Make sure the device is set to developer mode before running on the device.

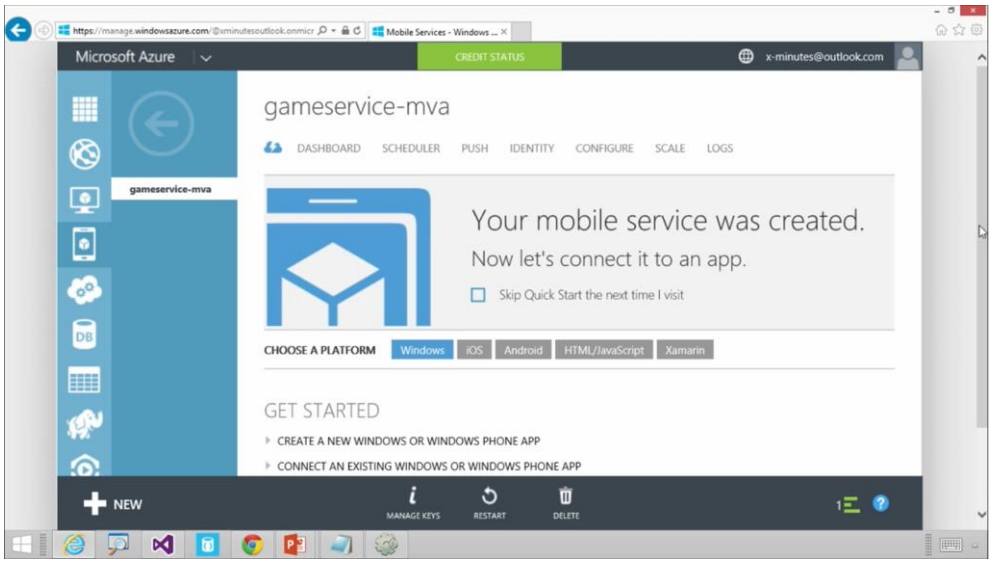
Appendix 4. Creating Azure Mobile Service

Azure mobile service need to be created to use it with the clients like unity projects. The steps to create a mobile service at the Azure website are below.

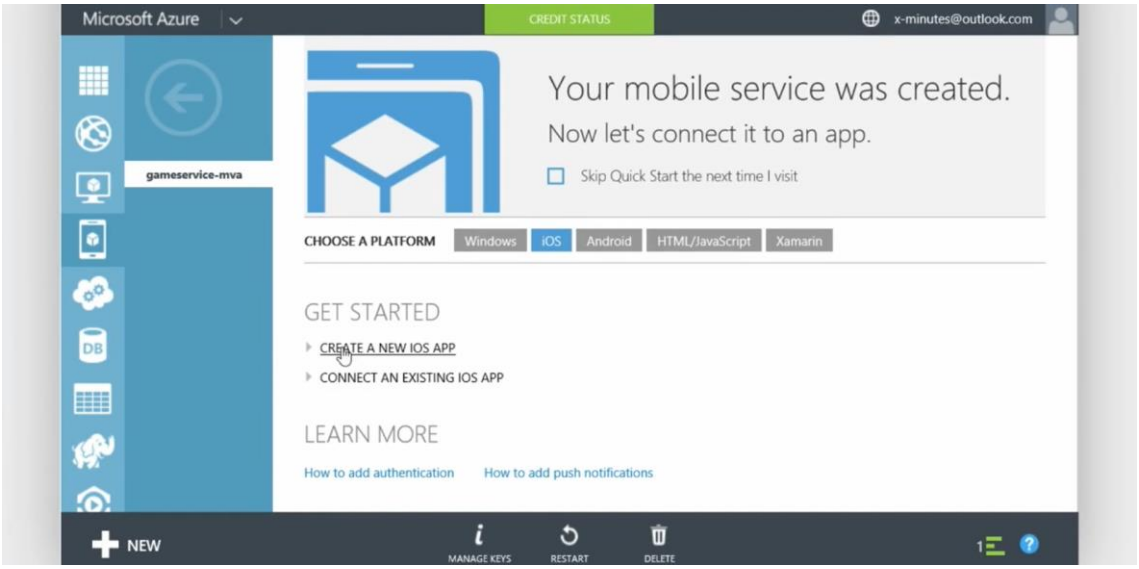
Step 1. Signup for Azure Services and login with the Microsoft live account at https://manage.windowsazure.com and select Mobile services Option. And Create a Database service, the screen shots below gives the details to do the same.


