國立交通大學

網路工程工程研究所

碩士論文

適用於多人線上遊戲動態負載管理之

混合同儕式雲端架構

An Efficient Hybrid P2P MMOG Cloud Architecture for

Dynamic Load Management

研 究 生：王金鴻

指導教授：王國禎　博士

中 華 民 國 一百 年 六 月

適用於多人線上遊戲動態負載管理之混合同儕式雲端架構

An Efficient Hybrid P2P MMOG Cloud Architecture for Dynamic Load Management

研 究 生：王金鴻　　Student：Ginhung Wang

指導教授：王國禎　　Advisor：Kuochen Wang

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institutes of Network Engineering

Department of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in Computer Sciencmae

June 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

適用於多人線上遊戲動態負載管理之混合同儕式雲端架構

學生：王金鴻　　指導教授：王國禎 博士

國立交通大學 網路工程研究所

摘 要

近年來由於大型多人線上遊戲(MMOGs)的蓬勃發展，引起了學界和業界的高度興趣。多人線上遊戲需要高度的彈性來面對環境的變化，因此我們利用雲端的特性來與MMOG做結合。在雲端遊戲環境中，我們利用虛擬機器來取代傳統實體遊戲伺服器。由於現今MMOGs主要系統架構為多伺服器架構，虛擬遊戲世界會被切割成數個遊戲區域，每個遊戲區域由一個或多個實體遊戲伺服器負責和客戶端玩家進行遊戲資訊的傳輸及執行。然而，在多伺服器架構下，會因為大多數客戶端玩家感興趣的地圖區域相同，而造成遊戲伺服器負載不平衡。在本論文中，我們在MMOGs中使用同儕式雲端計算架構，使MMOGs在雲端計算的環境下獲得較彈性的資源利用。同儕式雲端計算是一個結合高計算能力、高擴充性、高可信賴，以及分享伺服器資源及資料的新概念。本論文提出一個混合同儕式雲端架構，此一架構可以更適用於大型多人線上遊戲，它改進了現行多人線上遊戲多伺服器架構的缺陷。除了提出一個適用於多人線上遊戲之混合同儕式雲端架構外，我們也針對

每一個遊戲伺服器提出多門檻負載管理機制和遊戲伺服器之間的負載管理機制。本篇論文的架構與多伺服器架構相比較，在小於300毫秒的回應時間內，我們的架構比多伺服器架構可多服務10.31%的玩家量。此外，在中高負載時，我們的架構比多伺服器架構少27.9%的錯過期限比率。

關鍵詞：雲端計算、負載管理、同儕式多人線上遊戲、資源配置。

An Efficient Hybrid P2P MMOG Cloud Architecture for Dynamic Load Management

Student: Ginhung Wang **Advisor: Dr. Kuochen Wang**

Department of Computer Science
National Chiao Tung University

Abstract

In recent years, massively multiplayer online games (MMOGs) become more and more popular. Many researchers, both in academia and industry, are very interested in MMOGs. MMOG environments require a high degree of flexibility to respond to environmental changes, including load change. We combine cloud computing with MMOGs to increase flexibility of resource allocation. In an MMOG cloud environment, we use virtual machines (VMs), instead of traditional physical game servers. A game world is divided into several game regions. Each game region is serviced by at least one VM. However, in the multi-server architecture, loads of regional servers may be unbalanced because there may be some regions that attract more players. Peer-to-peer (P2P) cloud computing is a new approach that combines high computation power, scalability, reliability and efficient information sharing of servers. This paper proposes a hybrid P2P cloud architecture for MMOGs which includes two-level load management, multi-threshold load management for each game server and load management among game servers. It is suitable for players to interact with P2P cloud servers and it avoids bottlenecks of the current multi-server MMOG architecture. Simulation results show that the proposed architecture can support 10.31% more players under no deadline (300 ms) miss compared to the multi-server architecture. The proposed hybrid P2P cloud

architecture can reduce the average response time by 20.6% compared to the multi-server architecture under medium to high load through flexible allocation of resources (virtual machines). The proposed architecture also has 27.9% smaller deadline miss ratio than the multi-server architecture under medium to high load.

Keywords: cloud computing, load management, massively multiplayer online game, peer-to-peer, resource allocation.

Acknowledgements

Many people have helped me with this thesis. I am in debt of gratitude to my thesis advisor, Dr. Kuochen Wang, for his intensive advice and guidance. I would also like to express my appreciation for all the classmates in the *Mobile Computing and Broadband Networking Laboratory* for their invaluable assistance and inspirations. The support by the National Science Council under Grant NSC99-2221-E-009-081-MY3 is also gratefully acknowledged. Finally, I thank my father, my mother and my friends for their endless love and support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation for a new MMOG architecture

MMOGs have emerged in the past decade as a new type of large-scale distributed applications. An MMOG is a real-time virtual world with many players across the real world. Traditionally, MMOGs operate as multi-server architecture which is composed of many game servers. Game servers simulate a virtual game world. They receive and process commands from clients (players), and inter-operate with a billing and accounting system [3]. To support ever more and more players, MMOG environments require a new architecture with a high degree of flexibility to respond to environmental changes, including load change. Therefore, in this paper, we propose a hybrid P2P cloud architecture for MMOGs to provide flexible resource usages to deal with load change.

1.2 Load management

The game providers need to install and operate a large infrastructure, with hundreds to thousands of computers for each game [27]. For example, the operating infrastructure of an MMOG, World of Warcraft, has over 10,000 computers [5]. The traditional solutions to decrease server loads are deploying more servers to the heavy-loaded areas [27]. This solution is simple but has limitation of scalability. The other solution is using the multi-server architecture, where game servers can be added to heavy-loaded areas on demand. In this paper, we propose a hybrid P2P cloud architecture for MMOGs which includes two-level load management: multi-threshold load management for each game server and load management among game servers to resolve current architecture drawbacks.

1.3 Thesis organization

The rest of this paper is organized as follows. In Chapter 2, we introduce an existing load management procedure. In Chapter 3, we review existing MMOG architectures. We present the proposed hybrid P2P MMOG architecture and a two-level load management scheme in detail in Chapter 4. In Chapter 5, we discuss simulation results and compare the proposed architecture with the multi-server architecture in terms of response time and deadline miss ratio. In Chapter 6, we give concluding remarks and outline future work.

Chapter 2

Background

2.1 Load distribution techniques

Today's MMOG's used three main techniques, *zoning*, *replication*, and *instancing*, to serve hundreds to thousands of player, as shown in Figure 1. Through zoning, the virtual world was divided into smaller blocks which are handled by different servers. Replication takes place when the load of a zone is heavy because a huge number of players center on the zone. The players still can see each other. Finally, instancing is a simplification of replication, which distributes the session load by starting multiple parallel instances of the highly populated zones. The instances are completely independent of each other, meaning that two players from different instances will not see each other, even if they are located at nearby coordinates [15].

