

Measurement-based Peer-to-Peer Grouping for Networked Virtual Environment

Hajime Sogawa
Dep. of Info. and Comm. Eng.
Tokyo Denki University, Japan
hajime@unl.im.dendai.ac.jp

Niwat Thepvilojanapong
Dep. of Info. and Comm. Eng.
University of Tokyo, Japan
wat@mcl.iis.u-tokyo.ac.jp

Hiroki Saito
Dep. of Info. Systems
and Multimedia Design
Tokyo Denki Univ., Japan
hsaito@unl.im.dendai.ac.jp

Kaoru Sezaki
Center for Spatial Information Science
University of Tokyo, Japan
sezaki@iis.u-tokyo.ac.jp

Yoshito Tobe
Department of Information Systems
and Multimedia Design
Tokyo Denki University, Japan
yoshito@unl.im.dendai.ac.jp

Abstract

Networked games are newly emerging and increasing applications in the Internet community. Multiplayer Online Game (MOG) is one of such applications that can accommodate many users simultaneously. To realize MOGs in a distributed peer-to-peer environment, we propose a novel topology construction scheme that maps each player's coordinate in the real-world network to the coordinate in a virtual plane. The main features of our scheme is two-stage placement of a node: rough estimation and refinement. We also propose a key idea of super nodes in order to achieve scalability and efficiency. This paper describes the system design of our scheme.

1. Introduction

Multiplayer Online Game (MOG) is a general term for an online game which is simultaneously played by a number of players via the Internet. The examples of MOGs are WarCraft [3], StarCraft [1], Ultima Online [2], and XTux [4]. For some kinds of MOG, the game company must setup servers for the players, however, it takes a lot of cost to keep such servers running (e.g., electrical power, server maintenance). Moreover, this is not a scalable method because the game company must increase the number of servers according to the maximum number of players online at the same time, i.e., the number of players at peak time. On the other hand, a player can behaves

as a server and other players can join the game for some kinds of MOG. If we play such MOGs using peer-to-peer (P2P) network, it will be cost-free for the game company and it is a very scalable method. But the main problem is how to construct an efficient and scalable P2P network for playing MOGs because the network characteristic of each player may differ so much. Moreover, each node (especially the servers) must avoid maintaining information of all other nodes which may be in the order of thousand nodes. Furthermore, consistency which is an important issue in networked games must be considered. Namely, each player should have the same screen at the same time. This means that admission control is necessary in order to allow the players which have similar characteristic (for example, delay, bandwidth) play the games together. However, the existing P2P networks do not provide this feature. Therefore, we propose a scheme to construct and manage P2P groups for MOGs. Our scheme maps players' coordinate in the real world to the virtual plane. This coordinate can imply the network characteristics of each player. Hence, we can use such information to group the players together. The scheme composes of two main features, i.e., rough estimation and refinement procedure.

The remainder of the paper is organized as follows. Section 2 describes the motivations and the requirements of this work. Section 3 presents the problem statement. Section 4 presents the design of our proposal. Section 5 analyzes the proposed scheme. Section 6 discusses related work. Finally, we conclude the paper in Section 7.

2. P2P-Based MOG

Usually, MOGs need the server because of the following reasons.

- **Security.** First, the fairness is the most requirement issue for the players when playing the networked games. The server can prevent against many cheats. Second, the game company needs to authenticate the players in order to make a billing system.
- **Consistency.** Each node can synchronize its clock with the server. Thereby, every player can move with the same time step.
- **Reliability.** Since, all of the game data is managed by the trust server, it is difficult for the players to do cheating.

However, the server-based approach also has some disadvantages as follows.

- **Cost.** To keep the server running, it takes many costs consisting of electrical power, server maintenance, workers, etc.
- **Scalability.** The game company must install additional servers when the number of players increase.

We decide to use a P2P network because it can remove the above mentioned drawbacks occurring in traditional server-based approach. In particular, the game data is distributed and calculated at each node, therefore, we can overcome the scalability problem. The maintenance cost also decreases since the server is not necessary anymore. However, it is challenging to keep the advantages of server-based approach described above, i.e., security, consistency, and reliability. It is difficult to protect against cheating because the game data is independently run on each node. In order to simplify the problem we ignore the security problem. We will concentrate on scalability and performances.

