Dynamic Load Balancing for Massively Multiplayer Online Games Using OPNET

Sarmad A. Abdulazeez[1], Abdennour El Rhalibi[1]
[1] Department of Computer Science, Faculty of Engineering and Technology
Liverpool John Moores University, Liverpool, UK
Sarmadalrawi17@gmail.com, A.Elrhalibi@ljmu.ac.uk

Abstract. In recent years, there has been an important growth of online gaming. Today's Massively Multiplayer Online Games (MMOGs) can contain millions of synchronous players scattered across the world and participating with each other within a single shared game. Traditional Client/Server architectures of MMOGs exhibit different problems in scalability, reliability, and latency, as well as the cost of adding new servers when demand is too high. P2P architecture provides considerable support for scalability of MMOGs. It also achieves good response times by supporting direct connections between players. In this paper, we have proposed a novel dynamic load balancing for massively multiplayer online games (MMOGs) based this hybrid Peer-to-Peer architecture. We have divided the game world space into several regions. Each region in the game world space is controlled and managed by using both a super-peer and a clone-super-peer. The region's super-peer is responsible for distributing the game update among the players inside the region, as well as managing the game communications between the players. However, the clone-super-peer is responsible for controlling the players' migration from one region to another, in addition to be the super-peer of the region when the super-peer leaves the game. We have designed and evaluated the dynamic load balancing for MMOGs based on hybrid P2P architecture. We have used OPNET Modeler 18.0 to simulate and evaluate the proposed system. Our dynamic load balancer is responsible for distributing the load among the regions in the game world space. The position of the load balancer is located between the game server and the regions. The results, following extensive experiments, show that low delay and higher traffic communication can be achieved using dynamic load balancing for MMOGs based on hybrid P2P system.

Keywords: MMOGs; Client/Server; Peer-to-Peer; OPNET Modeler;

1 Introduction

Massively Multiplayer Online Games (MMOGs) have the possibility to support hundreds of thousands of synchronised players scattered across the world and participating with each other within a single shared game. The increase in the number of players in MMOGs has led to some issues with the demands for servers which generates a significant increase in costs for the game industry and impacts on the quality

of service offered to players. With the rapidly increasing player numbers, servers still need to work efficiently under heavy load. Generally, the game server of MMOG is responsible for handling massive numbers of game players, as well as providing the indispensable consistency in order to provide the same feeling for all the game players in the same game world[1]. In real online games, when a specific game server becomes overloaded to the other games servers in the game world, it is caused by many players preferring to play in a specific region of a game world; the game server can become unsteady. This problem can be caused by an increase in network delay, traffic send and received and bandwidth.
Massively multiplayer online games (MMOGs) are readily becoming one of the most significant form of entertainment and a major mechanism of learning. Many researchers have started to use MMOGs for an educational aspect. However, there is a rarity of research on the authentic culture/cognition of MMOGs gameplay, despite its necessity for sound theory and viable design. We can used our hybrid P2P architecture in addition to the dynamic load balancing in the aspect of learning to encourage the students to understand more about the game network in an easy way using OPNET Modeler.

There are several research works to balance the load of the game server [2] [3] [4] but most of the researches in this area are based on the client-server architecture. In this paper, we have proposed and designed a new dynamic load balancing technique for MMOGs based on hybrid P2P architecture [5].

2 Research Background

2.1 Load Balancing in P2P MMOGs

Load balancing can be defined as the process of effectively distributing processing requirements of applications across a number of various servers [6] and is the main concern for all distributed systems. In other words, load balancing technique is linked to the mechanism to distribute the load of processing that occurs when the peers join and leave the system. Actually, the scalability problem in MMOGs based on Client/Server architecture is intrinsically related to load balancing [7]. However, when applying load balancing for MMOGs' infrastructure, it must be implemented in an effective way to avert essential resources over-supplying on the server side. The basic method to load balancing is to migrate nodes from heavily loaded to the other lightly loaded and then to redistribute the load across the nodes. However, this load balancing approach is far from straightforward in a peer-to-peer system. In P2P architecture, there are two main issues. Firstly, how to determine that the node is overloaded or under-loaded. Secondly, how to find an appropriate partner node where to redistribute the load [8]. In [9], Naaz et al., presented three essential parameters which generally define the strategy a particular load balancing algorithm will employ. These parameters are the maker of the load balancing decision, the information used to make the load balancing decision, and the time of the load balancing decision being made. Load balancing can be categorised into two main types: static and dynamic.

2.2 Load Balancing Techniques

Load balancing techniques are primarily used for balancing the workload in distributed systems. The load balancing can be classified into two main types: “static load balancing” and “dynamic load balancing”. These kinds of load balancing are explained in the next sections. The main objective of load balancing techniques is to improve the level of performance by redistributing the load between available server nodes. Dynamic load balancing techniques react to the current state of the system, while static load balancing techniques rely on just the average system behaviour in order to achieve the load balancing of the system, transfer decisions are separated from the actual current state of the system. This situation makes the dynamic technique more complicated than static one. However, dynamic load balancing policies have the possibility to achieved the best performance compared to static load balancing [10]. The performance of different load balancing techniques is measured by several parameters such as fault tolerant, centralised or decentralised, scalability, reliability, stability, migration, resource utilization, and responsiveness [11].