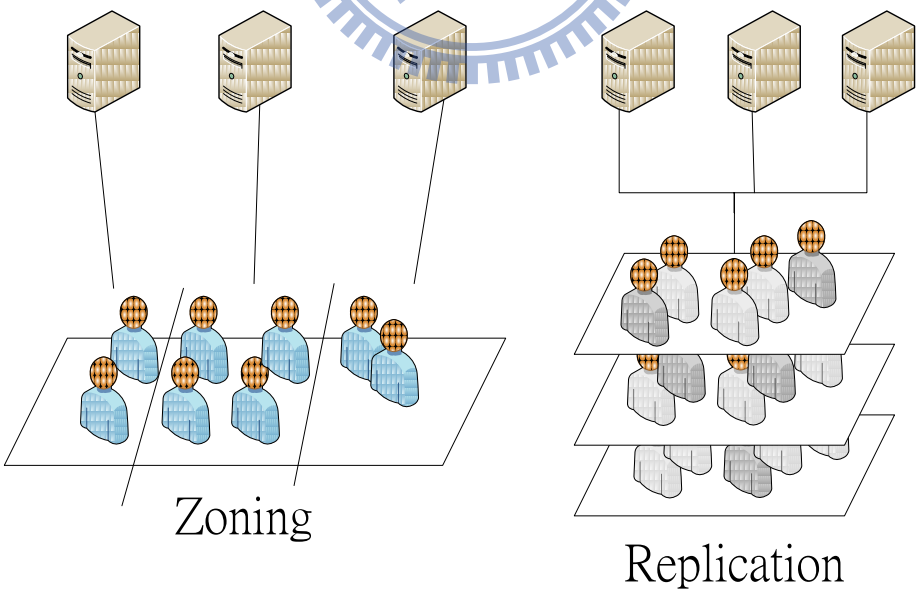

Figure 1: Load distribution techniques [27].

2.2 Load management in MMOGs

In [29], the authors proposed a new predictive resource provisioning method based on a stack of five services depicted in Figure 2. Firstly, a monitoring service collects online metrics related to the performance of an MMOG session which could be of two kinds: (1) game session-related, such as the number of entities and their positions in the game world, and (2) resource-related, such as the CPU, memory, storage, and network load. Secondly, a load prediction service is used to anticipating the future game world entity distribution from historical traces. A capacity planning service includes generic analytical models to map an entity distribution into a resource's load, such as processor, memory, storage or network load. In this service, it focuses on foreseeing potential hotspots in the game servers which make the game environment fragmented and unplayable. A resource allocation service is provisioning of a right amount of resources required for a proper execution that guarantees a good experience to all players. Finally, a load balancing service is balancing the active servers' load.

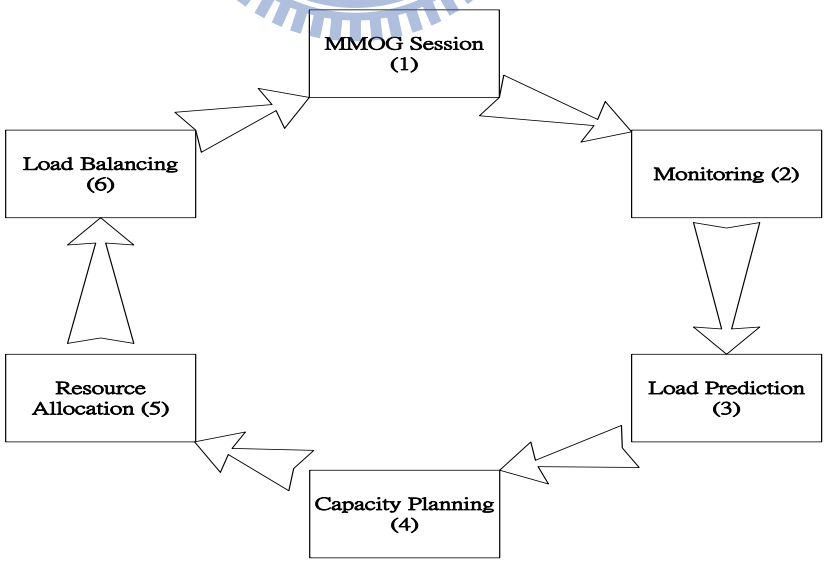

Figure 2: Load management cycle [29].

Chapter 3

Related Work

3.1 Client-server MMOG architecture

Client-server MMOG architecture has players (clients) that send requests to a single server. It is simple and easy to implement, but it is less scalable. As shown in Figure 3, commands from players must go through a single server, and the single server handles commands and returns state updates to players. The single server must be in the presence of insufficient bandwidth and copious waiting time for players. To resolve the problems, there are several approaches [28] to distribute traffic among multiple servers, such as a mirrored server [29], generic proxy [30], or booster box [31].

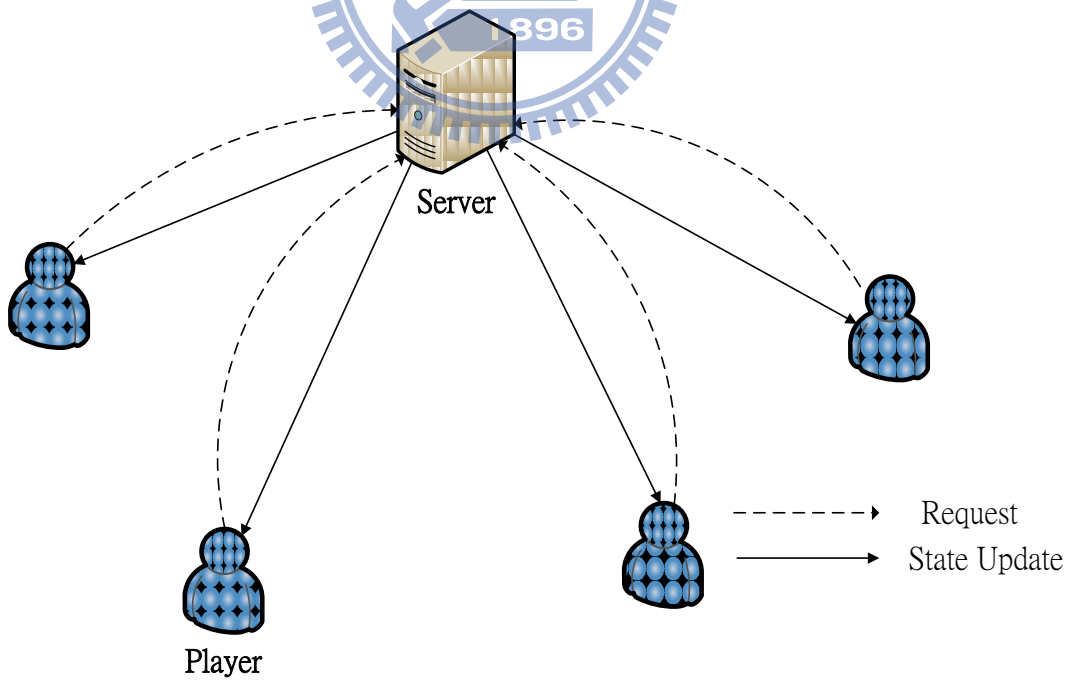

Figure 3: Client-server MMOG architecture [34].

3.2 Multi-server MMOG architecture

Multi-server MMOG architecture [19] has various kinds of servers which provide different functionalities as shown in Figure 4. The various servers are described as follows [19]:

- Master Server - This server handles access of players and communicates with operated servers. It assists players logging in a zone server and transfers data of players to the corresponding zone server from the character server.
- Character Server - This is mainly stored the data of players. It assists players in using the same characters on any zone servers.
- World Daemon - This server transfers players from a zone server to another zone server that players expected. It also monitors each load of zone servers.
- Zone Cluster Server – It is composed of zone servers and mainly serves players. A zone server is assigned to serve a specific zone.
- Client - A client moves between zone servers. A client sends a request to an interested zone server.

In this MMOG multi-server architecture, the players in the same zone connect to the specific zone server. Some zone servers will suffer heavy load if many players move into the same zone. Hence, load balancing is needed to prevent the downgrade of service. In [29], it creates a management server that monitors zone servers and decides a load balancing strategy. In this case, the management server would become the bottleneck of the system [20].

