Openlobby: an open game server for lobby and matchmaking

E M Zamzami, J T Tarigan, I Jaya and S M Hardi
Faculty of Computer Science and Information Technology, Universitas Sumatera Utara
elvi_zamzami@usu.ac.id

Abstract. Online Multiplayer is one of the most essential feature in modern games. However, while developing a multiplayer feature can be done with a simple computer networking programming, creating a balanced multiplayer session requires more player management components such as game lobby and matchmaking system. Our objective is to develop OpenLobby, a server that available to be used by other developers to support their multiplayer application. The proposed system acts as a lobby and matchmaker where queueing players will be matched to other player according to a certain criteria defined by developer. The solution provides an application programing interface that can be used by developer to interact with the server. For testing purpose, we developed a game that uses the server as their multiplayer server.

1. Introduction
Modern video games industry is flooded with games with online multiplayer feature. The idea of multiplayer computer games has been around since 1961. The first real computer game that allows multiple player to play in the same session is Spacewar, an MIT product used to demonstrate a new PDP-1 computer. The game allows two players to shoot torpedo at one another. In the early 70s, University of Illinois' ILLIAC build a computer system called PLATO. This multi purpose computer has a server and running on a nearly a dozen computers. The system also contains two games; Empire and Airfight, which are considered as the first computer games to allow multiplayer game via computer network. One of the significant milestone in computer network games is DOOM in 1993. It features a four player cooperative mode using IPX protocol from Novell [1]. It uses peer-to-peer network architecture and can be used in via LAN or modem connection [2].

While multiplayer feature is considered capable to lengthen the lifespan of a game by allowing player to interact to other players, not all in-game interaction can enhance the game quality. Player satisfaction is highly depending on other players. Since it is impossible to fully control in-game interaction, multiplayer game usually group players according to their properties. Most games uses experience (or a value equivalent to it) to group players based on their skill. Other game requires a more complex matchmaking formula that uses additional properties such as region, latency, kill/death ratio, or even play style. To meet this need, matching players has become a complicated problem of how to match a group of available player to ensure the quality of a session. Instead of directly matching available players, most games use a lobby and a matchmaker algorithm to group optimal players. Lobby is where the players meet and wait to be matched with other player. Dhupelia et al. gives a thorough architecture of game lobby and its properties in common modern games [5].

As the player waiting in the lobby, a matchmaking system will work to find the best group of player based on some criteria. This matchmaking problem is one of the most discussed topic in game networking. There are a few researches that focus on developing an optimal matchmaking system.
Agarwal et al. developed Htrae, a latency prediction system that can be used in matchmaking application [4]. A similar system was proposed by Manweiler et al. with an application called Switchboard [7]. Similar to Htrae, this application is a matchmaking system for mobile games that heavily rely on latency between game player. Armitage et al. [6] proposed REED, a matchmaking system focused on optimizing host selection in a client-server model multiplayer game session. Another interesting research by Delalleau et al. shows how matchmaking process can be optimized to increase game quality by applying neural network to find the optimal team amongst selected player [3]. The system proposed uses a more complex data set to calculate a player's attribute such as number of matches played, kill/death ratio, firing accuracy, and headshot percentage.

However, developing an online multiplayer feature in a game is not a trivia task. It requires a specific skills and tools to develop the feature. While large game developers already have enough resources to develop this feature, (network programmer, game/database server), small game developers might not be able to have these privileges. Common game engines such as Unity and Unreal has also provide a higher level API to decrease the complexity in developing a multiplayer feature. However, this feature still requires networking programming skill and developed specifically to a certain game engine.

In this paper, we propose a solution that able to help small developers to add an online multiplayer feature in their product. We will start by discussing the architecture of the system in the next section. Section 3 will discuss the implementation of the system and the specification of hardware and software used to develop and run the system. In section 4, we will discuss the testing process which involves developing a multiplayer game application using the OpenLobby API. We conclude our work in section 5 and present the future work regarding the system.