2.2.1 Static Load Balancing

Static load balancing algorithm works statically without the need of the present state of nodes. Static load balancing algorithm basically depends on the information about the average of the system work load without the need for the actual current system status. The performance of the processors is specified during the compilation time. Then according to their performance, the master processor is responsible for assigning the work load. However, the slave processors are responsible for calculating their assigned work and submitting the result to the master processor. A task is always implemented on the processor to which it is allocated, that is static load balancing algorithms are non-pre-emptive. The static load balancing algorithms work to distribute load according to a fixed group of rules, and these rules are linked to the kind of load, such as CPU power requirement and memory requirement [12]. All the prior information of the system is previously known, such as CPU power, memory availability, performance as well as data about node’s requirements for instance bandwidth. Static load balancing algorithm works with less complex situations, the reason is it does not need the information relating to the present state of the system [13]. This type of load balancing has serious disadvantages when the sudden failure happens in the system resources, tasks cannot be moved during its implementation for the load balancing. Round robin is considered one of the models of static load balancing algorithm which divides the traffic evenly between servers. However, there are several problems appearing in round robin algorithm, in order to cope with these problems, a new load balancing algorithm is proposed called Weighted Round Robin [13]. The major idea for this algorithm is that each server has been allocated a weight, therefore the servers that have the highest weight will receive more connections. The main objectives of static load balancing algorithm are to decrease the execution time of the processes and reduce the communication delay [11]. However, the main disadvantage of this technique is the algorithm does not check the load of other nodes in the network. Thus, they cannot

guarantee whether they balance up or not. This leads to decrease the performance of the system.

2.2.2 Dynamic Load Balancing

In this approach, the processes are assigned to different processors based on the new information collected [14]. Dynamic load balancing allocates processes dynamically when one of the processors becomes under loaded. Dynamic load balancing can provide a considerable improvement in performance when compared with static load balancing technique. The static load balancing algorithms are more stable compared to dynamic algorithms. When one of the processors becomes under loaded, dynamic algorithms can be allocated processes dynamically [15]. However, this approach provides an efficient load balancing but with additional cost for collecting and maintaining load information, thus it is significant to maintain these overheads within sensible limits [14]. This method is suitable for MMOGs because of the changing resources of the game world, also changes in avatar behaviour.

Dynamic load balancing algorithms work on present state of node and distribute the load between the nodes at run time. The decision of balancing is taken according to the present status of the system. Dynamic load balancing is implemented, when the load of the system and processes number is probably to change at run time. In this situation, there is a need for permanently monitoring the load of the system. This case will increase the overhead and makes the system more complicated compared to the static policies [16]. Substantially dynamic load balancer is used to track real time load distribution across the system, and all the decisions will be taken dependent on the system wide load. It also makes sure that the load is evenly distributed across all nodes. One of the key purposes of dynamic load balancing is to decrease the load produced from the game by equally controlling and real time monitoring the loads between servers (super-peers) in the game world. In other words, the load balancing is used to distribute load among a number of nodes in order to improve the benefit of the computation capability of each node, as well as to achieve optimum resource utilization which maximises throughput and minimises the task response time. Load balancing is very fundamental in MMOGs in order to improve the quality of service by dealing with continuous changes of the load over time which leads to improvement of the performance of the system. The load balancing leads to a reduction in the overall waiting time of the resources and averts too much loading on the node's resources. The main advantages of using load balancing is to increase the availability of resources, improve the performance of the system by increasing reliability, increase the throughput, maintain and increase the level of stability, optimize resource utilization and provide fault tolerance ability [10].

There are two main methods for dynamic load balancing: distributed and non-distributed [16]. In the distributed dynamic load balancing, the load balancing is done for all nodes in the system and the load balancing task is shared between them in the network system. Every node communicates with all the other nodes in the system. This leads to a high level of internal process communication. The benefit of using this method is to give a good fault tolerance for the system. When one or more nodes in the

system fail, it will not affect the whole load balancing process. This can improve the performance of the system.
However, in non-distributed dynamic load balancing, either one node or a set of nodes do the load balancing task. It can be described in two forms: centralised and semi-distributed. In centralised form, just one master node implements the load balancing in the whole system such as central node (server). The other remaining nodes communicate just with the master node. However, in semi-distributed form, all system nodes are divided into several clusters. Each cluster is working in centralised form. Election technique is used to select a central node for each cluster. The load balancing of the system is implemented by using the centralised nodes. Obviously, the dynamic load balancing is more complex compared to the static technique, but the dynamic technique has a priority rather than static technique due to the better level of performance that is provided by the dynamic load balancing. Also the interaction between nodes to achieve load balancing can be divided into two main types: cooperative interaction and non-cooperative interaction [16]. The cooperative nodes interaction are working side-by-side to achieve the load balancing. The main goal is to improve the overall level of system performance. However, in a non-cooperative interaction, each node in the system works independently to achieve its own goals. This methods is used to improve the response time of a local task