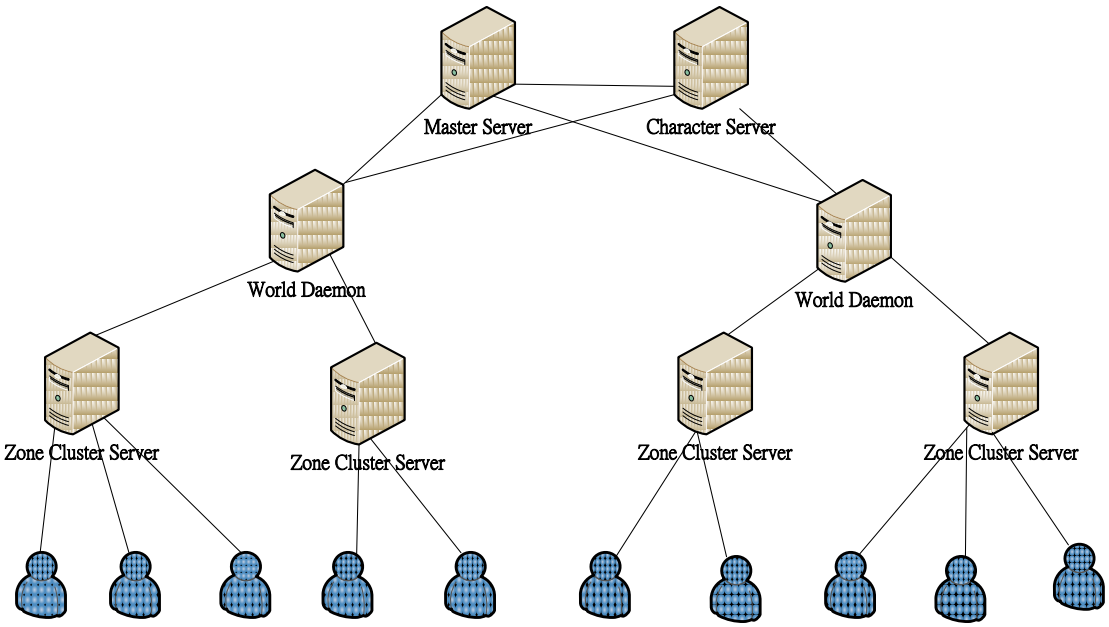

Figure 4: Torque MMOG multi-server architecture [19].

3.3 P2P MMOG architecture

In P2P MMOGs, they advance to multi-server architecture by lowering the cost of centralized infrastructures and by distributing the processing load [20]. But the security is their drawback. They have to request clients to share loading and clients could get some data of other clients. Because of this reason, the P2P MMOG architecture usually runs on lower security games.

As shown in Figure 5 [20], the server serves players and assigns some normal nodes to become service nodes. A normal node is a player. When the server loading is more than a threshold, the server will transform some normal nodes to service other players. This process could cut down server loading but the security emerges more problems.

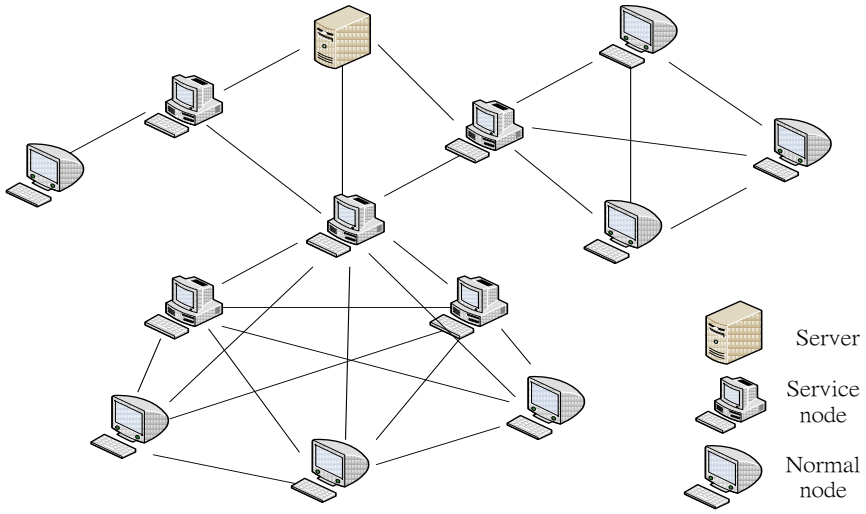

Figure 5: P2P MMOG architecture [20].

3.4 P2P cloud computing architecture

As shown in Figure 6 [26], the architecture includes three basic roles: User, Central Peer and Side Peer. The Central Peer and Side Peer form two P2P networks, called central P2P network and side P2P network, respectively. The central P2P network mainly maintains metadata of dynamic mapreduce and backups DHT P2P storage cloud. The side P2P network is mainly used to provide storage and computing resources.

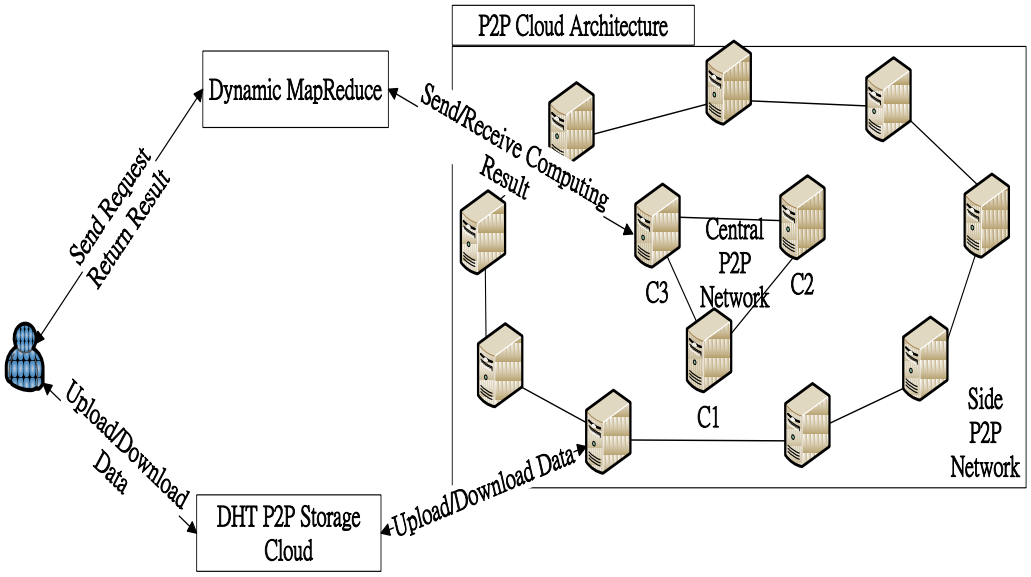

Figure 6: Cloud computing architecture based on P2P [26].

Chapter 4

Proposed Hybrid P2P MMOG Cloud Architecture

In this chapter, we propose a hybrid P2P cloud architecture for MMOGs which includes two-level load management: (1) multi-threshold load management for each game server and (2) load management among game servers. The components of the proposed architecture will be described in section 4.1. Game servers in the game server cloud share information by P2P manner. Section 4.2 introduces a game server cloud. In order to support this architecture, we propose a multi-threshold load management procedure in section 4.3.

