Massive Multiplayer Online Game Using SOA Web Services

Jashanpreet Singh
Research scholar

Chandni Dhawan
Research scholar

Abstract--In recent times Massive Multiplayer Online Game has appeared as a computer game that enables hundreds of players from all parts of the world to interact in a game world at the same time instance. Current architecture used for MMOGs based on the classic tightly coupled distributed system. While, MMOGs are getting more interactive same time number of interacting users is increasing, classic implementation architecture may raise scalability and interdependence issues. This requires a loosely coupled service oriented architecture to support evolution in MMOG application. In this paper we have proposed a service oriented architecture for massive multiplayer online game and web services. This increases the scalability and security of the system.

Index term---Massive Multiplayer Online Game (MMOG), Web services, Service Oriented Architecture (SOA)

1 Introduction

Online games give the player the ability to compete against other players over a network [1]. Massively multiplayer online game is a type of online computer game that enables hundreds or thousands of players from various parts of the world to simultaneously interact in a gaming environment they are connected to via the network. Game designers have successfully built multiplayer (MP) and massively multiplayer online games (MMOG) using different approaches. MMOGs were first introduced by various companies as Massive Multiplayer Online Role Playing Game (MMORPG). A key difference between the multi-player (MP) online game and MMOG suggested in [1] is scale and the associated infrastructure to support it. In MP games, the numbers of concurrent players are between 16 and 32. The game can be played either stand-alone or in multiplayer-network mode, and one of the players machines acts as the server. The game duration is short-lived and if the server crashes, the game is severely disrupted.

- *Jashanpreet Singh is currently pursuing masters degree program in Computer Science engineering in Lovely Professional University, India, PH-9592039300. E-mail: jashanpreet60@gmail.com*
- *Chandni Dhawan is currently pursuing masters degree program in Computer Science engineering in Lovely Professional University, India, PH-9646109932. E-mail: chand.dhawan@gmail.com*

Today MMOG's, with hundreds of thousands of players online at the same time; also span hundreds of servers. Game session must last for a long time requiring it to be run on dedicated servers equipped with a persistent database. Network bandwidth to support the game related traffic also comes with a cost.I proposed a game that allows users to interact in a network. This game use both Massive Multiplayer Online Game (MMOG) and Distributive systems for interaction between the different users which are playing game. The interaction between different users and game is with the help of web services. Because web service is the most secure medium to interact with server. Previously the interaction between client and server was done by the socket. The result is user make more than one attempt which result is misuse of the game. So we see how this problem overcomes.

2 Related Work

In this section we discuss current implementation approaches and some pros and cons of the existing MMOG architectures.

2.1 Client-Server Architecture

Some first person shooter online games like Quake and Doom typically use the client-server architecture. In this architecture a single server is responsible for handling of game states and clients. Quake II [2] like almost all virtual reality games, also follows a popular server based topology in which a single server maintains the state of the game

world. The game state is a collection of objects associated with which a small part of the game world. These parts can be computer controlled player, terrain etc. Functions determine the actions of the player and allow freedom to interact with other players. At each iteration server is responsible for function invocation and assigning players actions. Bharambe et. All in [3] also discusses to some extent the scalability analysis of Quake II. This architecture cannot be assumed to bear load of such a large number of users.

2.2 Mirrored & Scalable client-Server Architecture

This architecture is similar to client-server architecture but now each server maintains its own copy of game state. All copies of game states should be identical and there is critical requirement of synchronization between each server. Clients are now distributed amongst a number of servers which reduces the load on a single server. So this architecture presents flexibility in terms of scaling. This technique with the issue of synchronization between servers is discussed in [4]. It is difficult to maintain this synchronization technique when large numbers of players are online in MMOG and the environment has become highly dynamic. In addition since each server has a local copy of whole game state, when network becomes highly scalable, additional resources may be needed on the server for brisk processing capabilities.

2.3 Peer to Peer Games

Another area of extensible research presented some of which in [5], [6] and [7]. Game state is stored in peers and each peer is responsible for its own region. Peer to peer systems are not under the centralized control of the game server. Concept of multicasting is used where client in a periodic and distributed way send update to all other peers involved in a game session. This architecture consumes a lot of bandwidth too. It is also less reliable in terms of security as global game state is stored in local peer. So malicious peers can modify the game state and propagate to other peers. A specific middle-ware is also designed in [9] to work between application and network layer. This layer is specific to peer to peer games and very much application specific considering QoS concerns.