3 Literature Review

Recently, there has been a new technical challenge emerging in the gaming industry which focused on the possibility of managing the resources of game servers for massively multiplayer online games (MMOGs).
Denault et al. [18] introduced a dynamic load balancing mechanism that takes into consideration both the load related to game actions in addition to the load incurred by interest management. In this research, hybrid techniques have been used to split the main tasks the game's logic has to perform. Firstly, interest management (IM) and secondly, state update dissemination. The main concept is to partition the game world into small triangles that take into account the world geometry such as walls. A cell consists of many triangles connected with each other within the virtual world. There are two factors considered in this technique: the load associated with performing game actions and the load incurred over interest management [18]. The load balancing is achieved in two ways. The first way is called cell load model. In this model, when players and objects are equally distributed across the game world, the servers have the same cell size. However, it is uncommon that objects are equally distributed; also, most players resort to the most interesting zones in the game world. Thus, servers holding the heavily populated tiles have to do more processing for IM. The IM for each player has to be determined, and a large number of players in the cell. Therefore, the calculating load of the server is based on two components: the number of players into server's cell and the number of objects and players the server has subscribed [6]. The second way is called cell load distribution. When the cell skips the threshold value and become overloaded, it tries to move some of its tiles to a neighbouring cell. It is quite important to select the tiles to be transferred. To do this, the cell calculates a priority

value as follows; if the tiles neighbours and all members of the same cell, the priority is low such as 0. Otherwise, the priority is calculated according to the number of edge hops between this tile and the nearest tile of priority 0 [18]. However, the limitations of this technique are the authors did not mention the obvious criteria to calculate the threshold value for each player, and this research is based on the client-server system and it need more servers to cope with the increasing number of players. This causes the high cost for adding a new server to the game world.

Bezerra et al. [19] proposed a new mechanism for dividing the game world space into small regions based on kd-tree and achieves the load balancing of servers by repeatedly adjusting the split coordinates stored in its nodes. The important benefits for using kd-tree are: making this partitioning to allow a fine granularity to distribute the load and the readjustment of the regions becomes simpler. The load balancing approach is based on two main criteria: first, considering the servers as heterogeneous. It means each server may have a different quantity of resources. Second, the loads of the servers are not related to the numbers of players, but to the amount of bandwidth required to send state update messages to them. Because the number of messages sent by the players to the server will be growing linearly with the increase of player numbers, the number of state update messages sent by the server may not be good due to lack of bandwidth in the server. In this approach, using kd-tree with two dimensions, each node in the tree represents a region of the space and the node stores a split coordinate. Each node has two children and represents a subdivision of the region represented by the parent node. One of them represents the sub-region but has the parent node representing the region before the split coordinate. The region of the space is represented by the leaf node, which stores the list of avatars who are currently in that region. Ultimately, every leaf node is connected to the server of the game. When a server is overloaded, it performs the load balancing by using the kd-tree to modify the split coordinates that define its region, in addition to reduce the amount of content managed by the server. Also, each node in the tree stores two values: capacity and load of the sub-tree. The calculation of the load and capacity of a non-leaf node are equal to sum of the load of its children. However, the leaf nodes have the same values of the server connected to each one of them. The calculation of the load is dependent on the way of distributing the players among the regions. The players are deployed between the servers according to the bandwidth for each server. However, this method is not optimal when the number of messages transmitted between players is large. Using a brute-force method for calculating the loads by calculating the number of messages that should be obtained by each player by unit of time [19]. This architecture does not support the scalability, therefore this is not suitable for the MMOGs with high numbers of players.

Lu et al., [6] presented a load balancing technique based on clustered server to achieve scalability. Allowing servers to transfer player actions to each other, however the responsibility for processing players' actions remains with the server to which they are initially earmarked. The system consists of two main levels: the application level and the database level. Firstly, the player connects to the server cluster through a load balancer, and he/she is then linked to a particular server in the application level. Application level is dedicated to meet the runtime requirements of game play. Through the database level, an application server can have access to the virtual world constructs and players' statistics via the load balancer that exists between application level and the database level [6]. But, this method is ineffective because the load balancer has not

enough resources to cope with the increase of player numbers. Also, the cost of providing a large number of servers is very high as well as the cost of maintenance.
Usually zone size in MMOGs is fixed, but there are some techniques that cope with dynamic region size using for instance Voronoi diagrams. Most of these techniques do not have load-handling mechanisms. The problem is the difficulty to predict player density or deploy the players before the start of a game. Thus, the server cannot share its resources with surrounding region servers. This approach, presented a load balancing mechanism where region shapes are not predefined. Consider a system with a number of masters/servers with a large single region in the game world, the players can join and leave the game over time and at first only one server is involved. In this method [20], the game is divided dynamically depending on players' logical position and interaction pattern. Consequently, the zone shapes are not uniform. The bisection algorithm is used to divide a region into two sub-regions of roughly equal load while attempting to reduce the communication cost, such as the number of links crossing logical boundaries. Depending on the bisection methods, at the first time, the region is divided into one dimension to produce two sub-regions. Other partitions are made repeatedly in the new sub-regions if necessary [20]. However, this method is not efficient when the number of players is too large because we need extra servers to manage the load, then the cost of provide servers is high; furthermore, the maintenance