4.1 Proposed cloud architecture

MMOG environments require a high degree of flexibility to respond to environmental changes, including load change. Thus, we combine cloud computing with MMOGs to increase flexibility of resource allocation. Using virtualization technical on cloud could help us get more security and scalability. The proposed cloud architecture includes four basic components: *game server cloud*, *players*, *character database* and *game regions data storage*, as shown in Figure 7. We introduce each component's characteristics as follows:

- *Game servers cloud* – It consists a number of game servers and a game server has a number of virtual machines (VMs) to serve players. A game server receives requests from players, calculates new game states in regions, and responses to players. A game server is also in charge of creating accounts for players when they login for the first time.
- *Players* – They move between game regions of a game world. The load of a game region increases when players move into the game region.

- *Character database* – It stores the data of players, such as equipment, rank, site, etc. It also backups players' data. The backup method is based on backup method of hadoop.
- *Game regions data storage* – It maintains game states in each region. It also backups game states and object states in each game region.

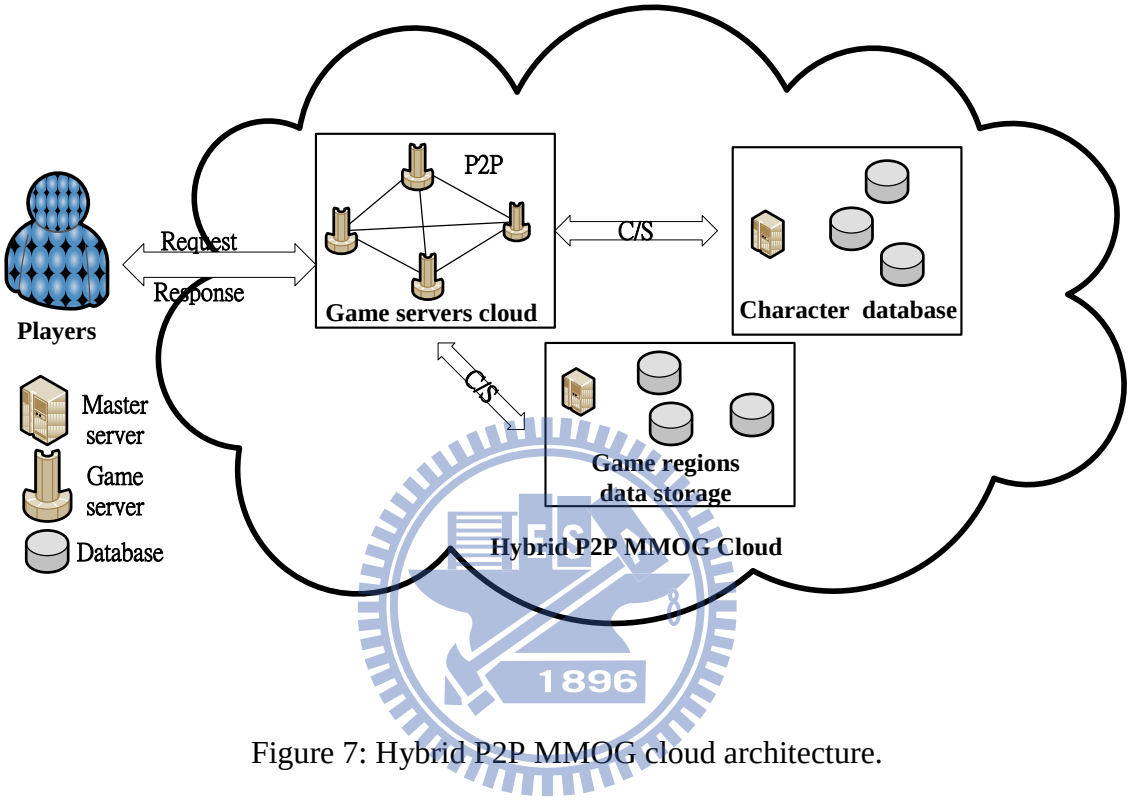

Figure 7: Hybrid P2P MMOG cloud architecture.

4.2 Game servers cloud

In MMOG environments, there are huge messages, especially communication messages between players. Game servers use a P2P protocol to exchange load information and game region data. Game servers also use the P2P protocol to exchange the information of players. By P2P flooding, each game server can know the other game servers' load. When a game server is overloaded, it will migrate some players to other game servers by the proposed game server load management procedure.

4.3 Multi-threshold load management

In the game servers cloud, we allocate VMs from game servers to service game regions. Game regions do not setup on specific virtual machines. In this way, it could help the system to get high scalability. We do not use a single master server in the game servers cloud, Game servers communication is based on a P2P protocol. Game servers can exchange load information by flooding in the P2P protocol. Game servers can perform load management by themselves, and a game server is also in charge of creating accounts for players when they login for the first time. This way can help resource allocation more efficient.

The multi-threshold load management focuses on VMs management. We use four thresholds (T_1, T_2, T_3, and T_4) to define five loading layers (L_1, L_2, L_3, L_4, and L_5) for each VM. T_1 is the lower bound of VM load. T_2 and T_3 define two preferred loading layers to service players. T_4 is the upper bound of VM load. As shown in Figure 8, each loading layer represents a different load and status.

- L_1 (Light): For a VM_i operating in this layer, its players will be migrated to another VM_j serving the same region in L_3, L_2, or L_4 (selection order). After the players in VM_i are migrated out, VM_i will be released.
- L_2 (Low): For a VM_i with maximal load in this layer, it can provide capacity for a VM_j in L_5 to migrate its players to this VM_i.
- L_3 (Medium): It is the optimal layer for VMs to stay.
- L_4 (High): For a VM_i operating in this layer, its loading is slightly high, but is tolerable.
- L_5 (Heavy): For a VM_i operating in this layer, part of its players will be migrated to another VM_j serving the same region in L_2, L_3, or L_1 (selection order) if there are available VMs in these layers; If there are no VMs in these layers, the game server load management will be executed.

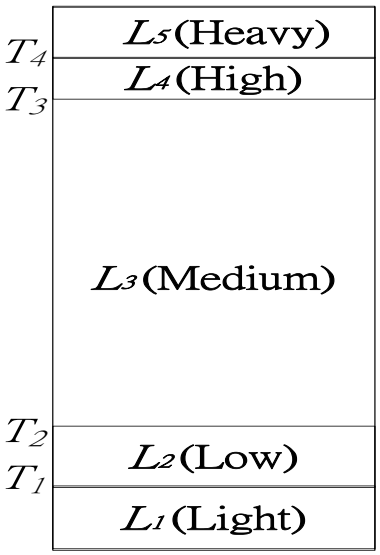

Figure 8: Multi-threshold load management.

Figure 9 and Figure 10 show the flow chart of the multi-threshold load management algorithm. The details of the multi-threshold load management algorithm are described in the following.

1) We monitor the VMs in each game region and sort the VMs in each game region according to their loadings in an ascending order.
2) If the VM in L_5 is empty, go to step 7. Otherwise, go to step 3.
3) In this step, we know some VMs exist in L_5. We check if there are VMs in L_2. If there are VMs in L_2, we migrate VM_i with maximal load in L_5 to the VM_j with minimal load in L_2, and then go to step 2. Otherwise, go to step 4.
4) We check if there are VMs in L_3. If there are VMs in L_3, we migrate VM_i with minimal load in L_5 to the VM_j with minimal load in L_3, and then go to step 2. Otherwise, go to step 5.
5) We check if there are VMs in L_1. If there are VMs in L_1, we migrate VM_i with minimal load in L_5 to the VM_j with minimal load in L_1, and then go to step 2. Otherwise, go to step 6.

6) Executing the VM creating procedure to create a VM to share the load in this game region.

