Hybrid Peer to Peer and Server Client System for Multiplayer First Person Style Games

Sergii Shepelenko
University of Tartu
sergii@ut.ee

Abstract
Nowadays Multiplayer online games become very popular, mostly because the gamers have the opportunity to interact and compete between each other. Most of MMOGs based on the Client Server architecture where players interchange periodic updates via Server. If the Server is overloaded, players start feeling the influence of network latency, which makes a game less playable. Therefore, the main idea of this article is to try to reduce the latency effects using Hybrid Peer-to-Peer architecture in Multiplayer online games.

1. INTRODUCTION

Considering Massively Multiplayer Online games (MMOs) or First Person Shooters (MMOFPSs) games we can observe, that the topology of the games in the most cases is based on Client Server architecture.
The main problem in Client Server network topology is that the performance of the game and session would be mostly depend on the server itself. When the server is overloaded, the latency is growing and the players have some problems with the game, which leads to players disconnection or even worst – losing the game items.
When the number of players is increasing, the connection to the server will become bottleneck, hence all players rely on a one single point (Server side).
Various topologies have been suggested to eliminate the single failure point, or, leastwise decrease it. In this case, Peer to Peer networks is good choice to explore. Pure Peer to Peer type of network does not have central server, and the bottleneck could be found only on the node itself, which is a part of the network [1].
The survey is carried out around Peer to Peer networks, specifically, analysis of implementation of P2P for online gaming, primarily Massively Multiplayer Online Role Playing Games (MMORPGs), based on this, trying to get a system that is catered for online First Person Shooter (FPS) games. It will be shown below, that such type of networks are not pure peer-to-peer, but such system combines Peer to Peer and Peer Supernodes that use methods to bound the extensiveness of a ”server” failure.

2. RELATED WORK

This work builds on previous research by Shawn Cassar, Prof. Matthew Montebello, Dr. Ing. Saviour Zammit from the University of Malta.

2.1 Problem Definition

When a game depend on a server it can happen that one failure is able to damage the entire gaming session, as a consequence it can cause the reduction in subscriptions. A lot of surveys [2] [3] are concentrated their interest on Peer to Peer and Peer to Peer, Client Server hybrid networks for utilizing in Massively Multiplayer Online Role Playing Games. However, it can be seen an incompatibility in surveys related to First Person Shooter games.
It is important to notice that MMORPGs are more tolerant to higher degrees of latency. It was presented by Claypool et al. [4], that playability latency in FPS in bounded to 200 ms (preferably it should not be greater than 180 ms). Also, hybrid P2P and Client Server systems for FPS games should be explored further. In online games the reaction time is crucial, thereby extreme latency is able to cause the difference between a hit or a miss. Hence suggested solution has to be effective (has low latency).

2.2 Aims And Objectives

The following aims and objectives were raised:

1. Explore feasibility and applicability of P2P in the context of limited user in FPS games in contrast to RPG games.
2. Explore applicability of P2P in games considering rapid growth of mobile devices and mobile games.
3. Have several libraries for:
    - providing network connectivity of P2P, hybrid P2P and Client-Server (library should be generic);
    - connecting over other systems that may not be maintained by having a generic library.
    - demonstrating the implementation of P2P network (with centralized node). This library should be easy integrable or it should be able to substitute already functioning network system.

2.3 Research

It is significantly important to understand latency effect, that can affect for games due to it and what type of games are more latency sensitive depending on player actions (quick or slow response). For instance, FPS games demand real time or almost real time response (if an action is slow it means that a player will be in the line of fire for longer period). Notice that majority of RPG games is less sensitive to higher levels of latency.

Analyzing an architecture of P2P, hybrid models implemented in such MMORPG systems, authors commence to search a pattern based on "area of interest". Area of interest locates around avatar (a game entity introducing the player to the game world); it is a region that determines the surroundings near player, which is visible to him. Applying "area of interest" method, one is able to commence to "partition" corresponding game world depending on avatars, instead of server structure. For instance, in "Peer@Play" [3], the system utilized is that players are bind between each other in P2P mesh. Communication among every "partition" is performed by choosing a peer (player) as a Supernode.

Second method utilizes a central server in combination with P2P topology - Peer to Peer with Central Arbiter (PP-CA) [5], players exchange updates in a P2P manner, but without carrying out verifications of consistency.

The consistency of the game is verified by a CA, that obtains all updates, but gets in touch with players in case if an inconsistency is discovered. The problem of synchronization is crucial, since not only synchronization need to ensure a consistent game state, but replication is able to assist with incident of cheating P2P networks.

Well-known Quake was considered in several survey papers for testing. The simulation for P2P system was used for 900 players. A method was suggested like the area of interest partitioning, however this method based on player interest. If we compare player interest and earlier technique it can be noticed a decrease in data that needs to be transmitted for avatars that located out of player's sight. Hence, packets can be sent rarer that leads to decreasing the overall loading on P2P mesh network.

