A Scalable Architecture for Massive Multi-player Online Games Using Peer-to-Peer Overlay

Jia Luo*, Huiyou Chang**
*Department of Information Science and Technology, Sun Yat-sen University, Guangzhou, China
**Department of Software, Sun Yat-sen University, Guangzhou, China
is01lj@163.com, isschy@mail.sysu.edu.cn

***Abstract*—Today, Massive Multi-player Online Game (MMOG) has taken an important part in the field of entertainment. But its increasing requirement of resources has brought a heavy burden to the game server. To address this problem, we propose a Structured Multi-Agent (SMA) model which divides the resources into groups by the Area of Interest (AOI) and builds itself on the pastry protocol. Then, every other agent can get the resources from the manager located by the pastry. Moreover, an algorithm called Assignment of Resource Processing License (ARPL) is implemented to constitute the core of SMA model. This algorithm guarantees the consistency of all resources by assigning events to the agents who have the processing licenses of resources. Through comparing with another P2P MMOG model, fine performance is indicated on SMA.**

Keywords- MMOG; Peer to Peer; Area of Interest; Interest Management; Pastry protocol.

I. INTRODUCTION

Nowadays, the Massive Multi-player Online Game (MMOG) is part of the most popular entertainment that attracts millions of people to take part in. However, for holding such a huge number of players, many servers are consumed to maintain the computing and storage of resources in the virtual world. In addition, the contradiction between increasing requirement of resources and limited load capacity of servers gradually became apparent. Therefore, many approaches based on P2P are proposed to address this problem.

Actually, each resource in the game world has an affected zone. It interacts only with resources adjacent. For example, a player is able to see the objects nearby and interact with them directly. But a bird thousands of miles away has no sense to the player. Thence, to distribute the resources, we divide the whole space into many zones called area of interest (AOI) [10] and map all resources in an AOI to a node for management. In this way, all events happened are localized to an AOI. When some nodes want to interact with resources, they just need to use the same mapping algorithm to locate the holder who maintains the resources in the AOI. Then, connect to it and download resources. This approach can solve the contradiction stated above, but it also involves some challenging issues [2] such as consistency, scalability and security. Consistency is the prerequisite for the correctness of game logic. Since the state of a resource can be stored in different nodes, changes in one state should be synchronized to other states of the same resource. If two events simultaneously happen on a resource in different nodes, it will cause the inconsistency and affect the fairness of game. Therefore, we should guarantee the uniqueness of modification on states of resources. Furthermore, in order to take full advantage of the computing power of all nodes, we need to assign resources evenly to these nodes. In this case, the game will be able to maintain more resources and process more events. Finally, because of the distributed storage and computing, there are many tricks of cheating existing in P2P MMOG. For protecting the benefits of players, some detect mechanisms would be adopted to prevent the game from cheating. In this paper, our main discussion focuses on the former two issues. The security will be the research themes in the future.

For distributing and locating resources, we construct our Structured Multi-Agent (SMA) model on the pastry protocol [21]. Pastry is a scalable distributed object location and routing protocols that can be used to build large-scale P2P system. Every resource in this system has been assigned an object identifier through the hash algorithm and released to a node whose identifier is closest to the object identifier. Generally, all the resources in an AOI are released to the same node. When a node enters in an AOI, it can get all the resources from the node located by pastry.

To achieve the consistency and scalability, we propose a load balancing approach called Assignment of Resource Processing License (ARPL) algorithm. This algorithm assigns resources to the nodes evenly according to their performance and guarantees the uniqueness of assignment for event processing. Meanwhile, we implement another three algorithms including node joining algorithm, neighbor discovery algorithm and node failure algorithm to form a scalable and stable system. Experiment shows that, SMA has a significant advantage in performance and scalability.

II. RELATED WORKS

There are several directions of researches in P2P MMOG: Decentralized Unstructured Topology (DUT), Decentralized Structured Topology (also called DHT), Hybrid Topology and Multi-layer Structure. In DUT, lockstep [3], Asynchronous Synchronization [3], New Event Ordering [4] are proposed. These approaches mainly solve the cheating in games. They are allowed to communicate directly among nodes for fast interaction. But the consistency of resources can't be guaranteed effectively

because of the unstructured feature. Also, the nodes are hard to be organized. So, we focus on the DHT which provides a better way to manage resources.