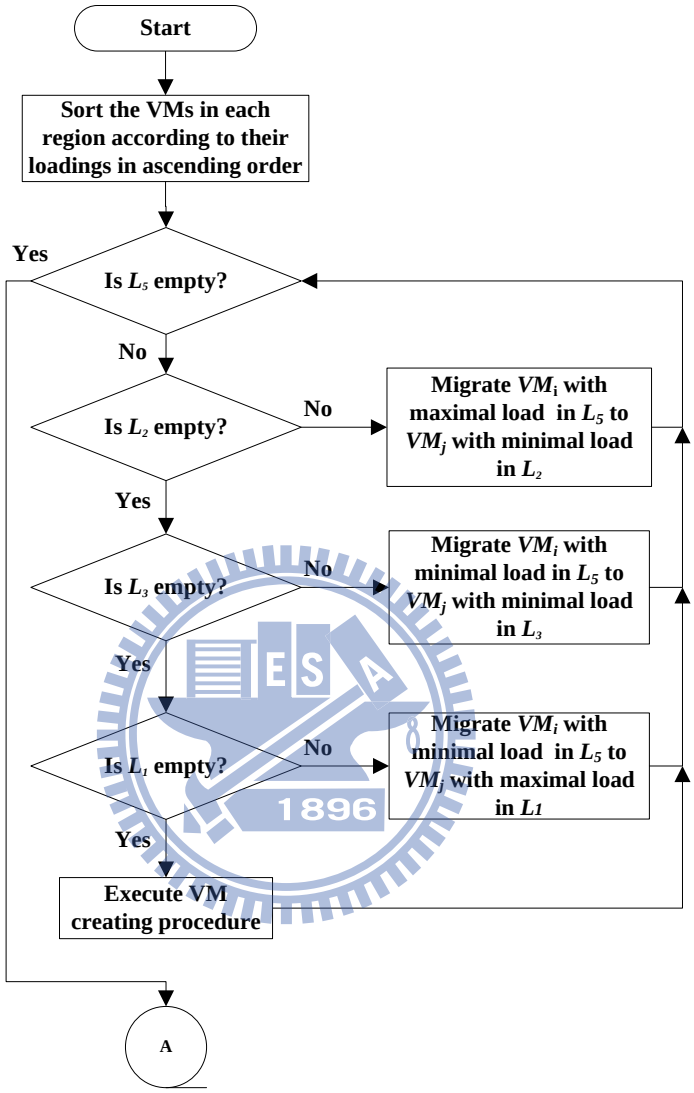

Figure 9: Multi-threshold load management algorithm (1/2).

7) In this step, we start to release VMs. Check if there is any VM in L_1. If there is no VM in L_1, exit this procedure. Otherwise, go to step 8.

8) We check if there are VMs in L_3. If there are VMs in L_3, we migrate VM_i with minimal load in L_1 to the VM_j with minimal load in L_3, release VM_i, and then go to step 7. Otherwise, go to step 9.

9) We check if there are VMs in L_2. If there are VMs in L_2, we migrate VM_i with minimal load in L_1 to the VM_j with maximal load in L_2, release VM_i, and then go to step 7. Otherwise, go to step 10.

10) We check if there are VMs in L_4. If there are VMs in L_4, we migrate VM_i with minimal load in L_1 to the VM_j with maximal load in L_4, release VM_i, and then go to step 7. Otherwise, exit the procedure.

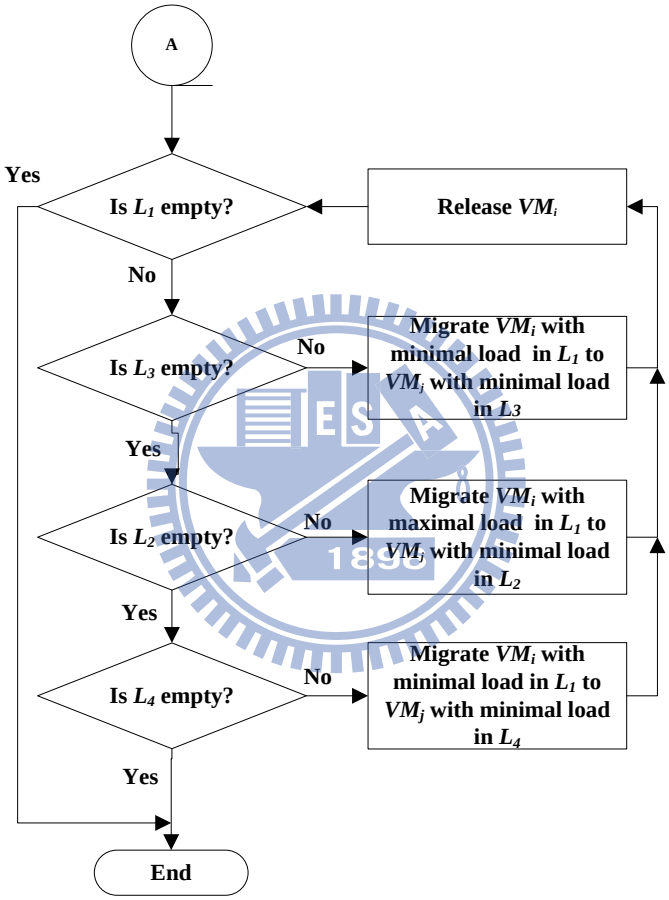

Figure 10: Multi-threshold local load management algorithm (2/2).

When a game region needs a new VM, the VM creating procedure is executed, as shown in Figure 11. Firstly, the game server checks its resource. If it has enough resources, the game server will create a new VM to service the game region. If not, the game server will turn down the VM creating request. The game server executes the game server load management procedure to transfer players to another game server.

In the game server load management procedure, we mainly consider the physical distance between players and game servers, because players need to be served with low response time. As shown in Figure 12, firstly we find a neighboring game server who is serving the same game region and has the lowest load. If not available, we select a neighboring game server that is not serving the same game region and has the lowest load. If not available, we select a distant game server that has capacity to service more players. The selected game server will service the migrated players. Finally, if we could not find a game server who has capacity to service more players, then it means all game servers are overloading.

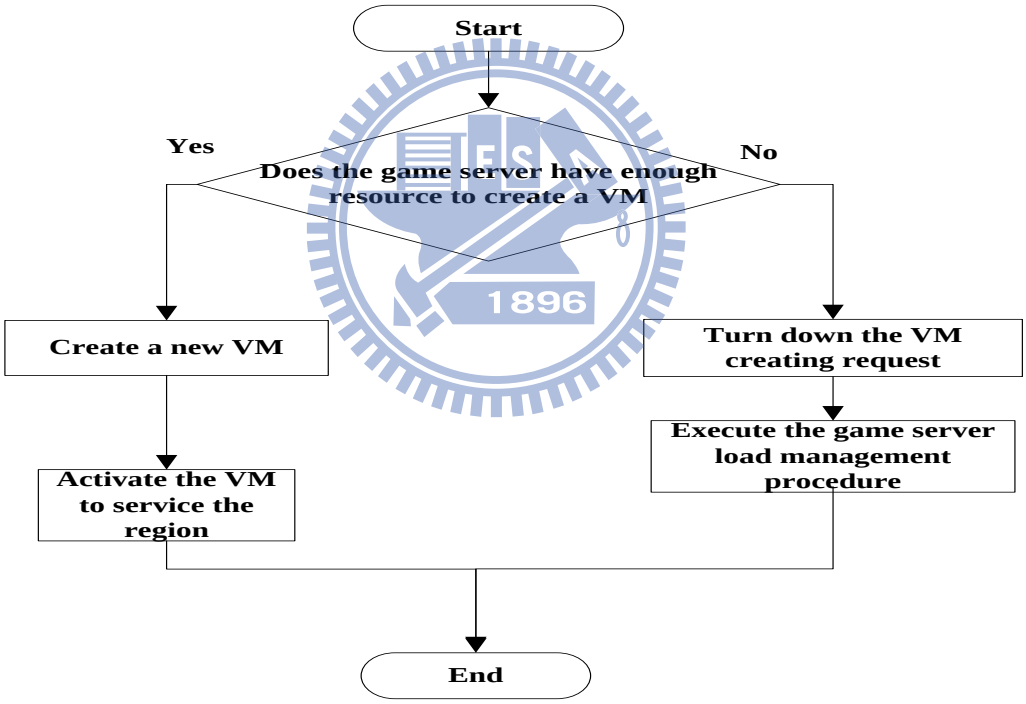

Figure 11: VM creating procedure.

