Slide: A Model to Seamlessly Switch Zones for MMOG

Jingbo Shen
Computer Science Department
University of Science and Technology of China
Hefei, China, 230027
zxfsjb@mail.ustc.edu.cn

Xufa Wang, Shanshan Liu, Jinlong Li
Computer Science Department
University of Science and Technology of China
Hefei, China, 230027
[xfwang|jlli]@ustc.edu.cn, ssliu@mail.ustc.edu.cn

ABSTRACT
Recently, P2P networks have been used to support massively multiplayer online games (MMOG). Because players in MMOG have localized interests, the whole game space is divided into multiple sub-spaces to reduce network traffic and increase scalability. However, when players switch sub-spaces or sub-spaces are partitioned or merged frequently, system jitter will occur. The 'Slide' model that we propose includes: (1) an advance resource discovery mechanism, which uses common peers to help each other to discover resources, in order to balance network traffic and reduce the dependence on super peers; (2) an event delivery mechanism delivering action messages instead of state messages, to reduce network traffic of the system; (3) a buffer technique to seamlessly switch zones, which sets buffers between zones to avoid players frequently switching zones. We compared the Slide model with the SimMud model, and the results show that the Slide model can reduce network traffic to 25.94% compared to that of the SimMud model, and thus avoid system jitter.

Categories and Subject Descriptors
H.3.4 [**Systems and Software**]: Distributed systems, Performance evaluation (efficiency and effectiveness)

General Terms
Design, Experimentation

Keywords
Zone Model, System Jitter, Resource Discovery, Event Delivery

1. INTRODUCTION
In recent years, Massively Multiplayer Online Games (MMOG) have become popular. In order to reduce bandwidth consumption and the burden of server and increase scalability of MMOG, the scalability of peer-to-peer (P2P) overlay has been taken into account [1]. Because of the lack of support from central server in MMOG based on P2P, we need to conquer problems of resource discovery and event delivery. Besides, some MMOGs have latency requirements below 500ms [2], such as EverQuest, Star Wars Galaxies and so on. Most peers in MMOG have localized interests. It means they only need part of information in the whole game world. To reduce the traffic and latency, an efficient method is to divide whole game world into some zones [3], and design game based on zones. But in those zone models, when players frequently do some special actions, such as switching zones, the network traffic and latency will suddenly increase for a while, which we called system jitter. System jitter will affect the availability and scalability of MMOG.

In this paper, we analyze the cause of system jitter for MMOG based on zones, propose a zone model called 'Slide' to prevent system jitter, and achieve seamless switching of zones. This paper is organized as follows: After discussing the related work in Section 2, we present our novel model in Section 3. A comparison of our model with other model is reported in Section 4. Finally we conclude the paper in Section 5.

2. ZONE MODEL FOR MMOG
In the literature, some studies have been performed on zone models for MMOG. Most of them have the problem of system jitter.

Limura *et al.* proposed ZF model [4] using Distributed Hash Table (DHT) for resource discovery. There are two kinds of super peers in each zone, one is zone owner and the other is data holder. Data holder is responsible for resource discovery by storing all resources in its zone. Zone owner keeps connections with all peers in its zone, and is responsible for computing and delivering the global states. When a player joins or leaves the game or switches zones, it may cause the changes of super peers, which will bring a lot of traffic, and may cause system jitter when it happens frequently.

Gauthier Dickey *et al.* proposed N-Tree protocol [1] which divides the game world into sub-spaces for resource discovery, and allows peers to deliver updates by themselves. The model dynamically partitions and merges zones based on the number of peers in each zone. So when a player joins or leaves the game or switches zones, zones may be partitioned or merged, which cause the N-tree overlay to be rebuilt and the resources in each zone to be checked. It will cause a lot of traffic and a high delay, and may cause system jitter when it happens frequently.

Knutsson *et al.* proposed SimMud model [5] which divides the game world into fixed zones, and uses Pastry protocol [6] for resource discovery and Scribe protocol [7][8] for event delivery. SimMud limits each player to listen in one zone at a time to keep its prototype simple. But in real games there are some zones

which are too big for single coordinators to be responsible for updating the states of objects. So they are needed to be divided into several small zones, it means that peers have to join several multicast trees to communicate with players in other zones. When players move through some special positions, peers need to join new multicast trees to get resources, and may cause system jitter when it happens frequently.

Y.He *et al.* [9] thought it is difficult to merge parallel sub-spaces caused by network outages in SimMud. It will cause a lot of traffic and conflict adjustments. It is also a cause of system jitter. They proposed a fully distributed algorithm based on state-stack matching to resolve the merging issue, but they did not solve system jitters which are caused by other reasons such as peers change multicast trees.