2. Application Architecture

The overall architecture of our server is as follow:

![OpenLobby Application Architecture](image)

The server contains 3 important items; web server, database, and daemon program. The web server is responsible in receiving and processing request from client and sending the appropriate response to client. Currently, our server is able to process basic functions of a game server. The first basic function of the server is to register a new player who would like to join a multiplayer game session. When a client is requesting to join a multiplayer game, it send a request to join to the server. The server is then kept the basic information related to the request such as request arrival time and client's IP address. The second basic function is to inform the client whether the client has been joined to a multiplayer session. Our system follows a pull-notification method to perform this function; clients send a request to get the update from the server. The server is then to update the player's information related to the game such as score, win/lose, rank, etc. This information is sent after a multiplayer session has completed.
In designing the database server, we have to consider the various requirement of games that may connect to our server. Different games have different requirements and criteria to create an ideal session. Most games, if not all, rely on player's rank and experience to define an ideal set of players. Some games use additional player's properties such as region, latency, and preference. To provide the different requirements of games, we have to develop a flexible concept to store player's properties. We can not define one player's property as a single column since each game may require different properties. We came up with a simple solution to store various properties; a text based values delimited with a specific unusable character. Hence, we only need a single column to store all properties defined by the game. The figure below shows the overall architecture of our database design:

**Figure 2. OpenLobby Database Design**

3. **Implementation**
   In this section, we will go through the implementation of the architecture previously described. The web server is built using PHP. It uses HTTP protocol to communicate with the clients. In communicating with the server, application may use this 5 methods:

1. **queue**: this function is called when a user wants to be listed as queueing in a game queue. It requires player ID and game ID as its parameter. If the game ID is not registered in the system, the request will be rejected. Otherwise, the server will check the player's information based on the player ID. If the player's ID is not listed on the game database, the system will register the player ID with default experience. After both player and game are confirmed to be registered, the server will return the queue id to the player.
2. **get_status**: this function is called by user to check its queue status. The server is then return the user's status whether its waiting, found, or cancelled. When the user has been assigned to a session, the response will contain the host identification. Additionally, if a player has already queued for 5 minutes, the queue will be dropped and the player has to resend the request.
3. **get_session_info**: this function is called by user to find information regarding the session it is in. The information sent contains the host IP address, session_id.
4. **session_started**: this function is called by the host right after a session is started.
5. **update_result**: this function is called by the host after a session is over. This function will update the player's information based on the result of the session. Host can only update a session.
6. **remove_queue**: this function is called when a user wants to be removed from a queue. This action can not be performed if a player has already been matched to a session.


4. Test and Result
To test the usability of the server, we build a game that uses OpenLobby as player management in its multiplayer feature. The game developed on Android Platform and built using Java. The game is a simple math-based quiz game, Cogitare. In this game, two players will compete to answer a math question by pressing a larger number. In each session, both players will be given a limited amount of time to answer a series of questions. Each question will give an amount of point based on its difficulty. The player with more points win the session. The game also implements an adaptive difficulty method which makes the question becomes harder if the player able to answer an amount of correct answers consecutively. Harder question will grant more points. Figure 4 shows the screenshot of our game Cogitare that uses the OpenLobby to add lobby and matchmaking. To increase the quality of each session, the game also requires players with similar experience to compete in a session. The game holds two value as player's properties. The first property, experience, is linear to the amount of game played by the user. Each game completed by the user is calculated 10 experience point. The experience point is tripled if the player wins the game. The second property is correct/incorrect ratio which holds the comparison between correct and incorrect ratio of a player. However, to calculate the ratio, we need to keep two values; the amount of correct and incorrect value.