4 Design Dynamic Load Balancing Technique for MMOGs

Dynamic load balancing techniques rely on recent system load information in addition assigning the job for each server node at run time. The load balancing decisions are dependent on the current system state. Thus allowing the workload to dynamically shift from an overloaded node to the light-load node in order to increase the response time of server nodes. The main advantage of using the dynamic load balancing is the capability of responding to difference in the system. We have designed a new dynamic load balancer for MMOGs based on hybrid P2P system. Our load balancing system measured the load of each super-peer in the game world space in order to find out the availability for each region. The measurement of the load is based on the node resources such as power of the CPU, memory available, bandwidth, disk space as well as stability and reactivity. When the first player joins the game for the first time, it must be registered in the game server. The server will assign this player as a super-peer for the region. After that, each player wanting to join the game must be registered in the game server. The game server sends the ID of the player to the dynamic load balancer to find the suitable region to connect to it. The load balancer is responsible for sending a message to the super-peer of the region to inform him that a new player will join the region. When the player joins the second time or subsequent times, it should join the game through the game server in order to find the appropriate region to allocate to it. However, in the case of a player leaving the game, it should inform the super-peer or clone-super-peer of the region about that. The super-peer is responsible for informing both the dynamic load balancer and the game server. The dynamic load balancer updates the load of each region when the new player joins the game or when the player leaves the game. When the super-peer of the region leaves the game, it must inform

both the game server and the clone-super-peer of the region about that. The clone-server will become the super-peer of the region and assign a new clone-super-peer for that region. All the information will be updated in the game server and the dynamic load balancer. If all the regions in the game world space are overloaded, the load balancer will inform the game server to create a new region in order to handle the work load of the players. While, if there are regions that do not contain any players, the dynamic load balancer will work with the server to destroy the region and allocate the super-peer and clone-super-peer of that region to another convenient region in the game world space. The dynamic load balancing algorithm is explained in the next section.

4.1 Dynamic Load Balancing Algorithm

Each player must be registered in the game server at the beginning of the game. The first player will be assigned as the super-peer of the first region in the game world space. However, the second player will be allocated as the clone-super-peer of that region. All the information of the super-peer and clone-super-peer will be sent to the load balancer in order to inform it. After that, each player will be assigned as a normal player in the region by sending all the players from the game server to the super-peer through the load balancer. When the first region is full, the load balancer is responsible for informing the game server about that in order to create a new region and assign both super-peer and clone super-peer for that region. If the normal player wants to leave the game, he/she will inform the super-peer of the region about that leaving. The super-peer turn to inform both load balancer and the game server to update the load of this region. However, if the player decides to migrate to another region, he/she will inform the clone-super-peer and send the region ID. The clone-super-peer will check the availability and the possibility of that region to join the player through the load balancer. If the new region is available and possible to accept the player, the load balancer is responsible for allocating the player to that region and update the load information for the two regions. All the updated information is sent to the game server and the super-peers of the regions concerned. As for the leaving or migration of the super-peer of the region, he/she will send a notification the game server, load balancer and clone-super-peer of the region. The clone-super-peer will be assigned as the super-peer of this region and he/she has the possibility to allocate a new clone-super-peer for the region. All the updated information is sent to the game server and the load balancer. However, if the clone-super-peer leaves or migrates the game, he/she will inform the super-peer of the region to assign new clone-super-peer of the region. All the updated information is sent to the load balancer and the game server.

4.2 Simulation of Dynamic Load Balancing

In order to design the simulation for dynamic load balancing for MMOGs, we have fail all the nodes of players before the simulation. It will help us to start the game from the base. We have used the deterministic technique for generating failures and recoveries for each node (peer) in the game world space. All the nodes will start to recover according to the specific time that will be given to each node in the game world space.