2.4 Distributed Deployment Techniques

Some research designs including specific middle-ware design techniques have used server clusters to improve the scaling of server oriented design. Although server control is desirable for tight administrative policies in some cases, a distributed architecture can address many challenges. As discussed in previous section, scaling is the critical issue for MMOG which can be well addressed having distributed architecture. Single point of failure risk in case of client-server architecture can also be eliminated by distributed architecture. Such design techniques are presented in [8], [9] and [10]. Main idea is to take advantage from locality of interest to distribute the game across several game servers and reduce computational constraints on each single server as well as bandwidth requirement. These designs give ability to handle a very large number of simultaneous users and provide enough computational capability to simulate the gaming algorithms. Distributed design can make use of third party server deployment also but comes with inter-node communication costs and latencies. These distributed architectures belong to tightly coupled systems making strong assumptions about the interface of interconnected components. So changes in one component's interface reflect it to the entire components interface having significant impacts on all the components.

As a result these systems are more difficult to modify and they require a retest of the entire components associated with the same interface. These systems also suffer limitations in independent and incremental development, lack of support to impedance mismatch and no dynamic real time adaptation. In each game server assigns dynamic microcells, each of which contains a very small portion of the large game state. The Microcells can be distributed between servers to balance game load efficiently.

3 Proposed Work

Game Server: It is a simple online game server that allows user to play single player or multiplayer games.

Game Client: This server deals with the information related to the player or client. Main purpose of this server is to want register user, which communicate with the Game servers.

Game Engine: Provide the gaming interface to the game client where client or user can play our game.

In the figure 3.1 a server is connected to the entire game client. Firstly we start a server. When a server is start they appear a message "Server Is Started" when a new player wants to enter in game. A request is made by the client to server. This request is transfer to the server.

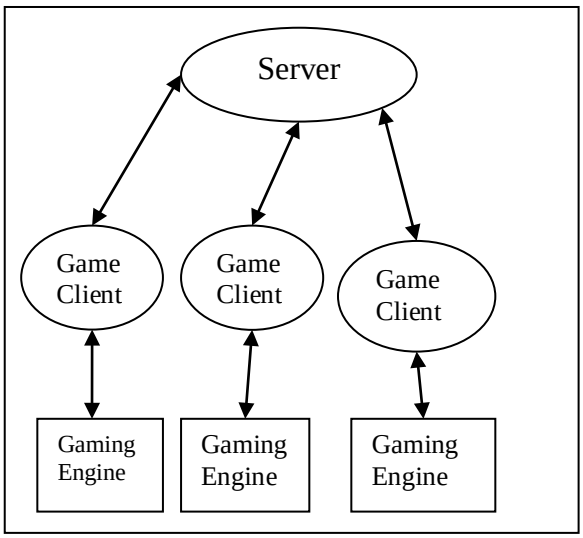

Figure 3.1

A server responds back to the client. A special web service is launch and appear a client game interface to the player

In figure 3.2 tell us about how game client works. A client will be first registered, than only he can play the game.

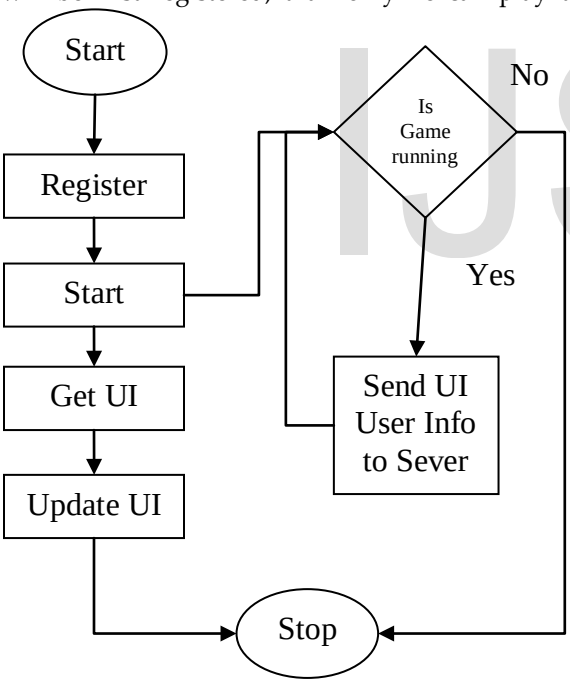

Figure 3.2

If the game is presently running then client or user information is send to the server by the user interface. This information is shared by the server to all the other users. Which are playing the game. The game is presently not running than it will stop all the services after sending all information to the server. Once this is done the user gets. If the player want to register than he play the game. Otherwise a game is not played by the player. In figure we see se a game client is connected to the server. Server provides the interface to the client. Game engine is helpful to playing the game. Now we break down large applications into smaller modules as services and unify different processes. Game server, game client and game engine are the smaller modules. The new user interface information. This information is update rapidly to the server when game is being played.