![Figure 3. Screenshot of Cogitare: Menu (left), Session Found after matchmaking process (middle), and in-game (right)](image)

The multiplayer session begins when a player sends a queue request to join a multiplayer game. The server will store the queue in the system to be processed later by the matchmaker. Every 5 second, the game checks the status of the queue; should a session has been found, it may request more information regarding its session such as players’ id, name, and properties. The server will also decide the host's of each session and send the host's IP to all players. If the player has been chosen as a host, it will wait for other player to communicate with the host. When both players are ready, the multiplayer session starts. When the session is over, the host is responsible to calculate properties changes of both players and pass the information to the server. The server stores the adjusted data to the database and the OpenLobby's protocol ended.

5. Conclusion and Future Works
We have successfully build OpenLobby, an open player management service for games regarding their platform. The server acts as a lobby and matchmaking service and provide an API for communicating with games via HTTP Request. The server also provides player's properties database to keep in track of player's progress. To test the functionality of the server, we developed a game that uses OpenLobby's service. While the system already works as intended, it does not yet applicable for commercial use due to lack of some important elements such as players and games registration process. Another essential feature that should be implemented in the near future is the security of communication process. Currently, we do not have any security feature implemented in our games. Additionally, add the social experience in the server, we would also like to implement leaderboards and friend-networking features in the next phase of the development.
References

[1] Novell 2001 Novell NetWare 6: Internet Packet Exchange www.novell.com/documentation.

[2] Armitage G, Claypool M and Branch P 2006 Networking and Online Games (England: John Wiley & Sons, Ltd.) p 12

[3] Delalleau O, Contal E, Thibodeau-Laufer E, Ferrari R C, Bengio Y and Zhang F 2012 Beyond Skill Rating: Advanced Matchmaking in Ghost Recon Online IEEE Transactions on Computational Intelligence and AI in Games 4 p 167-177

[4] Agarwal S and Lorch J R 2009 Matchmaking for Online Games and Other Latency-Sensitive P2P Systems Proceedings of SIGCOMM 2009 (Barcelona: Spain) p 315-326.

[5] Dhupelia S and Kirmse A 2004 General Lobby Design and Development Game Programming Gems 4 (Charles River Media, Inc.) p 544.

[6] Armitage G and Heyde A 2012 REED: Optimizing First Person Shooter Game Server Discovery using Network Coordinates ACM Transactions on Multimedia Computing, Communications, and Applications 8

[7] Manweiler J, Agarwal S, Zhang M, Choudhury R R, and Bahl P 2011 Switchboard: A Matchmaking System for Multiplayer Mobile Proceeding of MobiSys 2011 p 71-84

[8] Ducheneaut N and Moore R J 2004 The Social Side of Gaming: A Study of Interaction Patterns in a Massively Multiplayer Online Game Proc. of CSCW 2004 (New Orleans, LA: USA) p 360-369.

[9] Riegelsberger J, Counts S, Farnham S D and Philips B C 2007 Personality Matters: Incorporating Detailed User Attributes and Preferences into Matchmaking Process Proc. of the 40th Hawaii Int. Conf. on System Sciences (Waikoloa, Big Island, HI: USA) p 1-10.

[10] Siitonen M 2009 Exploring the Experiences Concerning Leadership Communication in Online Gaming Groups Proceedings of MindTrek 2009 p. 90-93.

[11] Fristch T, Voigt B, Schiller J 2007 The next Generation of Competitive Online Game Organization Proc. of Netgames (Melbourne: ACM).

[12] Elo A E 2008 The Rating of Chess Players, Past and Present Ishi Press.

[13] Unity Technologies 2017 Matchmakers, Unity Documentation https://docs.unity3d.com/520/Documentation/Manual/UNetMatchMaker.html, 2015, retrieved October 25, 2017.

[14] Minka T and Zaykov Y 2005 TrueSkill Ranking System Microsoft Research, https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/.

[15] Wang H, Yang H and Sun C 2015 Thinking Style and Team Competition Game Performance and Enjoyment IEEE Transactions on Computational Intelligence and AI in Games 7 p 243-254.

[16] Achterbosch L, Pierce R and Simmons G 2008 Massively Multiplayer Online Role-Playing Game: The Past, Present, and Future ACM Computers in Entertainment 5.