Above-mentioned analysis indicated that system jitter related to the existent of sudden increase of resource discovery, the method of event delivering which excessively relies on super peers, the method which players choose when they switch zones frequently. Based on these three causes for system jitter, we propose a model which uses progressive resource discovery, directly delivers action messages to reduce the dependence on super peers, and sets buffers on the boundaries of zones to reduce and balance network traffic of the whole system. The model can avoid system jitter, and achieve seamless switching zones for MMOG.

3. SEAMLESS SWITCHING ZONES FOR ZONE MODEL

To avoid system jitter caused by changing of super peers, partitioning and merging of zones, switching zones and so on, we propose a novel zone model called 'Slide' to achieve the goal of seamless switching zones. The base structure of Slide is presented first.

3.1 Base Structure of Slide

Some definitions are given:

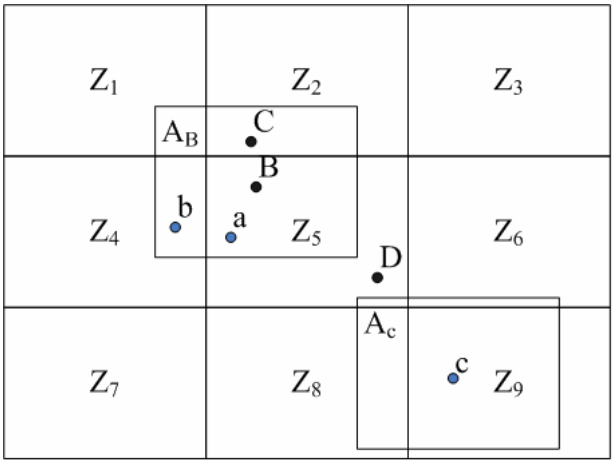

Figure 1: a Small Part of Virtual Environment of a Game

Zone: a sub-space of the game in geometry, including all resources in the sub-space, such as players and objects. Objects include foods, weapons, non-player-characters (NPC) and so on. As shown in Figure 1, we use rectangles to represent zones. A zone does not overlap with others in Slide. All zones compose the whole game world.

The sizes of zones should be set first. If the size of a zone is too large, the number of peers in a zone will be too large, which is likely to increase the computation and communication of a peer. If the size of a zone is too small, the peer will have to switch zones frequently, which increases unnecessary communications. The size of a zone should be set according the requirement of a real game. Slide lets the size of a zone equal to vision coverage area.

Vision Coverage Area: the set of resources which a player can see or perceive.

The vision coverage area of a player is related to player's perceptive capability, not restricted by the area of a zone. A player's vision coverage area can relate to four zones at most. In Figure 1, A_B is Player P's vision coverage area when he stands at Position B.

The vision coverage area of an object is the area in which a player can see the object. In Figure 1, A_c is an object's vision coverage area when it stands at Position c.

Computation Area: the set of resources which a player has to compute and storage.

A resource's computation area is the intersection of resource's vision coverage area and the zone which the resource belongs to, to prevent from cheating. And the state of a resource is not computed by a single peer, but by all peers in vision coverage area of the resource and zone which the resource belongs to. In Figure 1, Player P's computation area is area $A_B \cup Z_5$ and the computation area of the object in Position c is $A_c \cup Z_9$.

Super Peer: the peer taking charge of a zone. Super peer has the same storage and computation area as a normal peer standing at the same position. It just saves and computes states of all players and objects in its vision coverage area and zone which it belongs to.

As a special peer in Slide, super peer is mainly used to help other peers discover resource and deliver updates. When a peer enters a new zone, it has to get addresses of all peers and states of all objects in the zone from the super peer of the zone; when a peer can not get the resources which it needs from other peers in its computation area, it has to ask the super peer of the related zone for resource discovery.

To simplify problem, super peers are random selected in Slide. If a super peer left or failed, it is just needed to select a new super peer. Because super peer has the same storage as a normal peer, no date is needed to be transferred from one super peer to its successor.

3.2 Avoiding System Jitter in Slide

Slide avoids possible system jitter in three ways: resource discovery, event delivery and seamless switching zones, and can implement a seamless zone model for MMOG.

3.2.1 Resource Discovery in Slide

In Slide, each peer holds three lists: an addresses list of relevant peers, an addresses list of relevant super peers, and a states list of resources. The addresses list of relevant peers contains addresses of all peers in computation area. The addresses list of relevant

super peers contains addresses of super peer of the zone and super peers of eight zones around the zone which the player belongs to. The states list of resources contains states of all resources in computation area.

When the computation area changes, there will be some new resources present to computation area. The player needs to confirm those new resources and add them to the states list of resources, which called resource discovery.