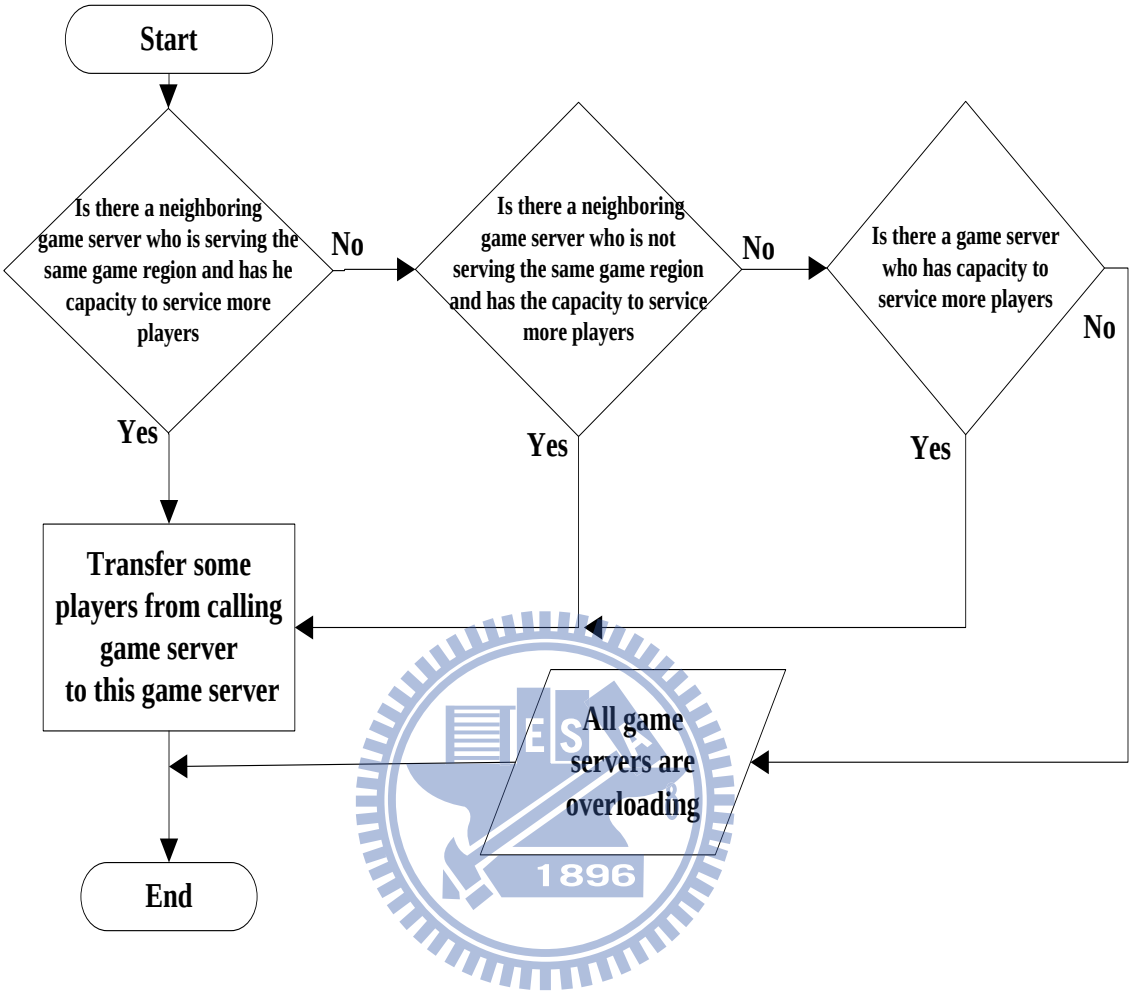

Figure 12: Game server load management procedure.

Chapter 5

Performance Evaluation

In this chapter, we first describe simulation setup and evaluation metrics. Then, we evaluate the proposed hybrid P2P MMOG in terms of *average response time,* and *average deadline miss ratio.*

5.1 Simulation setup and evaluation metrics

We use CloudSim [27] to simulate players' behaviors by sending service requests to game servers. Players' requests were collected from *Stendhal MMORPG* using Wireshark. The Stendhal MMORPG is running on Intel i7 2.93 GHz CPU with 512 MB RAM and 100 Mbps Ethernet. The average service time to serve a request is 0.2 *ms*. There are three game servers and the upper bound of the response time is set to *300 ms* according to [30]. Each game server can create at most 40 VMs. The capability of a VM is 2000 MIPS with 1024 MB RAM and 100 Mbps Ethernet (P3 capability). Each VM can support at most 59 players (obtained from CloudSim). The total number of game regions is 30. In addition, a game region is served by at least one VM, if there are players.

5.1.1 Average response time

The *average response time* is defined as the elapsed time between the time a player sends a request to the time that a player actually receives the response, which is formulized as follows:

$$T_{avg} = \frac{1}{mp} \sum_{k=1}^{m} \sum_{j=1}^{p} \sum_{i=1}^{n_j} \frac{TR_i - TS_i}{n_j}$$

where T_{avg} = average response time

TR_i = the time interval for player to receive a response, and i is the i^{th} request.

TS_i = the time interval for player j to send a request, and i is the i^{th} request.

p = the total number of players

n_j = the total number of requests for player j

k = the k^{th} round

5.1.2 **Average deadline miss ratio**

The *average deadline miss percentage ratio* is defined as the response time being more than a real time bound, and we set the real time bound to 300 *ms* according to [32], which is formulized as follows:

$$\mathrm{D_{avg}} = \frac{\sum_{j=1}^{m} \frac{RD_j}{RT_j}}{m}$$

where D_{avg} = average deadline miss ratio

RD_j = the total number of responses that miss the deadline in a game server j

RT_j = the total number of responses in a game server j

j = the j^{th} game server

5.2 Comparison between multi-server architecture and proposed hybrid P2P cloud architecture

Figure 13 shows the average response time of the multi-server architecture and proposed hybrid P2P cloud architecture. The multi-server architecture has the lower response time initially because a regional server's capacity is higher than a VM's. When the number of players increases to 3490, the multi-server architecture has higher response time. This is because its regional servers serve specific game regions. As a result, players are queued up for a specific regional server. In our architecture, each VM has lower capacity, but our architecture has higher scalability. VMs can be migrated to a busy game region and the response time of players can be reduced. Because of this, our architecture performs better after the number of players exceeding 3490.

Figure 14 shows the deadline miss ratio between the multi-server architecture and proposed hybrid P2P cloud architecture. In the multi-server architecture, it has the lower deadline miss ratio under lower number of players. It shows that with enough resources in the multi-server architecture, the multi-server architecture can have good performance. The proposed architecture has a smaller deadline miss ratio under medium to high load than the multi-server architecture.