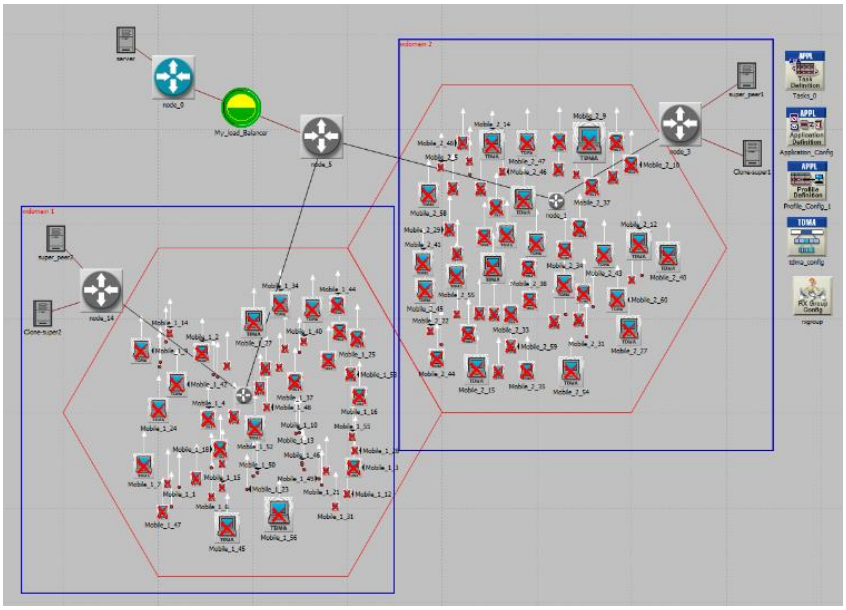

Fig.1. Dynamic Load Balancing for MMOGs based on Hybrid P2P System with 125 peers

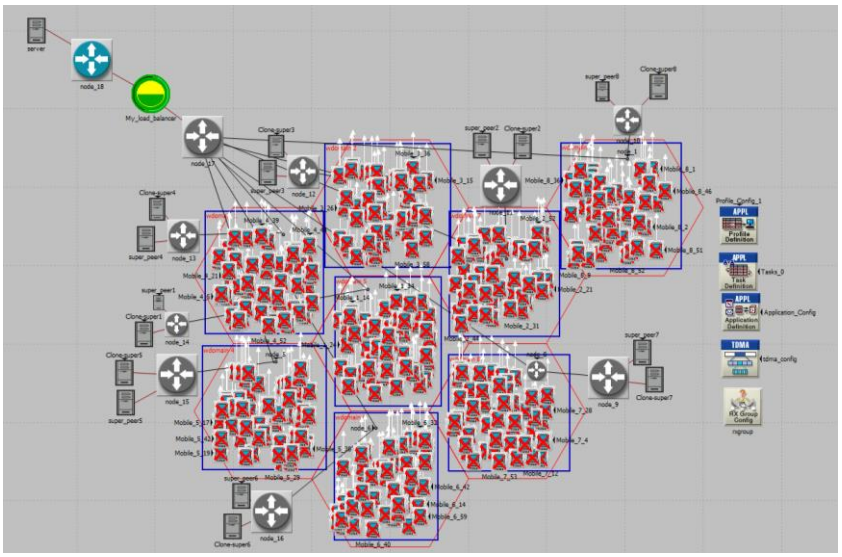

Fig.2. Dynamic Load Balancing for MMOGs based on Hybrid P2P System with 500 peers

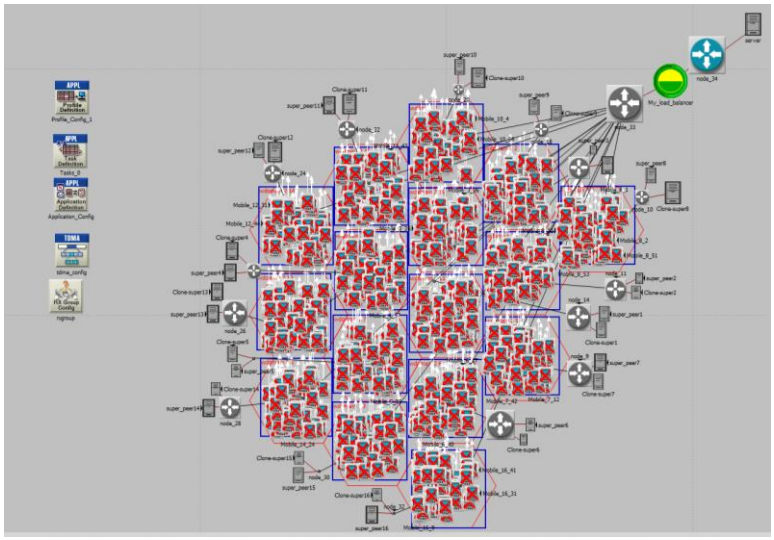

Fig.3. Dynamic Load Balancing for MMOGs based on Hybrid P2P System with 1000 peers

After the player's node recovers, it will start the game by registration in the game server. The game server is responsible for creating a profile for each player and sends this information to the load balancer in order to enable the load balancer to distribute the players among the regions inside the game world space. The load balancer is responsible for distributing the nodes (peers) among the regions in the game world space. Figures 1, 2, and 3 show the simulation design of dynamic load balancing for MMOGs based on hybrid P2P system with 125, 500, 1,000 peers respectively.