NEW MOBILE SERVICE
Specify database settings
NAME
gameservice-mva_db
SERVER
New SQL database server
SERVER LOGIN NAME
gameserver-mva
SERVER LOGIN PASSWORD
CONFIRM PASSWORD
REGION
West US
CONFIGURE ADVANCED DATABASE SETTINGS
NEW


Microsoft Azure
CREDIT STATUS
x-minutes@outlook.com
ALL ITEMS
WEB APPS
VIRTUAL MACHINES
MOBILE SERVICES
CLOUD SERVICES
SQL DATABASES
STORAGE
HDINSIGHT
MEDIA SERVICES
mobile services
NAME
STATUS
SUBSCRIPTION
BACKEND
LOCATION
URL
gameservice-mva
Creating
X-Minutes BizSpark
NEW
MANAGE KEYS
RESTART
DELETE

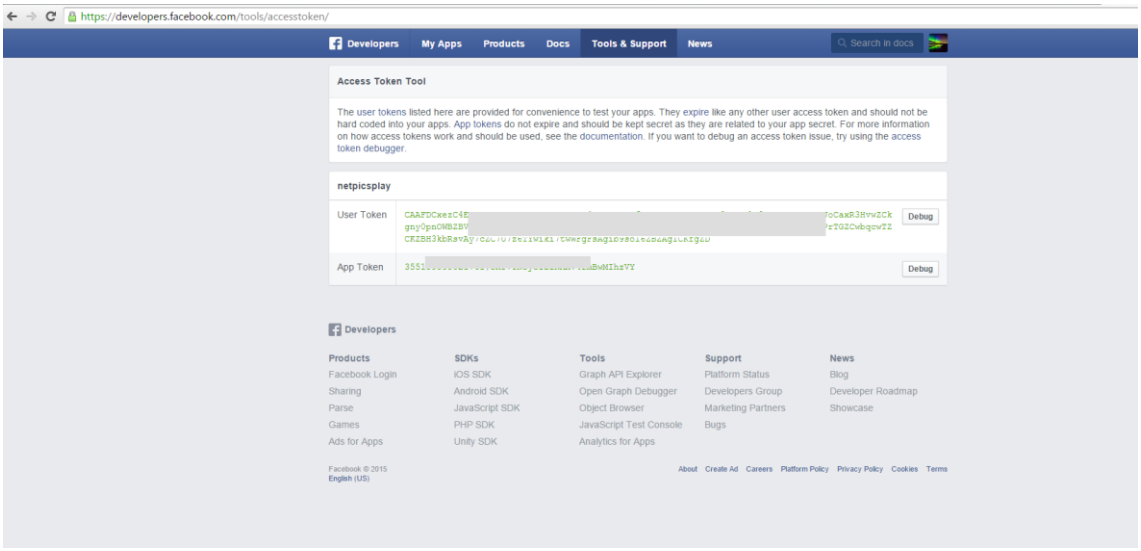

Microsoft Azure
CREDIT STATUS
x-minutes@outlook.com
gameservice-mva
DASHBOARD
SCHEDULER
PUSH
IDENTITY
CONFIGURE
SCALE
LOGS
Your mobile service was created.
Now let's connect it to an app.
Skip Quick Start the next time I visit
CHOOSE A PLATFORM
Windows
iOS
Android
HTML/JavaScript
Xamarin
GET STARTED
CREATE A NEW WINDOWS OR WINDOWS PHONE APP
CONNECT AN EXISTING WINDOWS OR WINDOWS PHONE APP
NEW
MANAGE KEYS
RESTART
DELETE

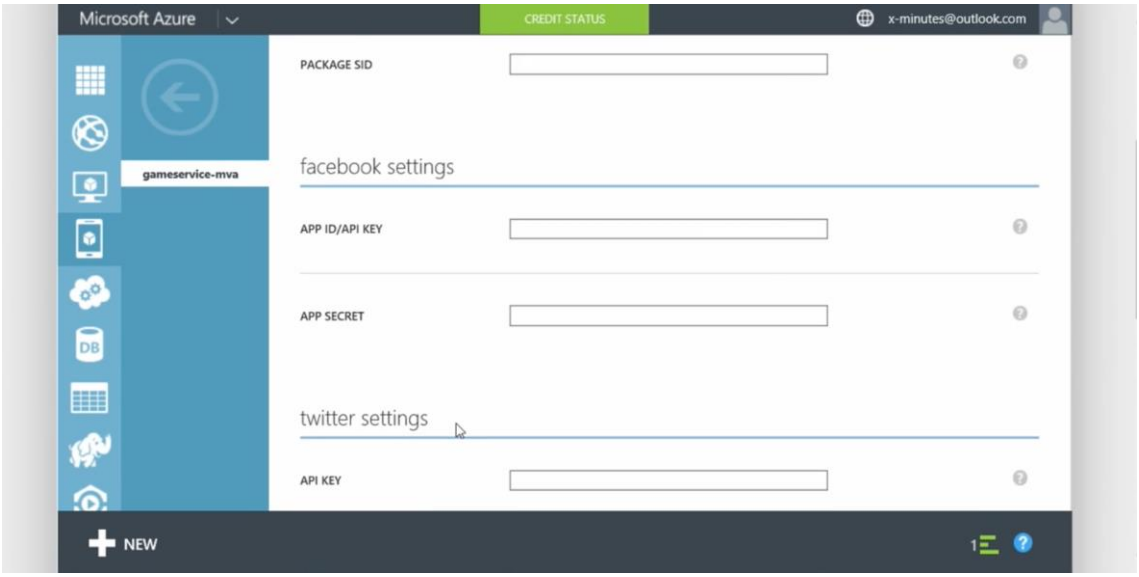
Step 2. After creating the Mobile Service , user can either create a new app option to get the client sample or use the Connect to existing App option to get the mobile app service link and ID to set it up for the already existing client.