Development of available game engines to have P2P and P2P, Client-Server hybrid functionality can have several positive signs:

- one would be already aware of the system quite well, very likely even being able to create games or have a game ready, thereby acting as a test bed;
- giving the functionality to tools that software developers already utilize well, increasing the popularity and the usage, detection and correction of bugs, improvement cycle. This can lead to crowd sourcing and open sourcing, hopefully having a more stable, more efficient, low latency, low cheats, high end peer to peer and client server hybrid library [1].

2.4 Methodology

Suggested system for a limited user, P2P and Client-Server hybrid model will need to implement a basic P2P system, which works over multicast, then have a particular function to allow the supernode to exchange information with a central server. Supernode is just a peer node that begins the session and creates session ID that the rest peers will use to contact and therefore join

the same session. When an external server is obtainable, the supernode can set a connection; the person who is using the offered library can decide which information to transmit and which not.
Ultimately, authors have introduced a network library that is open source, it is made with common exposed functions in mind, concealing the complication of P2P and Client-Server connectivity going on deep down.

2.5 Testing

In order to test the library, tests can be divided into 5 pieces:
1. Functionality test – testing P2P capabilities, after this Client-Server and overall P2P + Client-Server together. In case when test breaks down, data won't be sent, therefore foundation of this library will be worthless.
2. Latency test. Authors is trying to bring in external latency – having 180 ms latency the game should still work for at least two rival players. Applying this test authors is trying to demonstrate that library is bounded by a set Quality of Service parameters.
3. System resilience test. Shows which effect will lead to dropping Client-Server link (in the worst there should not be an interruption of running game session, but it should be limitation of adding new players).
4. Method signature count test. Method signature count compared to other game (and networks) engines. Comparison will allow to understand the similarities of method signatures and if having a system, that does P2P tasks under the hood, really makes the suggested system more difficult for developers integrate into games. At this test should be used at least 2 initialization procedures for P2P and Client-Server, send and receive methods for both networks.
5. Scalability verification and evaluation test.

2.5 Results

In this section we can observe how generic library version of ZephyrisNET can be used with a different existing engines such as: Unity3D, Source Engine, XNA, UDK and compare functions provided by every engine with ZephyrisNET's functions. Also, results show a concept of networking in every engine and how to expand existing network concept using ZephyrisNET library to peer-to-peer and Client-Server hybrid networking functionality.
Further we will consider the performance of the library (via the game) with respect to latency and number of players.

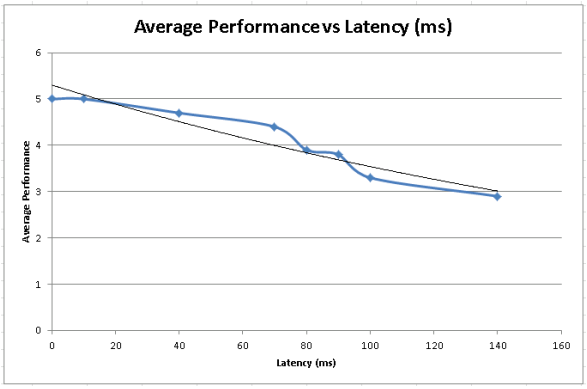

Fig. 1 Average performance vs Latency

Looking at the Figure1, we can already see a trend – latency is increasing and leads to the decrease of the library performance (and hence the game). The system performed adequately well even at the 180ms-200ms mark.

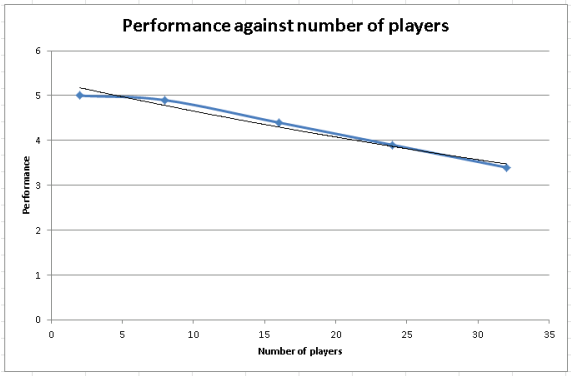

Fig. 2. Performance vs number of players

Also, performance is reduced when the number of players in a session is increased. Nevertheless, after the 24 players mark, the game was barely playable, because mobile devices seemed to have issues rendering 32 players on screen. This may suggest that during testing authors may have reached a hardware limit.

3. ACHIEVED WORK

The authors of the article, that I was using as main material in y research, said that they want to implement new library called ZephirisNET and try to connect it on existing game engines deploying the hybrid peer-to-peer network structure. Also, they put forward idea, that the server conveys the management rights to supernode. Supernode is also a player (Client) which takes this responsibility from the Server and begins to manage sessions of other players. The network topology is depicted on Figure 3.