Every peer holds the states of all players and objects in its computation area, and only when a player just joins the game or moves it need to discover resource. When a player just joins a game, the peer holds no information of any players or objects. When a player moves, vision coverage area changes along with the player moving. Each step, the peer must discover resource to confirm whether there are some new players or objects present to its vision coverage area. Just like in Figure 2, if Player P made a move from Position B to Position C, vision coverage area moves one piece left. Player P's vision coverage area is A_C now, and it has to find whether there are some resources in area M and N. This process is coherent.

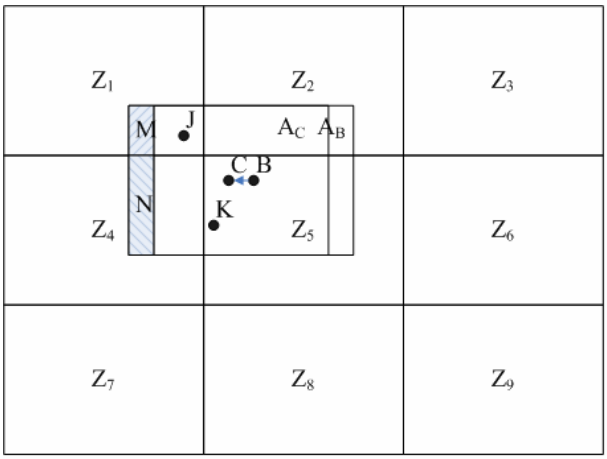

Figure 2: the Area to be Discovered Resource in Slide

1. Player P_i moves one step
2. P_i sends update to all relevant players
3. P_i computes the area E_i for resource discovery
4. P_i searches for a Player P_k in relevant players whose vision coverage area contains E_i
5. If there is a P_k existed

   P_i asks P_k for resource discovery in E_i

   else

   P_i asks the super peer of E_i for resource discovery
6. P_i gets the states of objects and addresses of players P_j in E_i
7. P_i sends P_j the state of itself and asks for the state of P_j
8. P_i gets the state of P_j

Figure 3: the Flow Chart of Resource Discovery in Slide

A progressive method is used to discover resource while moving, and the flow is presented in Figure 3. This progressive resource discovery method divides massive resource discovery into several small quantities of resource discovery to proportion the suddenly increase traffic. As cost, the times of resource discovery increases, and because each communication of resource discovery contains ask messages and control messages, the total network traffic will increase. But these ask messages and control messages are small, and are distributed to each time of resource discovery, will not cause an obvious effect to peer.

In order to reduce the burden of super peers, Slide mostly uses normal peers to help for resource discovery. When a Peer P_i wants to discover resource in a new Area E_i, the peer has to first check its addresses list of relevant peers in order to find a Peer P_k whose vision coverage Area E_k contents the follow expression:

$$E_i \subseteq E_k$$

If there is a Peer P_k existed, P_i can ask P_k for resource discovery. Only when there is no P_k existed, P_i has to ask super peer for help.

In Figure 2, when Player P at Position C needs to discover resource in area M and N, the peer has to first check its addresses list of relevant peers, and will find that Player P_J and Player P_K can help for resource discovery. The peer then sends messages to P_J and P_K in order to get resources in area M and N. If there is no Players P_J and P_K existed, the peer has to ask super peers of Zone Z_1 and Z_4 instead.

In order to assure the rate of resource discovery, ask message will not be transmitted. The method discovers resource in O(1) hops, and can reduce the delay of resource discovery efficiently.

Players can compute and deliver their states themselves directly. And when Player P_A saw Player P_B, not only Player P_A needs to get the state of Player P_B, but also Player P_B needs to get the state of Player P_A. So when resource discovery result is a player, answer message only contains address of the peer. Then Peer P_A will connect with Peer P_B, get the state of P_B and send it self's state to P_B. It will reduce the traffic of peers which help for resource discovery, because message of address is much smaller than message of state.

When resource discovery result is an object, the object dose not have the capability of compute and communicate, so the peer which helps for resource discovery has to send answer message which contains state of the object.

The method of resource discovery distributes traffic to the process of players' moving, to avoid sudden increase of traffic caused by massive resource discovery. It basically uses normal peers for resource discovery, which efficiently reduces the burden of super peers and the possibility of system jitter.

3.2.2 Event Delivery in Slide

The changed state of resource should be sent to all peers in vision coverage area, and maintain the latest edition. This process is called event delivery.

To improve the security, resources in Slide are multiple controlled. It means that all changes of a resource are computed by all peers in computation area of the resource. So it is unnecessary to send message of state. Instead, message of action is sent in order to notify of the change, and the traffic will be efficiently reduced, because the size of action message is much smaller than that of state message.

The change of a player's state is sent by the player self using a message of action, and then peers in the player's computation area compute new state of the player.