The approaches based on DHT usually assign a collaborator to manage the AOIs. We name these approaches the Structured Single Collaborator (SSC) model. For example, T. Hampel [6] and B. Knutsson [5] divide the world into many zones and assign each zone to a collaborator. All events generated by resources and players are sent to this collaborator for processing. Then, they use Scribe, an application level multicast built on top of Pastry, for the game state dissemination. Meanwhile, the collaborators connect to each other for facilitating players to cross the zones. T. Iimura [7] then proposes a mechanism of zoned federation. It assigns a data holder and a manager to catch the states of resources and process events. This method enhances the system stability, and accelerates the speed of access to resources. J. Zhou [8] improves this approach by using Locality Embedded Naming Strategy (LENS) algorithm to choose manager who has the shortest communication time averagely with other nodes. Because the interaction between two adjacent zones has not been taken into consideration in above approaches, the resources can't interact across zones. Still, there are many improvements made on SSC model [9], [11], [18]-[20]. But all these approaches face a problem of scalability. When the number of resources or players increases in an AOI, the collaborator will bear a great burden.

Thus, the hybrid approaches are proposed to achieve the high scalability. This scheme also contains two directions. One is the combination of the server architecture and DUT [13], [14]. But in this paper, the hybrid structure is referred specifically to the combination of DUT and DHT. It organizes resources by the DHT and assigns events to all nodes for processing. Once the state of a resource has a change, the update message will be broadcasted to other nodes in the AOI. Anthony [1] proposes an approach to implement the neighbor discovery algorithm. It allows the collaborators to exchange node information and broadcast resources to all nodes in neighbor AOIs. I. Kazem [12] implements a visibility-driven approach to make the partitioning transparent to users. If a player node has a visibility range surpassing its own zone, it receives necessary messages through a specific communication channel linking it to the other zone's hybrid node, Y. Kawahara [24] describes a fully-distributed scheme where each user keeps track of a fixed number of nearest neighbors. Nodes constantly exchange neighbor list with their own neighbors. After sorting through the list by distance, each node may learn of new nodes and update existing links [22].. In order to enhance the performance of multicast, C. GauthierDickey [23] uses N-Trees for event propagation. This organization allows peers which are close by in the virtual world to order events without needing to communicate with other peers that are farther away. However, all the approaches above are failed to ensure the uniqueness of event processing for a resource. That is, the consistency of states especially stored in nodes in different AOIs can't be guaranteed effectively. So, in this paper, we involve an algorithm called Assignment of Resource Processing License (ARPL). It gives the assurance that events of a resource could only be processed by one node at any time.

III. ARCHITECTURE OF SMA

As presented before, we use pastry for locating resources. That is, every agent can find resources on some manager through pastry. After that, the agent registers itself on the manager and communicates with other agents in the registration list. Then, we use the Assignment of Resource Processing License (ARPL) algorithm to decide how to assign events to these agents in an AOI. For explaining this process in detail, we will first introduce the structure of SMA.

A. Structure of SMA

SMA is composed by three layers. As shown in fig.1, they are pastry interface, cooperation management and game interface. Pastry interface and game interface are abstract layers of pastry application and game logic. They could be implemented according to a certain standard so that any pastry application and game engine can be integrated with them. Cooperation management, as the core part of SMA, is composed by six important modules described as follow:

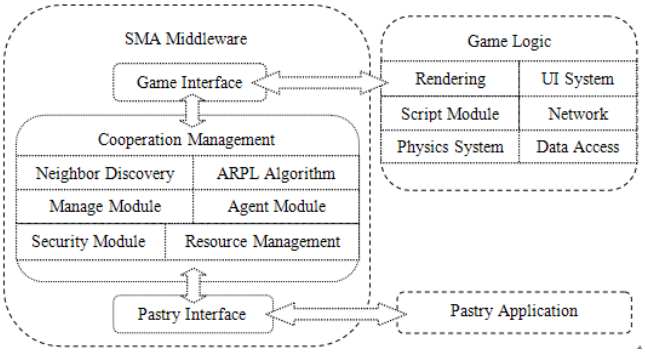

Fig1. Structure of SMA

Among these modules, neighbour discovery is a necessary approach for searching neighbour managers and agents. So any two resources can interact with each other in different AOIs. In later section, we will describe this approach more clearly. Next, the manage module and agent module, corresponding to the functions of manager and agent, are respectively implemented for the node management and events processing. Any joined node in our theory acts as an agent. The manager who executes the manage module is also an agent.

However, besides the manager and agent mentioned before, there is another node we need to involve in our theory for login authentication and states preservation. It's the data server. Though it is a global node provided by game operator, it is quite different from the traditional server which processes all events and maintains resources of whole world. The data sever would be enough to maintain a large number of resources and players.

As we know, there are some agent lists of local AOI and neighbor AOIs cached by the manager. Once a push or pop operation happens on any of these lists, the ARPL algorithm would be executed. In addition, resources are assigned to all agents according to the performance of each node. About the consistency of ARPL, we will detail it later. The resource management and the security module are responsible for resource access and preventing our model from cheating. Since these two modules is not the focus of this paper, we do not discuss them in depth.

B. Procedure of SMA

After understanding the function of each module, we can further describe the entire working procedure of SMA. It is divided into following three parts.

1) Node Joining: When an agent joins in an AOI, it connects to the data server for authentication and downloading user information. Then it uses pastry to locate the manager of local AOI. If the manager hasn't been created, then it will be created and all resources in the AOI will be initialized by the agent. Otherwise, the agent registers itself to the manager and downloads all information of resources from other registered agents. After the communications with other agents are established, they execute the ARPL algorithm so that the new agent can be assigned to process events. When any state of resource changes, it will be updated to all the agents in the AOI.

2) Neighbour Discovery: In order to interact with the resources in neighbour AOIs, we use neighbour discovery approach to find the agents in these AOIs. This course is completed by the manager who also uses the pastry to locate the managers in neighbour AOIs and exchanges the agent list with them. If there isn't an agent in the agent list of neighbour manager, then register all the agents in local AOI to neighbour manager. When an agent joins in the neighbour AOI, all the agents in the neighbour manager will be replaced by the new agent. Anyway, an agent can establish the communications with all other agents in neighbour AOIs and interacts with them or resources updated from them.

3) *Cross-Domain and Failure:* When an agent crosses the border between AOIs, it will inform all the agents in both old and new AOIs to execute the ARPL algorithm. When agents fail in an AOI, all other agents will be notified and execute the ARPL algorithm. Meanwhile, the state of the failed agent will be saved to the data server.

We can see from above procedure that an agent is able to communicate with any other agent in local AOI or neighbor AOIs as long as it wants to. So, the whole chain of connected nodes is very stable.

C. The ARPL Algorithm

In SMA, every resource has a processing license. Only the agent who has been assigned the license can process the corresponding events of this resource. The assignment algorithm is as follow:

$$n = Node[id \% \sum_{i=1}^{n} Q_i]$$
$$Q_i = \left\lceil \sum_{i=0}^{m} W_i R_i \right\rceil, and \sum_{i=0}^{m} W_i = 10 \quad (1)$$

In this formula, we divide the performance of a node into *m* part, such as CPU usage, memory usage, bandwidth usage and so on. W_i represents the weight of each part. R_i is the ratio of real performance and standard performance of each part. For example, the real memory usage is 2G; the standard usage is 1G, then the ratio is 2. Q_i represents the whole performance index of a node. So, we assign to each node the slots by number of Q_i and push then into the array *Node*. After that, the license can be assigned to the node by mapping resource's *id* to the subscript space of array *Node*. It can be seen that, this formula gives a sufficient consideration to the performance of each node and achieves a greater degree of load balancing.