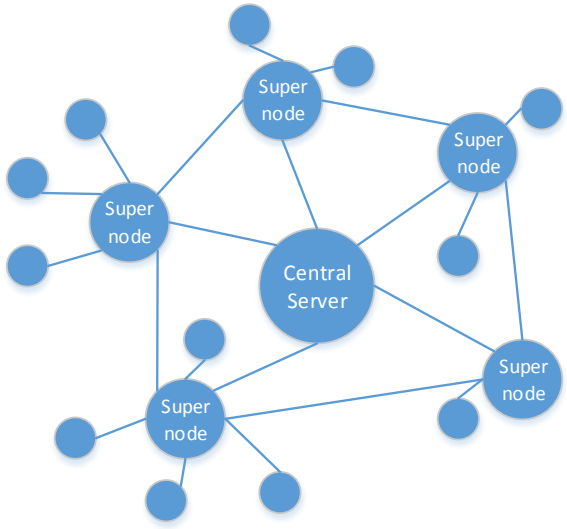

Fig.3 General Hybrid P2P topology

Despite the fact that supernode exists, it was exciting for me to figure out:

- how the virtual world divided between the supernodes and players?
- what will happen with peers that are connected to supernode in case the last leave the sessions?

In the used article, the authors say that they achieved the results, that when the supernode leaves the sessions, the players, which connect to this node, continue the game. However, they have not described the algorithm how they did it from network point of view.

Since the ZephyrisNET is an open source library, so first I started analyze their code trying to find any responses on the questions, which I mentioned before.

After having a look on the code of the library I have understood that it is more less beta version of the library and now it is no way to deploy it on exciting game engine.

ZephyrisNET library is a simple Client - Server architecture with some attempts to deploy hybrid peer-to-peer architecture.

The library firstly creates a network by calling the StrartNetwork() function and assign a supernode in this network. From the theoretical point of view, the supernode should be a node (player) which elected in the certain area of the network using predefined rules. Depending of the rules, here can be any number of supernodes. Moreover, the interactions and behaviors between supernodes should be predefined.

Considering the FSP games, the supernode should be assigned depending of "area of interest", which can be some area in virtual world. All nodes (players), that are going to start a game in this area, should be connected to this supernode. The super node is responsible for processing the game's session for these players. In ZephyrisNET the supernode is assign during the creation of the network and therefore there is no way to distinguish how this super node was elected and which rules the peer should obey to become a supernode.

ZephyrisNET library gives the opportunity to create any numbers of players randomly distributed in the map. Every player is join to the multicast group, from which it receives notifications about any changes in the game. The supernode is responsible for sending these multicast messages.

Also, during the testing of this library, all virtual players, which have been created in the network, are related to the supernode. When the supermode leave the session, the players no longer receive updates that lead to a shutdown the players from the game. Furthermore, there is no mechanism to reelect new supernode in the existing network. In this case, you have to recreate a new network and the players should reconnect to game sessions, despite the fact that the authors say otherwise.

However, there is no way to create more than one supernode in the network. Therefore, this library can considered only for small number of players, as authors said - no more than 32 players, and ZephyrisNET cannot be deploy over existing game engine, where the number of people can be achieved a thousands of players.

4. FUTURE WORK

The analyzed implementation of the ZephyrNET cannot compete with the description, which Shawn Cassar et al. have provided, therefore many questions remain open. Nevertheless, this work can be a good background for future studies. Based on this work it may be asked to next contribution:

- should be implemented independent hybrid peer-to-peer system using supermode approach.
- determine the method of supernode selection. Specify all characteristics, that peer should have to be a supernode.
- specify the interaction between supernodes.

5. CONCLUSION

Following the results obtained one can conclude the following, overall showing that the system performs as not expected, but it is still a good base for future work and expandability.
The results of the library shows that it is possible to use it with the small amount of players, however there is no explicit distribution of responsibility between server and supernode.

6. REFERENCES

1. Shawn Cassar, Prof. Matthew Montebello, Dr. Ing. Saviour Zammit : "Hybrid Peer to Peer and Server Client System for Limited Users Multiplayer First Person Style Games"
2. R. Cecin, R. Real, R. D. O. Jannone, G. Martins, and J. Luis, "FreeMMG : A Scalable and Cheat-Resistant Distribution Model for Internet Games," *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp. 83–90, 2004.
3. G. Schiele, R. Suselbeck, A. Wacker, J. Hahner, C. Becker, and T. Weis, "Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming," *Seventh IEEE International Symposium on ClusterComputing and the Grid CCGrid 07*, pp. 773–782, 2007.
4. M. Claypool and K. Claypool, "LATENCY AND PLAYER," *ACM*, vol. 49, no. 11, pp. 40–45, 2006.
5. J. D. Pellegrino, "Bandwidth requirement and state consistency in three multiplayer game architectures," *NetGames*, pp. 52–59, 2003.