4.3 Results of Dynamic Load Balancing

In this section, we have analysed the network performance under simulation scenarios from the view of two primary parameters.

4.3.1 Overall End-to-End Delay

Figures 4, 5, and 6 illustrate the overall end-to-end delay results for the dynamic load balancing for MMOGs based on hybrid P2P architecture and compare the results with static load balancing for MMOGs based on hybrid P2P architecture. The figure below illustrates a considerable variation of delay when using static load balancing for MMOGs hybrid P2P system with 125, 500, and 1,000 players compared to the static load balancing for MMOGs based on hybrid P2P architecture.

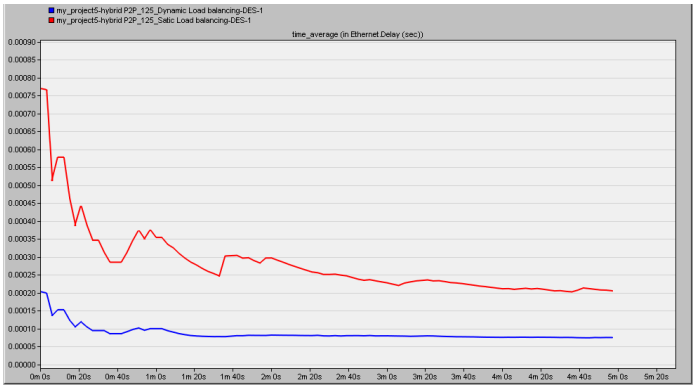

Fig.4. Overall Delay for Dynamic Load Balancing for MMOGs with 125 Peers

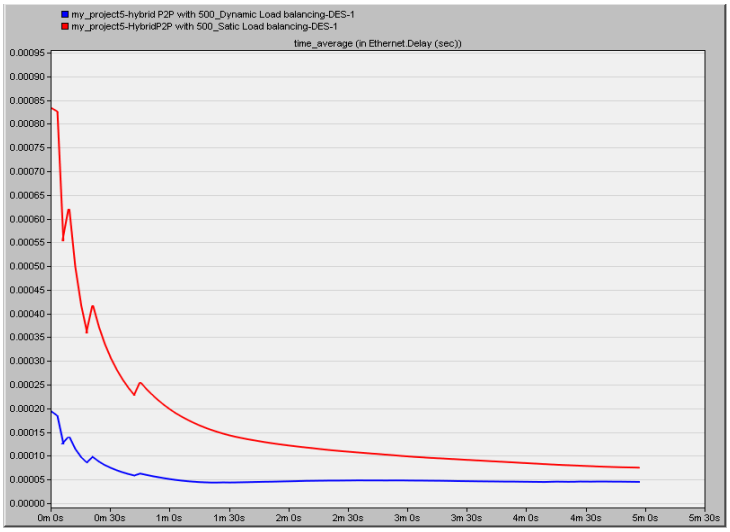

Fig.5. Overall Delay for Dynamic Load Balancing for MMOGs with 500 Peers

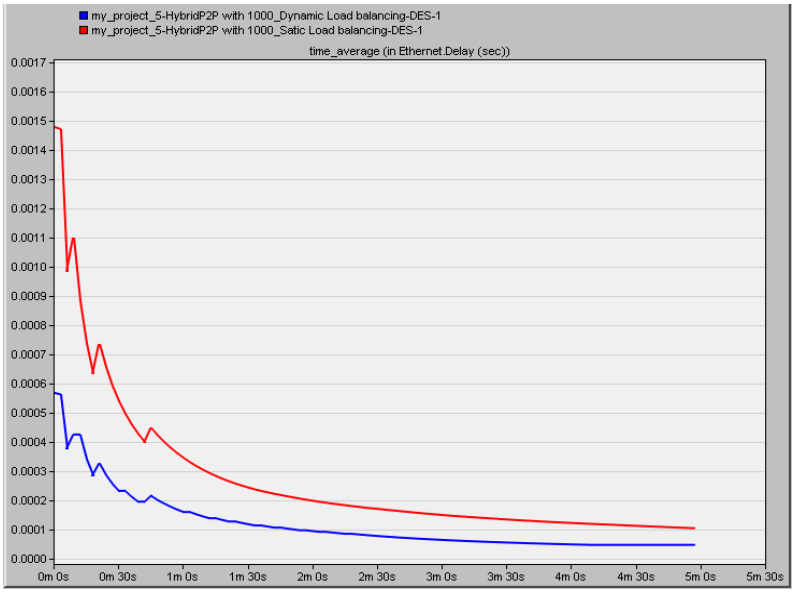

Fig.6. Overall Delay for Dynamic Load Balancing for MMOGs with 1,000 Peers

4.3.2 Traffic Received

Figures 7, 8, and 9 illustrate the traffic received results for the dynamic load balancing for MMOGs based on hybrid P2P architecture and compare the results with static load balancing for MMOGs based on hybrid P2P architecture. The figure below shows a big variation of traffic received when using dynamic load balancing for MMOGs based on hybrid P2P system compared to the static load balancing for MMOGs based on client-server architecture.