3. Problem statement

In this paper, we propose an efficient scheme to determine an appropriate group for each MOG player. A user plays the MOG via the Internet on the same virtual space and time with other users. The key idea of our approach is to find the coordination of each player on a *Virtual Environment map* (VE map) according to the delay. Then the nearby players are grouped together. As a result, when playing the MOG the delays to other peers in the group are low although the player connect to the Internet using low-speed connection. Therefore, the players can play the MOGs smoothly without any disturbance especially when playing the first shooter game which is very sensitive to the delay.

We assume that the game company setup a server (S) to maintain a list of players including IP address of current n players ($1, \dots, i, \dots, n$), their coordinates in VE map (x_i, y_i) , and currently participated groups (g_i). Each node i determines its own threshold as a desired condition in participating a group. This threshold is the maximum delay (D_i) that this node can be tolerable or acceptable. Consequently, the node will choose a group whose delays to any nodes are less than this threshold. The delay threshold can be easily mapped to a distance threshold (R_i) in the VE map according to some predetermined ratio (e.g., delay 1ms is equal to one unit in VE map). The delay measured between any two nodes i and j (d_{ij}) is also mapped to the distance (r_{ij}) with the same ratio. The coordinate is determined by the delay between the nodes, i.e., nearer node has lower delay than farther node. Therefore, each node tries to connect with the nearer nodes in order to reduce the delay which is a main factor of networked game. After deciding a group, each node needs to know only the information within its group composed of IP address and coordinate of current group members. The problem needed to be solved is how to determine an appropriate group for each player.

4. Our Proposal

A proposed scheme composes of two stages: rough estimation and refinement. The rough estimation is used to find approximate placement of a node in VE map before entering the refinement stage where an exact position and an appropriate group are calculated. We also propose to use super nodes in order to achieve scalability and efficiency.

4.1. Rough Estimation

When a node i (player i) need to play MOG with other players, it asks current VE map from the server (S) whose IP address is known in advance. The IP address of the server is published by the game company. First, the new node measures the delay with three randomly chosen nodes appeared in the VE map in order to calculate its approximate coordinate (x_i, y_i) in the same virtual space. The calculation can be done by preliminary geometry. If there are less than three nodes in the current VE map, the node measures the delay with all possible nodes (one or two nodes). The first node in the VE map is always at coordinate $(0, 0)$. The second node can be at any point on the circumference of the circle whose center is at $(0, 0)$ and the radius is r . The delay measurement can be done by any available tools such as ping. This step is called a *rough estimation* procedure.

Algorithm 1 The grouping algorithm.

```
1: procedure GROUPING
2:   get VE map from the server
3:    $(x_i, y_i) \leftarrow (0, 0)$ 
4:   if  $n \geq 1$  then
5:     call ROUGH procedure
6:      $G_{candidate} \leftarrow$  a set of groups whose all
       members are within radius  $R_i$  centered at
        $(x_i, y_i)$ 
7:     call REFINE procedure
8:     while  $(g_i = \emptyset) \ \& \ (D_i < D_{max})$  do // (node
        $i$  does not have a group) & (the threshold
       delay is still less than some defined value)
9:       increase threshold  $R_i$  and  $D_i$ 
10:       $G_{extend} \leftarrow$  a set of groups whose all mem-
        bers are within radius  $R_i$  centered at
         $(x_i, y_i)$ 
11:       $G_{candidate} \leftarrow G_{extend} - G_{candidate}$ 
12:      call REFINE procedure
13:    end while
14:  end if
15: end procedure
```

4.2. Refinement

The new node chooses a group whose all of the nodes are within the radius R_i centered at itself (x_i, y_i) to perform a *refinement* procedure. There may be many groups that satisfy this requirements. A set of such groups is called a *candidate group* ($G_{candidate}$). Let l_{max} be the distance between the new node and the farthest node in a group. If there are many candidate groups, the new node will choose a group whose l_{max} is the minimum. Then it performs delay measurement to all nodes in the chosen group and check whether the measured delays are less than its defined threshold (D_i). If this condition is satisfied, the new node enters this group and re-calculates its location (x_i, y_i) in the VE map according to new measurements. Otherwise, the new node chooses the next candidate group and performs the refinement procedure again. The next candidate group will be a group whose l_{max} is the minimum except the groups that have been tested already. If there is no any group that satisfies its threshold, the new node will increase the threshold (D_i), i.e., enlarge the radius (R_i) in the VE map and perform the above process again.

The pseudo-codes of the above mentioned procedures are shown in Algorithm 1, 2, and 3. Algorithm 1 is an main algorithm while Algorithm 2 and 3 show the rough estimation procedure and the refinement procedure respectively.