Figure 15 shows the average response time of the proposed hybrid P2P cloud architecture under different thresholds settings. We set different thresholds to T_1, T_2, T_3 and T_4. Before the number of players reaches 6400, the 0.2-0.3-0.7-0.8 setting has lower average response time. This is become that new VMs are created to service players under lower load (0.8). After the number of players is over 6400, the average response time for this setting becomes higher than that of the other setting. Because VMs are activated early to service game regions, there may be no VM to service really hotspot game regions later.

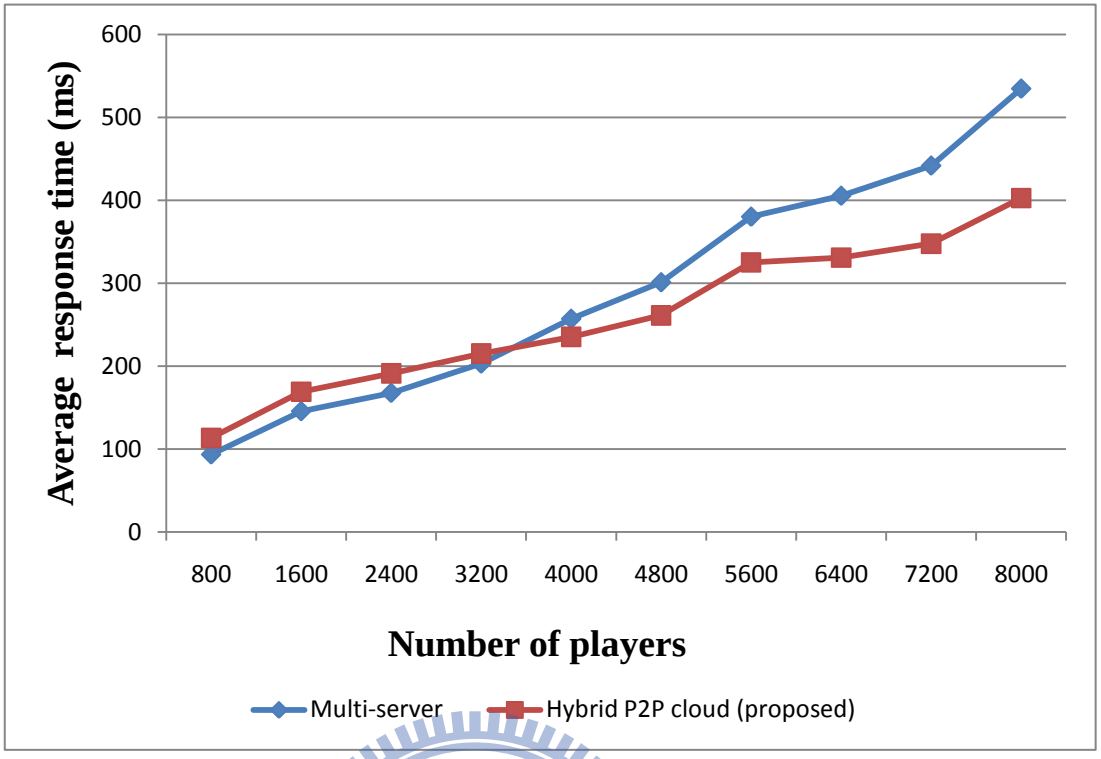

Figure 13: Average response time between multi-server and proposed hybrid P2P cloud architecture.

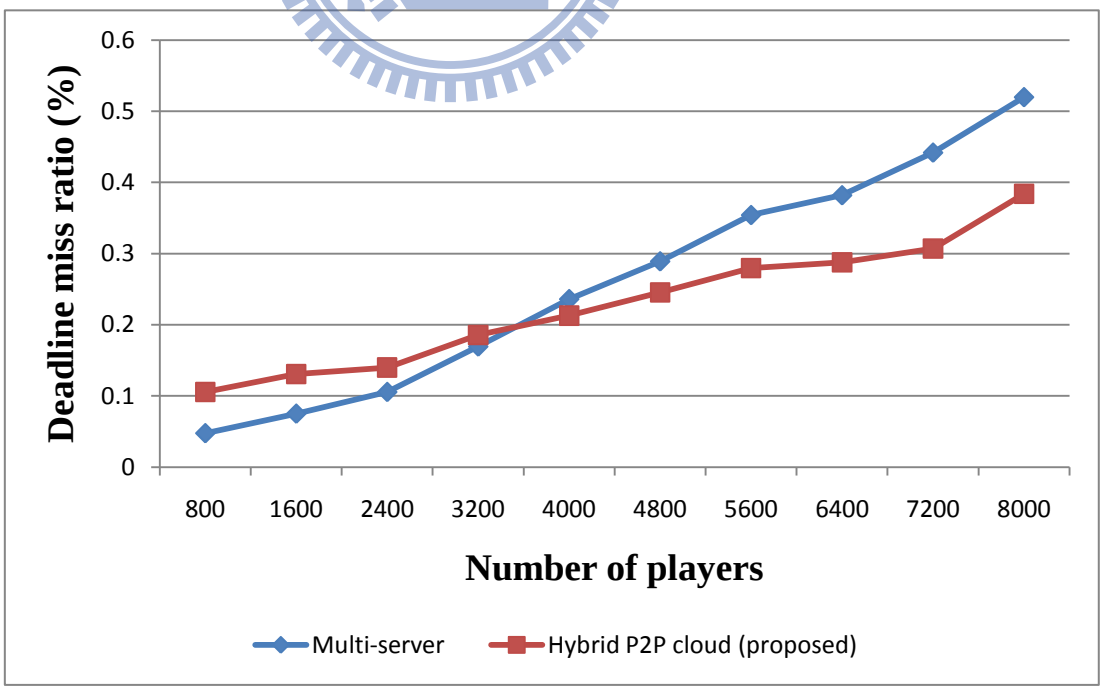

Figure 14: Deadline miss ratio between multi-server and proposed hybrid P2P cloud architecture.

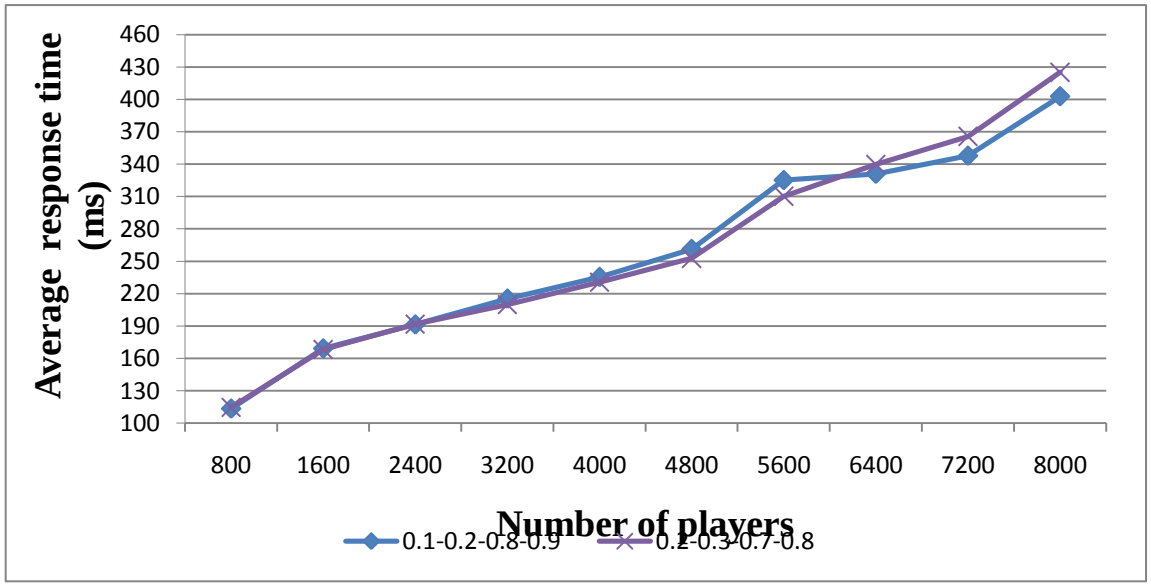

Figure 15: Average response time under different thresholds of load for the proposed architecture.