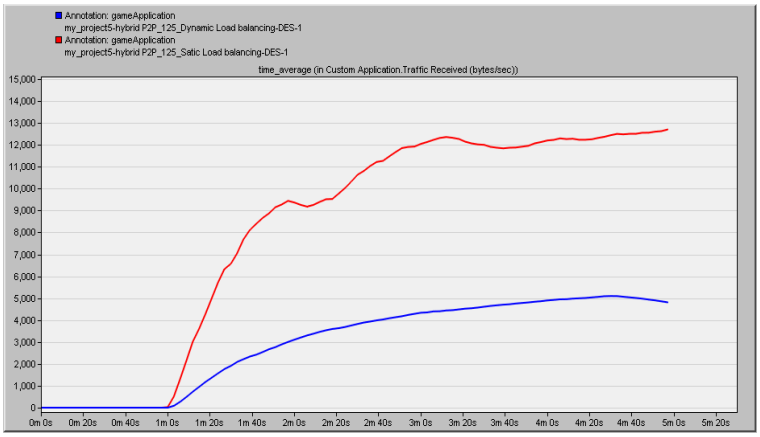

Fig.7. Traffic Received for Dynamic Load Balancing for MMOGs with125 Peers

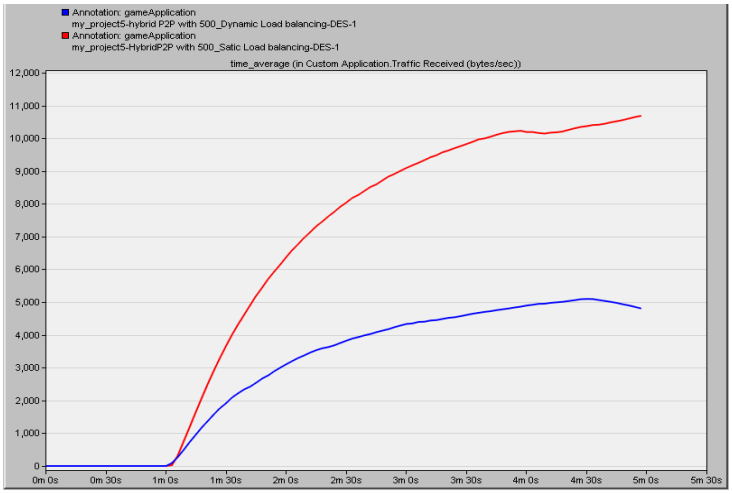

Fig.8. Traffic Received for Dynamic Load Balancing for MMOGs with 500 Peers

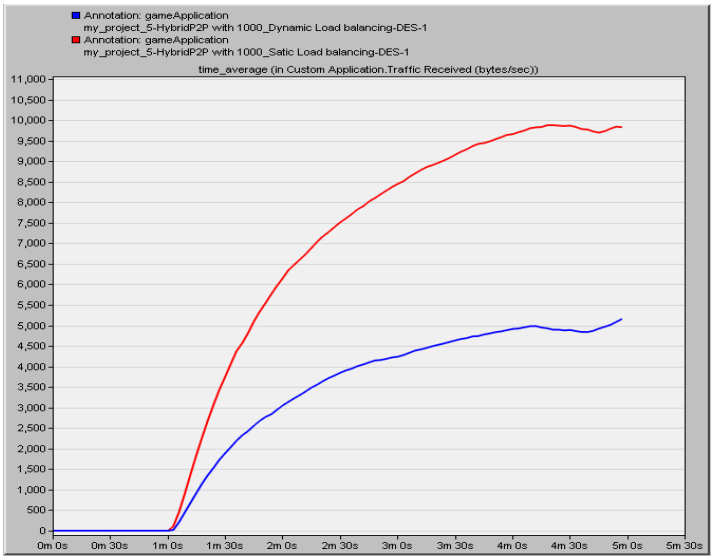

Fig.9.Traffic Received for Dynamic Load Balancing for MMOGs with 1,000 Peers

5 Discussion

This section discusses the different results of the dynamic load balancing evaluation. Starting with the limitations that influence the results of the evaluation which are illustrated in the previous sections. Followed by the evaluation findings of the dynamic load balancing simulation.

5.1 Results Discussion

In general, our dynamic load balancer for MMOGs based on hybrid P2P architecture aims to reduce the costs of a traditional game server infrastructure. It provides a good level of scalability, a mechanism to deploy the players among the regions in the game world space, AoIM technique to reduce the number of receiver groups, and control of the load of each region inside the game world space compared to the load balancer for MMOGs based on client-server architecture. Also the dynamic load balancing provides an efficient mechanism to manage and control the migration of players from one region to another. The results illustrate that the use of dynamic load balancing for MMOGs based on hybrid P2P provides an easy way to manage the load of the regions and provides low delay and traffic received when compared to the load balancing for MMOGs based on client-server. Our dynamic load balancing for MMOGs has the flexibility to apply in the real life system because it has the suitable mechanism to control and manage peers joining, migrating, and leaving the game. Also, it has a robust mechanism to manage the AoIM for each player in the game world.