Algorithm 2 The rough estimation algorithm.

```
1: procedure ROUGH
2:   if  $n < 3$  then
3:     measure delay with all current nodes
4:   else //  $n \geq 3$ 
5:     measure delay with 3 random nodes
6:   end if
7:   calculate rough  $(x_i, y_i)$ 
8: end procedure
```

Algorithm 3 The refinement algorithm.

```
1: procedure REFINE
2:   sort the groups in  $G_{candidate}$  according to  $l_{max}$  in
   ascending order
3:    $g_{test} \leftarrow$  first group in  $G_{candidate}$ 
4:   repeat
5:     for all nodes in  $g_{test}$  do
6:       measure delay
7:     end for
8:     if the delays to all nodes in the group  $< D_i$ 
       then
9:       calculate exact  $(x_i, y_i)$ 
10:       $g_i \leftarrow g_{test}$ 
11:      node  $i$  enters this group
12:     else
13:        $g_{test} \leftarrow$  next group in  $G_{candidate}$ 
14:     end if
15:   until (node  $i$  found an appropriate group) or (all
     groups in  $G_{candidate}$  has been used up)
16: end procedure
```

4.3. Dynamic Grouping

However, there are both static and dynamic components of delays which contribute to fluctuations of the measurement. Moreover, when the packets are sent from the same source to the same destination, they may pass through many different routes depending on congestion control policy, router behavior, etc. Based on the previous study [10, 13], the routes in the Internet often change along the time, especially at the high congested routers. Therefore, we realize that the nodes in our proposal should adapt along with current network characteristics. In particular, if the current delay is higher than the threshold (D_i), it may change to another groups by using the refinement procedures as described above. Moreover, the players can request to change group by himself if the current condition does not satisfy. In this case, it is not necessary to perform an active measurement, the nodes can use the delays measured from the packets sent and received by the game itself, i.e., passive measurement.

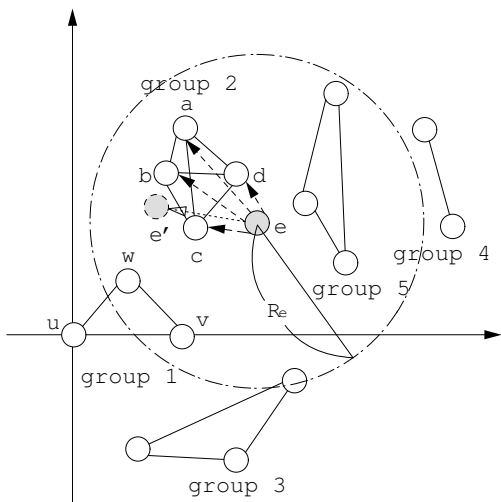


Figure 1. Refinement procedure.

4.4. One Example

An example of group selection is illustrated in Figure 1. In Figure 1, node u firstly joins the game, therefore the position of u is $(0, 0)$. The next player is v and it chooses to be at $(0, r_{uv})$. In fact, node v can be at any point on the circumference of the circle whose center is $(0, 0)$ and radius is r_{uv} . After the third node w has known r_{wu} and r_{wv} , it can calculate its approximate location. Let consider more detailed scenario where the current VE map composes of all nodes shown in the figure except node e which are going to join the game. Currently, there are already five groups in the VE map. The nodes in the same group are connected by the solid line. After node e has got a VE map from the server and calculated its rough location, it found that there are two candidate groups (group 2 and 5) within in its defined radius R_e . Node e then performs the refinement procedure with the group 2 because l_{max} of group 5 is less than that of group 5. It measures the delay with all members in group 2, i.e., node $a, b, c,$ and d . The measured delay is less than its threshold D_e , therefore node e decides to join this group and it also updates its location according to new measured delays.

4.5. Super Node

We also introduce *super node* which is a representative node of each group in order to achieve more efficient and scalable approach. Consequently, the rough estimation procedure is slightly different from the algorithm described so far. Instead of randomly choosing three nodes from all of the current nodes, the new node will randomly

chooses three nodes from a set of super nodes. A procedure used to determine the super node is as follows.

Each node measures the delay with all members in the group, it then sums all of the measured delays. A node whose this sum is minimum will be the super node. However, a node which can be the super node must join the current session more than a specified period of time (T_{join}), i.e., the node must play the game more than T_{join} minutes. Additional rule is that the first node who joined the group will be super node automatically. This selection will be done every time that a new member joins the group. As an optional, we can also force to perform this selection periodically in order to rotate the role of super node.