4 Results And Discussion

Sr. No.	Distributive System	SOA
Implementation	Easy	Easy
Stability	Lesser	Higher
Security	High risk	Secure
Scalability	Low	High

Table 4.1

If we talk about the implementation part, services oriented architecture (SOA) provides us to make an application and it breaks down large applications into smaller modules as services and different processes. Different groups of people both inside and outside of system can use these applications. The problem like heterogeneous system is removed by services oriented architecture implementation.

Scalability: scalability as another important issue in multiplayer online games, by building the different module like game server, game client our game engine, these also remove the scalabity. If we want change in module it will be easily changed or increase which will overcome the scalability problem.

Security: it is main issue in the massive multiplayer online game. Because we exchange the sensitive information to the users, lots of information is bye pass to the other user. So as a security purpose we use the web services.web services is the most secure medium to exchange information from one system to another.web services passes the information into xml format. This is readable by every system. So as compare to distributed system a soa provide the high security.

5 Conclusions And Further Work

MMOG architecture require much more than classic tightly coupled distributed services. In this paper we have proposed a service oriented architecture for the deployment of MMOGs. Such architecture provides loosely coupled distributed services. Our proposed architecture achieves much of its capabilities from an underline ensures service

oriented. For the scheme presented in this paper we considered web services standards in distributed system. To demonstrate the capabilities of our proposed loosely coupled service oriented architecture, we implemented a services oriented architecture web services. The ease with which a new process can integrate in the implemented prototype and level of independence available to the processes, show the usefulness of such architecture. Due to this architecture the application is more scalable, stable and secure.

In future we can provide more web services to massive multiplayer online game to enhance its functionality. Future work will concentrate on extending our architecture for MMOG.

6 ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my advisor Asst. Prof. Krishan Bansal for the continuous support of my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this paper. I could not have imagined having a better advisor and mentor for my study. Also I thank my friends in Lovely Professional University: Chandni Dhawan, Akhil Sharma, Kulwinder Singh, and Abhay Puri who has helped me in every possible way.

Last but not the least, I would like to thank my family: my parents Dr. Bhupinder Singh and Gurjeet Kaur, for giving birth to me at the first place and supporting me spiritually throughout my life.

REFERENCES

1. C. E. Sharp and M. Rowe. "Online games and e-business: Architecture for integrating business models and services into online games", IBM Systems Journal, Volume 45, Nov 1, 2006.
2. Quake II. http://www.idsoftware.com/games/quake/quake2/
3. A. Bharambe, J. Pang and S. Seshan, "Colyseus: a distributed architecture for online multiplayer games", Proceedings of the 3rd conference on Networked Systems Design & Implementation, p.12-12, May 08-10, 2006, San Jose, CA.
4. Cronin E., Kurc A. R., Filstrup B., Jamin S., "An Efficient. Synchronization Mechanism for Mirrored Game Architectures", in proceedings of ACM net games, April 2002.
5. R. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, X. Zhuang, "Donnybrook: enabling large-scale, high-speed, peer-to-peer games", SIGCOMM 2008: pp 389-400.
6. KNUTSSON, B. ET AL. "Peer-to-peer support for massively multiplayer games". In INFOCOM, July 2004.
7. H. Jin, H. Yao, X. Liao, S. Yang, W. Liu and Y. Jia, "PKTown: A Peer-to-Peer Middleware to Support Multiplayer Online Games", Multimedia and Ubiquitous Engineering, 2007. MUE '07, international Conference on, April 2007.
8. M. Assiotis and V. Tzanov, "A distributed architecture for MMORPG", Proceedings of 5th ACM SIGCOMM workshop on network and system support for games, 2006.
9. C. G. Dickey, D. Zappala, V. Lo, "Distributed architecture for Massively-Multiplayer Online Games", ACM NetGames Workshop, August, 2004.
10. A. Bharambe, J. Pang, S. Seshan, "Colyseus: a distributed architecture for online multiplayer games", Proceedings of the 3rd conference on Networked Systems Design & Implementation, 2006.