6 Conclusion

In this paper, we have introduced the subjects and issues related to our research and explained the main contributions for dynamic load balancing. We have also introduced the main types of load balancing, as well as presented the design and simulation of dynamic load balancing for MMOGs based on both hybrid P2P and client-server system. We use OPNET Modeler 18.0 to model and simulate the new load balancing system. We have used OPNET simulation to enable the networks construction, study of communication infrastructure, design of individual devices, and simulation of protocols and applications. The results illustrate that the dynamic load balancing for MMOGs hybrid Peer-to-Peer system produces low delay and low traffic received in the network topology when compared with static load balancing for MMOGs based on client-server system.

References

1. J. C. S. Lui, M. F. Chan, and S. Member, “An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems,” vol. 13, no. 3, pp. 193–211, 2002.

2. Y. Deng and R. W. H. Lau, “Dynamic Load Balancing in Distributed Virtual Environments using Heat Diffusion,” vol. 0, no. 0, pp. 1–20, 2013.
3. H. Jiang, A. Iyengar, E. Nahum, W. Segmuller, A. Tantawi, and C. P. Wright, “Design , Implementation , and Performance of A Load Balancer for SIP Server Clusters.”
4. S. Zhou, “A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games.”
5. S. Abdulazeez, A. El Rhalibi, M. Merabti, and D. Al Jumeily, “Survey of Solutions for Peer-to-Peer MMOGs”, 2015 IEEE International Conference on Computing, Networking and Communications (IEEE ICNC), California, USA.
6. F. Lu, S. Parkin, and G. Morgan, “Load balancing for massively multiplayer online games,” Proc. 5th ACM SIGCOMM Work. Netw. Syst. Support games - NetGames ’06, p. 1, 2006.
7. R. Chertov and S. Fahmy, “Optimistic load balancing in a distributed virtual environment,” Proc. 2006 Int. Work. Netw. Oper. Syst. Support Digit. audio video - NOSSDAV ’06, p. 1, 2006.
8. C. Link, Q. H. Vu, C. Ooi, M. Rinard, and K. Tan, “Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems,” 2014.
9. R. Naaz, Sameena; Alam, Afshar; Biswas, “Load Balancing Algorithms for Peer to Peer and Client Server Distributed Environments,” vol. 47, no. 8, pp. 17–21, 2012.
10. C. Science and M. Studies, “A comparative study of static and dynamic Load Balancing Algorithms,” pp. 386–392, 2014.
11. D. Wadhwa and N. Kumar, “Performance Analysis of Load Balancing Algorithms in,” vol. 4, no. 1, pp. 59–66, 2014.
12. S. Rajani and N. Garg, “A Clustered Approach for Load Balancing in Distributed Systems,” vol. 2, no. 1, pp. 1–6, 2015.
13. N. K. Mishra, “Load Balancing Techniques : Need , Objectives and Major Challenges in Cloud Computing- A Systematic Review,” vol. 131, no. 18, pp. 11–19, 2015.
14. S. Malik, “Dynamic Load Balancing in a Network of Workstations,” 95.515F Res. Rep.
15. S. Sharma, S. Singh, and M. Sharma, “Performance Analysis of Load Balancing Algorithms,” World Acad. Sci. Eng. Technol., vol. 38, pp. 269–272, 2008.
16. A. M. Alakeel, “A Guide to Dynamic Load Balancing in Distributed Computer Systems,” vol. 10, no. 6, pp. 153–160, 2010.
17. P. B. Soundarabai, S. R. A, R. K. Sahai, J. Thriveni, K. R. Venugopal, and L. M. Patnaik, “Comparative Study on Load Balancing Techniques in Distributed Systems,” vol. 6, no. 1, pp. 53–60, 2012.
18. A. Denault, C. Canas, J. Kienzle, and B. Kemme, “Triangle-based obstacle-aware load balancing for massively multiplayer games,” 2011 10th Annu. Work. Netw. Syst. Support Games, pp. 1–6, Oct. 2011.
19. C. E. B. Bezerra, J. L. D. Comba, and C. F. R. Geyer, “A Fine Granularity Load Balancing Technique for MMOG Servers Using a KD-Tree to Partition the Space,” 2009 VIII Brazilian Symp. Games Digit. Entertain., pp. 17–26, 2009.
20. D. T. Ahmed and S. Shirmohammadi, “Uniform and Non-Uniform Zoning for Load Balancing in Virtual Environments,” 2010.
21. “OPNET .” [Online]. Available: http://www.riverbed.com/products/performance-management-control/opnet.html. [Accessed: 21-January-2018].