The responsibility of Super Node The super node has a responsibility to report the current location of all group members to the server. The super node can use passive measurement to calculate the current locations. It can report the information of many nodes within one packet. Actually, the super node can report such information if there is any changes, otherwise it does nothing. If we do not have the super node, all nodes must report its own location by itself which incurs a large amount of traffic compared to the case of using the super node as being analyzed in Section 5.

5. Performance Analysis

Since we have two versions of the proposal depending on whether we use the super node. For the sake of referring, we name *basic* and *super* scheme for a proposal with and without super node respectively. To show the efficiency and scalability of our scheme, we decided to use the number of transmissions (T_x) as a metric in the evaluation. We note that this metric counts only overhead packets.

Assume we have n nodes and such n nodes construct m groups where $m \ll n$. It is explicitly that the number of transmissions is n and m for the basic and super scheme respectively. Actually, n -order is much more than m -order. For example, StarCraft and WarCraft can support up to eight players simultaneously which means that the super scheme sends the control packets less than the basic scheme eight times at the extreme case. This highly affects the network condition especially the popular games which has many thousand users at some point of time.

6. Related Work

Andersen et al. [5] introduced a Resilient Overlay Networks (RON) which react quickly against network failure or degradation. Generally, it takes a few minutes to

find a new path by BGP-4 but RON takes only a few seconds which much less than traditional approach. Moreover, users can choice prioritize latency, loss rate or throughput.

Pantel and Wolf [9] studied an effect of delay in networked game. As reported in the paper, over 500ms delay is a cause that the action and reaction do not fit together. The players can control and it is possible to adapt its own style and situation in 200ms delay network but the overall behavior is not realistic. 100 ms delay is acceptable if there is no high demands. It is hard to notice 50ms delay. They concluded that the delay highly affects the networked game.

Jehaes et al. [8] proposed a QoS scheme to manage the delay in networked game by discriminating the game packets from other packets (e.g. web traffic). High priority packets will be sent quickly.

Hu and Liao [7] proposed a fully-distributed peer-to-peer architecture to solve the scalability problem of Networked Virtual Environment. To solve such problem, they explain how a node in P2P network find neighboring nodes by using voronoi diagram in virtual environment. In contrast, we proposed to construct the topology according to network characteristic.

CAN [11], Chord [12], Pastry [6], and Tapestry [14] use Distributed Hash Table to find resource in P2P networks quickly and efficiently. These schemes are also scalable and robust to node participation and secession.

7. Conclusion

In this paper, we have proposed a measurement-based topology construction using VE map for P2P networked game. This method decides an appropriate group according to desired thresholds determined by each player which may highly differ. The key idea is to measure each node's delay in order to decide a coordination and choose a group that meets the requirements. This is done in two stages, *i.e.*, rough estimation and refinement procedure. Our future work is to implement it as a middleware and evaluate our scheme.

References

- [1] StarCraft. <http://www.blizzard.com/starcraft/>.
- [2] Ultima online. <http://www.uo.com/>.
- [3] World of Warcraft. <http://www.worldofwarcraft.com/>.
- [4] XTux. <http://xtux.sourceforge.net/>.
- [5] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Banff, Canada, Oct. 2001.
- [6] P. Druschel and A. Rowstron. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.
- [7] S.-Y. Hu and G.-M. Liao. Scalable peer-to-peer networked virtual environment. In *Proc. of ACM SIGCOMM 2004 Workshops on NetGames'04*, pages 129–133, Portland, Oregon, Aug. 2004.
- [8] T. Jehaes, D. D. Vleeschauwer, T. Coppens, B. V. Doorse-laer, E. Deckers, W. Naudts, K. Spruyt, and R. Smets. Access network delay in networked games. In *Proc. of the 2nd Workshop on Network and System Support for Games*, pages 63–71, Redwood City, CA, May 2003.
- [9] L. Pantel and L. C. Wolf. On the impact of delay on real-time multiplayer games. In *Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 23–29, Miami, Florida, May 2002.
- [10] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, Department of Electrical Engineering and Computer Science, April 1997.
- [11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 161–172, San Diego, CA, Aug. 2001.
- [12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, San Diego, CA, Aug. 2001.
- [13] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proc. of the Internet Measurement Workshop (IMW)*, San Francisco, CA, November 2001.
- [14] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, EECS Department.