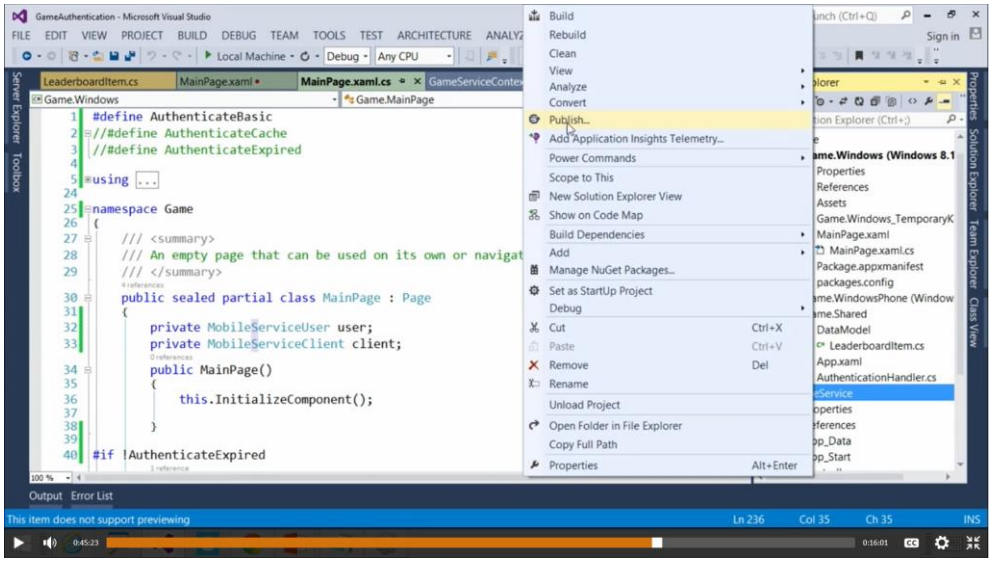
Step 3. Setup Facebook access tokens for using facebook login services.

Setup a facebook app in the developer.faceboook.com web site and copy the tokens and App ID ,as show in the fields below.





Step 4. Build the Azure Solutions and Publish the Game Services.



Appendix 5: Steps to try the Azure plugin (BitRave) sample with Unity

To use the Azure Mobile services with unity one need to write a plugin for unity project, and example for this plugin could be found from the link http://www.deadlyfingers.net/azure/unity3d-game-dev-with-azure-mobile-services-using-bitrave-plugin/ , the summary of steps explained in the link are below.

- Create a Mobile Service in Azure management portal. New users without an Azure account yet, can register for the Cloud GameDev Offer. A Mobile Service takes a minute to setup and just a couple of minutes more to become active and ready to use.
- Download BitRave plugin from the link https://github.com/bitrave/azure-mobile-services-for-unity3d.
- Create new Unity3D project.
- Copy BitRave's AzureMobileServicesUniversalPlugin/Assets into Unity3D project's 'Assets' folder.
- Get JSON.NET Unity asset to enable cross platform support.
- Open TestAzure Scene in the project.
- Open AzureUI.cs script and replace the connection strings with your own Mobile Service URL & API Key, copied from Azure Mobile service portal.
- The BitRave demo uses Authentication with Facebook. User will need to create a Facebook app for your Mobile Service and copy & paste the App Id and App Secret into your Mobile Service's IDENTITY Facebook section. Then generate the Facebook Access Token, under Facebook's Tools-> Access Tokens.in the Facebook application page, this step could be referred from the Appendix 4. Copy the Access Token and paste into AzureUI.cs script's Access Token value. Remember to save changes.
- In Unity select the Main Camera and remove the Script in the Inspector panel.
- Reattach the AzureUI.cs script. (Drag & drop the script onto the Camera.)
- Add the demo TodoItem table (in Azure Mobile Service's get started section).
- Run in Unity Editor and connect to Mobile Service. Once logged in the user can add a TodoItem and then go to the Azure Mobile services portal and check the added item is listed in the data storage. User can query or list all items.