Moreover, there are two types of events defined in SMA model: the atom event and the complex event. Atom event just allows one target to be handled, but complex event handles many. Complex event can be decomposed into a number of atom events. So in this paper, we use atom events to instead of all events. For example, an attack action reduces both the magic value of the attacker and the life value of the victim. This event will be divided into two procedures. Messages in these two procedures will be separately sent to the nodes which have the processing license of the two targets.

Formula (1) makes any resource have only one processor, but in fact, when nodes join in or leave an AOI, there could be a short time that resource may have two processors before licenses are reassigned completed. So, we need a mechanism to avoid this situation. Following are the steps:

1) Synchronization: When the joining event, crossing event or failure event of an agent happens in an AOI, all agents will be notified immediately. Then, they broadcast confirmation messages to others and stop processing events. However, they could still keep saving the arrived events to a special buffer. After receiving all the confirmation messages from others, they generate the *Node* array in (1) and start to process events.

2) Mechanism of ARPL: When an agent wants to interact with any resource, it will easily get the agent by mapping the *id* of resource to the subscript space of *Node* array and send the events to the agent. Moreover, the events saved in the special buffer will be reallocated to the correct agents first.

3) Behaviour of Resources: In P2P MMOG, there are thousands of resources which have intelligence. For example, a patrol warrior, an attacking beast or a grocer, each of them has its own behaviour. In order to maintain the AI of a resource, we also assign the agent who has its

processing license to simulate its behaviour. Therefore, the state of resource is still changed by one agent. The pseudo-code of ARPL is as follow:

```
1   for each e in event list;
2     if e is not joining, crossing or failure event
3       calculate the agent who should process e
4       if the agent is self
5         process event and update state
6       else
7         send event to the agent;
8     else
9       send confirmation message;
10      generate node array;
11      break;
12  if received all confirmation messages
13    go to 1
14  else
15    go to 12
```

In this program, ARPL is defined as a callback function called when the agent has received all confirmation messages. The processing of events and synchronization are executed alternately. So, the state of any resource would never be changed in two agents simultaneously. That is, ARPL algorithm can guarantee the consistency of all resources.

D. Bandwidth Inprovement

Any state of resource can be divided into an inner state and an outer state. The inner state is used for calculating and the outer state is used for display. So, we have following formula:

$$s_i = s_j \Leftrightarrow i = j,\ s_i \subseteq s \text{ and } s_j \subseteq s \qquad (2)$$

It means two states are equal if and only if they are both the subset of state of a resource; s is the complete state of a resource.

For every node i, there is a global state which preserves the states of all resources in local AOI and neighbor AOIs. The collection of these states is marked as:

$$GS = \{s \mid s \in \bigcup_{j=k1}^{kn} S_j\} = \bigcup_{j=0}^{n} s_j^o \cup \bigcup_{j=0}^{n'} s_j^i \qquad (3)$$

In (2), k_i is the key of local AOI and neighbor AOIs; S_j is the collections of states in an AOI; $\bigcup_{j=0}^{n} s_j^o$ is all the outer states in local AOI and neighbours; $\bigcup_{j=0}^{n'} s_j^i$ is some inner states in local AOI.

For most of the agents who don't have the processing license of a resource also do not have to preserve the inner state of that resource. They just need the outer state for display. Thence, we improve our approach of state update by sending the outer states to other agents.

However, when the ARPL algorithm is executed, the inner states of some resources whose processing licenses are changed will be transferred to the corresponding agents. There is an exception. All the inner states preserved in an agent will be lost when the agent fails. Under such circumstances, the manager would be a cache of all inner states in the AOI. So, anyone can access these inner states from manager at any time.

It is obviously that this improvement significantly reduces the data transmission among agents. Though the cost of storage in manager increased, as we know, the manager is an agent who also has a reduction in storage. Therefore, average cost of storage in manager will not increase.

IV. EXPERIMENT

In order to prove the advantage of SMA, we conduct an experiment comparing with the SSC model proposed in [5]. The experimental environment provides 100 computers. As our purpose is to explain the performance of SMA, the applications based on Pastry have not been introduced into this experiment. Instead, we use the data server to be an id generator and resource locator.