Chapter 6

Conclusion

6.1 Concluding remarks

In this paper, we propose a hybrid P2P cloud architecture which includes multi-threshold load management and game server load management for MMOG cloud environments. The multi-threshold load management can make the utilization of virtual machines more efficient. The game server load management transfers some players from an overloaded game server to a neighbor game server with available capacity. The proposed hybrid P2P cloud architecture can support 10.31% more players under no deadline (*300* ms) miss compared to the multi-server architecture. The proposed hybrid P2P cloud architecture can reduce the average response time by 20.6% compared to the multi-server architecture under medium to high load through flexible allocation of resources (virtual machines). The proposed architecture also has a 27.9% smaller deadline miss ratio than the multi-server architecture under medium to high load.

6.2 Future work

We may apply the proposed hybrid P2P cloud architecture to multi-game environments for flexible allocation of resources. We will integrate an efficient load prediction scheme with the proposed hybrid P2P cloud architecture to achieve more efficient allocation of resources so as to further reduce the deadline miss ratio and average response time.

Bibliography

[1] G. Dolbier and A. Goldschmidt, “The business of interactive entertainment,” in *Proceedings of the IBM Digital Media Solutions*, May 2006.

[2] E. Cronin, B. Filstrup, and A. Kurc, “A distributed multiplayer game server system,” *Technical Report the University of Michigan*, May 2001.

[3] A. Shaikh, S. Sahu, M.-C. Rosu, M. Shea, and D. Saha, “On demand platform for online games,” *IBM Systems Journal*, vol. 45, no. 1, pp. 7–20, 2006.

[4] B. Hack, M. Morhaime, J.-F. Grollemund, and N. Bradford, “Introduction to vivendi games,” [Online] Available: http://www.vivendi.com/, Jun 2006.

[5] S. Sukhyun, et al., "Blue Eyes: Scalable and reliable system management for cloud computing," in *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing*, pp. 1-8, 2009.

[6] C.Majewski, C. Griwodz, and P. Halvorsen, “Translating latency requirements into resource requirements for game traffic,” in *Proceedings of the International Network*, pp. 113-120, 2006.

[7] J. Slegers, I. Mitrani, and N. Thomas, "Evaluating the optimal server allocation policy for clusters with on/off sources," *the Performance Evaluation*, vol. 66, pp. 453-467, 2009.

[8] R. Stanojevic and R. Shorten, "Load balancing vs. distributed rate limiting: an unifying framework for cloud control," in *Proceedings of the IEEE International Conference on Communications*, pp. 1-6, 2009.

[9] W. Streitberger and T. Eymann, "A simulation of an economic, self-organising resource allocation approach for application layer networks," *the Computer Networks*, vol. 53, pp. 1760-1770, 2009.

[10] Y. Lai and S. ZhongZhi, "An efficient data mining framework on Hadoop using Java persistence API," in *Proceedings of the IEEE 10*th *International Conference on Computer and Information Technology*, pp. 203-209, 2010.

[11] Z. Liang-Jie and Z. Qun, "CCOA: Cloud Computing Open Architecture," in *Proceedings of the IEEE International Conference on Web Services*, pp. 607-616, 2009.

[12] V. Nae, A. Iosup, and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, pp. 1-1, 2010.

[13] Torque MMO Kit - Server Architecture http://www.mmoworkshop.com/trac/mom/wiki/ServerArchitecture

[14] A. E. Rhalibi and M. Merabti, "Interest management and scalability issues in P2P MMOG," in *Proceedings of the Consumer Communications and Networking Conference*, pp. 1188-1192, 2006.

[15] X.-B. Shi, D. Yang, L. Du, and Z.-W. Wang, "Research on service management techniques for P2P MMOG," in *Proceedings of the International Conference on Internet Technology and Applications*, pp. 1-4, 2010.

[16] C. Yang, W. Tianyu, and L. Jianxin, "An efficient resource management system for on-line virtual cluster provision," in *Proceedings of the IEEE International Conference on Cloud Computing*, pp. 72-79, 2009.

[17] G. Zhenhuan, P. Ramaswamy, G. Xiaohui, and M. Xiaosong, "SigLM: signature-driven load management for cloud computing infrastructures," in *Proceedings of the 17*th *International Workshop on Quality of Service*, pp. 1-9, 2009.

[18] K. Il Kon, Z. Pervez, A. M. Khattak, and L. Sungyoung, "Chord based identity management for e-healthcare cloud applications," in *Proceedings of the 2010 10*th *IEEE/IPSJ International Symposium on Applications and the Internet*, pp. 391-394, 2010.

[19] H. Chen and C. Cao, "Research and application of distributed OSGi for cloud computing," in *Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering*, pp. 1-5, 2010.

[20] P. Zhao, T.-l. Huang, C.-x. Liu, and X. Wang, "Research of P2P architecture based on cloud computing," in *Proceedings of the International Conference on Intelligent Computing and Integrated Systems*, pp. 652-655, 2010.

[21] C. Carter, A. E. Rhalibi, M. Merabti, and A. T. Bendiab, "Hybrid client-server, peer-to-peer framework for MMOG," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1558-1563, 2010.

[22] V. Nae, R. Prodan, and T. Fahringer, "Cost-efficient hosting and load balancing of Massively Multiplayer Online Games," in *Proceedings of the IEEE/ACM International Conference on Grid Computing*, pp. 9-16, 2010.

[23] Q. Zhaoyang and W. Yanguang, "The design of the substation simulation model of distributed virtual environment," in *Proceedings of the Second International Symposium on Computational Intelligence and Design*, pp. 249-252, 2009.

[24] Su Min Jang and Jae Soo Yoo, "An efficient Load balancing mechanism in distributed virtual environments," *the ETRI Journal*, Vol.30, NO.4, PP.618-620, 2008.

[25] B. Hariri, S. Shirmohammadi, and M. R. Pakravan, "A distributed topology control algorithm for P2P based simulations," in *Proceedings of the Distributed Simulation and Real-Time Applications, IEEE International Symposium*, pp. 68-71, 2007.

[26] V. Nae, A. Iosup, S. Podlipnig, R. Prodan, D. Epema, and T. Fahringer, "Efficient management of data center resources for Massively Multiplayer Online Games," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-12, 2008.

[27] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services http://www.buyya.com/gridbus/cloudsim/

[28] E. Cronin, B. Filstrup, A.R. Kurc, and S. Jamin, “An efficient synchronization mechanism for mirrored game architectures," *in Proceedings of the 1st workshop on Network and system support for games*, pp.67-73, ACM Press, 2002.

[29] E. Cronin, B. Filstrup, and A. Kurc, “A distributed multiplayer game server system," University of Michigan Course Project Report, May 2001.

[30] M. Mauve, S. Fischer, and J. Widmer, “A generic proxy system for networked computer games," *in Proceedings of the 1st workshop on Network and system support for games*, pp.25-28, ACM Press, 2002.

[31] D. Bauer, S. Rooney, and P. Scotton, “Network infrastructure for massively distributed games," *in Proceedings of the 1st workshop on Network and system support for games*, pp.36-43, ACM Press, 2002.