In the game logic, we just implemented a visual tool to display the entities of clients and resources. The network connections between nodes are created by using the Raknet library. Following is the curve of CPU usage when number of nodes increases.

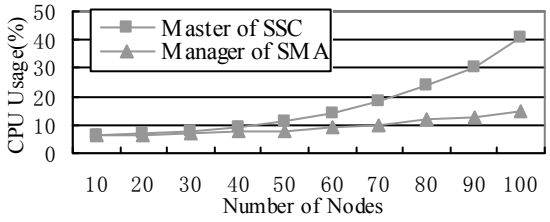

Fig.2 CPU usage when number of nodes increases

We can see from fig.2 that when there are few nodes in an AOI, the CPU usages of SMA and SSC are quite the same. Once the number of nodes increases, the gap between the two curves becomes apparent.

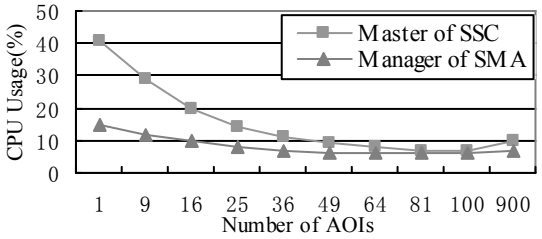

Fig.3 CPU usage when number of AOIs increases

Fig.3 explains the importance of AOI division in P2P MMOG. It greatly reduces the unnecessary communications among nodes. However, the number of AOIs also has a

limit. As shown in fig.3, when the number of AOIs is more than 900, the CPU usage will increase because of the frequent cross domain operations.

It can be seen from this experiment that SMA has a good scalability. Using this model, we will be able to accommodate more resources and players in the game.

V. CONCLUSION

In this paper, we proposed a load balancing model called SMA for P2P MMOG. It uses the ARPL algorithm to assign the resources averagely to all agents according to their performance. In addition, this algorithm effectively guaranteed the consistency of resources. Experiment show that, SMA has a nice performance compared with other models.

However, there are some problems we haven't solved at present. First, the states of resources will be stored only in the manager when no agent left in local AOI and neighbor AOIs. If this manager fails, these states will be lost. Any agent who returns to this AOI will only get the initial resources. So, we need a solution to save these states, even to maintain the behalves of them. Second, we haven't mentioned any idea to prevent our model from cheating such as fixed-delay cheat, timestamp cheat, collision cheat and so on. These two problems would be the main researches in our future work.

ACKNOWLEDGMENT

The authors are grateful to Y. Q. Chen, J. Zhou, and B. D. Liu of Sun Yat-sen University for their helpful insight.

REFERENCES

[1] A. Yu, and S. T. Vuong. "MOPAR: A Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games". in *Proc. the international workshop on Network and operating systems support for digital audio and video*, *ACM* , 2005, pp. 99-104

[2] C. Neumann, N. Prigent, and M. Varvello, "Challenges in Peer-to-Peer Gaming". *ACM SIGCOMM,* vol.37, pp. 79-82, 2007.

[3] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and distributed online games," in *Proc. IEEE INFOCOM* 2001, pp. 104–113

[4] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low latency and cheat-proof event ordering for peer-to-peer games," in *Proc of the 14th international workshop on Network and operating systems support for digital audio and video. Cork .ACM*. 2004, pp. 134-139

[5] B. Knutsson, H. Lu, and W. Xu, "Peer-to-Peer Support for Massively Multiplayer Games," *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 96-107

[6] T. Hampel, T. Bopp, and R. Hinn, "A Peer-to-Peer Architecture for Massive Multiplayer Online Games," in *Proc. 5th ACM SIGCOMM workshop on Network and system support for games*, Singapore, 2006, pp. 1-4

[7] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multi-player Online Games," in *Proc of 3rd ACM SIGCOMM workshop on Network and system support for games*. *ACM* 2004, pp. 116-120

[8] J. Zhou, L. Tang, and K. Li, "A Low Latency Peer to Peer Approach for Massively Multiplayer Games," *Springer Verlag Berlin Heidelberg*, 2006, vol. 4118, pp. 120-131

[9] S. Douglas, E. Tanin, and A. H arwood, "Enabling massively multiplayer online gaming applications on a p2p architecture," in *Proc of the International Conference on Information and Automation. Colombo, IEEE* 2005. pp. 7–12

[10] D. T. Ahmed, and S. Shirmohammadi, "A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs." in *Proc. 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications. IEEE Computer Society*, 2008, pp.27-34

[11] J. Kienzle, C. Verbrugge, and B. Kemme. "Mammoth: A Massively Multiplayer Game Research Framework". in *Proc. the 4th International Conf. on Foundations of Digital Games ACM*. April. 2009, pp.308-315

[12] I. Kazem, D. T. Ahmed, and S. Shirmohammadi. "A Visibility-Driven Approach to Managing Interest in Distributed Simulations with Dynamic Load Balancing". in *Proc. 11th IEEE Symposium on Distributed Simulation and Real-Time Applications. IEEE Computer Society*, 2007, pp.31-38

[13] A. E. Rhalibi, M. Merabti, "Agents-Based Modeling for a Peer-to-Peer MMOG Architecture" *ACM Computers in Entertainment,* Vol. 3, No. 2, April 2005, pp.3-21

[14] A. E. Rhalibi, M. Merabti, Y. Shen. "AoIM in Peer-to-Peer Multiplayer Online Games." in *Proc. ACM SIGCHI international conference on Advances in computer entertainment technology*, June, 2006, Hollywood, California, USA. Article No.71

[15] N. Singh, S. Sudarshan, "Distributed Event Delivery Model for Collaborative Virtual Simulations." in *Proc. 2008 ACM CoNEXT Student Workshop*, December 9, 2008, Madrid,SPAIN. Article No. 30

[16] X. Jiang, F. Safaei, P. Boustead. "Enhancing the multicast performance of structured P2P overlay in supporting Massively Multiplayer Online Games" in *15th IEEE internation Conf. on Networks ICON* 2007, pp.124-129

[17] S. Ferretti, M. Roccetti "Fast Delivery of Game Events with an Optimistic Synchronization Mechanism in Massive Multiplayer Online Games" in *Proc. ACM SIGCHI International Conference on Advances in computer entertainment technology*, Valencia, Spain. 2005, pp.405-412

[18] R. Balan, K. Ebling,, M. P. Castro, and A. Misra. "Matrix: Adaptive Middleware for Distributed Multiplayer Games" in *Proc ACM/IFIP/ USENIX International Conf. on Middleware, LNCS* 3790, Grenoble, France, 2005, pp. 390–400

[19] L. Fan, H. Taylor, P. Trinder. "Mediator: A Design Framework for P2P MMOGs" in *Proc. 6th ACM SIGCOMM workshop on Network and system support for games,* September 19-20, 2007, Melbourne, Australia, pp.43-48

[20] Y. He, Y. Zhang, and J. Guo. "On Mitigating Network Partitioning in Peer-to-Peer Massively Multiplayer Games" *LNCS 3619*, 2005, Springer Berlin / Heidelberg, pp. 481–490

[21] A. Rowstron1, and P. Druschel. "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems" in *Proc. 18th IFIP/ACM International Conf. on Distributed Systems Platforms*. Heidelberg, Germany, November 2001, pp.329-350

[22] S. Y. Hu, G. M. Liao. "Scalable Peer-to-Peer Networked Virtual Environment" in *Proc. of 3rd ACM SIGCOMM workshop on Network and system support for games*, 2004, Portland, pp.129-133

[23] C. GauthierDickey, V. Lo, D. Zappala. "Using N-Trees for Scalable Event Ordering in Peer-to-Peer Games" in *Proc. of the international workshop on Network and operating systems support for digital audio and video*, June 13–14, 2005, Stevenson, Washington, USA, pp.87-92.

[24] Y. Kawahara, T. Aoyama, H. Morikawa. "A Peer-to-Peer Message Exchange Scheme for Large-Scale NVEs." *Telecommunication Systems* Vol. 25, pp 353–370, 2004.