Objects have no capability of computing and can not deliver updates by themselves, so some methods are needed to deliver updates of objects. The spontaneous changes of an object, such as change of object's color, are caused by random number of the object. And all peers in object's computation area have known the random number and are computing the object, so they can figure out the changes. So the spontaneous changes of an object do not need to be sent. The passive changes of an object are changes such as the reduction of a monster's life caused by a hit from a player in the game. The passive changes of an object can not be figured out from random number, so event delivery is needed. Instead of object, the player who made an action to object figures out the passive change of object, and sends the action message, such as action message of "be hit by a shoot which execution is 50", to all peers in object's computation area.

The method of event delivery that peers send messages of action directly can reduce the traffic of system and the dependence of super peers, will avoid multiple hops and reduce the latency of the network.

3.2.3 Seamless Switching Zones in Slide

In Slide, when players move inside of a zone, resource discovery is needed each step to avoid sudden massive traffic and system jitter. But when a player moves across the boundary of a zone, the resource in new zone is needed, because that player's computation area is intersection of the zone which the player belongs to and player's vision coverage area. The player needs to get the resources in new zone which are not yet in vision coverage area, and it may cause a sudden massive traffic.

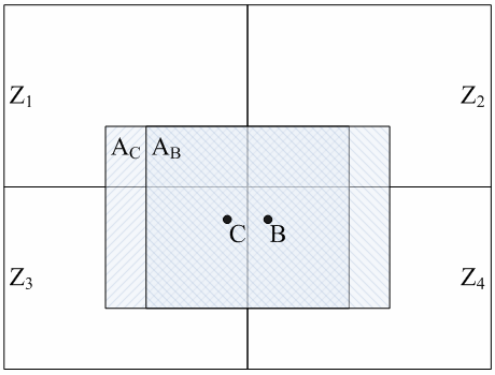

Figure 4: the Change of Computation Area

In Figure 4, when Player P moves from Position B to Position C, the player needs to get all resource in area $Z_3 \bigcap \overline{A}_B$. The traffic is much larger than the traffic caused by player moving inside a zone. If a player moves around the boundary of a zone, it may cause system jitter.

In order to solve the system jitter caused by switching zones, Slide uses buffers to cushion the traffic. Beside the boundary of a zone, a piece of area is reserved for buffer. The size of buffer should be adjusted based on demand. If the buffer is too large, computation area is large, which will increase the burden of peer. If the buffer is too small, players move across the buffer frequently, and seamless switching zones will not be achieved. In Slide, the size of buffer beside the boundary of a zone is set to be 1/8 of a zone, but it could be adjusted according the requirement of a real game. If the size of a zone is too large, the size of buffer will be reduced; and if the speed of player moving is too fast, the size will be increased.

There is a Player P moving from Z_4 to Z_3. When he just moved across the boundary, he will not change the zone which he belongs to, and still gets the resources in new vision coverage area each step by progressive resource discovery method. Only after he moved across area M, does he change the zone which he belongs to. Then he belongs to Zone Z_3, gets and computes all resources in Zone Z_3.

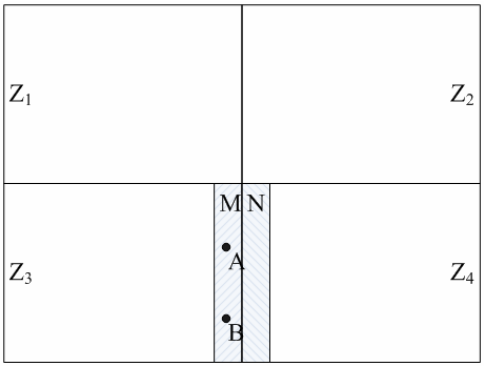

Figure 5: the Buffer between Zone Z_3 and Z_4

Though the buffer helps for seamless switching zones, it brings some of confusions to the system. In Figure 5, the Player P_A and P_B are all standing in Zone Z_3. But Player P_A is a peer moving from Z_4 to Z_3, who still stands in the buffer, and still belongs to Zone Z_4. However, Player P_B belongs to Zone Z_3. It means that different peers standing in the same place may belong to different zones, so we can not use the position of a peer to estimate the zone which it belongs to. An identifier is needed for a player to indicate the zone which it belongs to.

The method of buffer can prevent system jitter caused by frequently switching zones in a short time. And before player changes zones, vision coverage area has already covered more than half of the new zone, so the resources which need to be discovered are much smaller than the resources which need to be discovered, while a player just moves across the boundary and changes zones immediately. The traffic is reduced too. This method can reduce the burden of switching zones, and help for seamless switching zones.

4. PERFORMANCE ANALYSIS

We compare the performance of Slide model with SimMud model, and analyze the latency of network and traffic from theory and experiments.

4.1 Performance Analysis

The analysis is carried on two conditions. One is player's common actions, and the other is player's frequently moving across the boundary.

There are some suppositions which are similar to the simulation of SimMud:

1) The game world is divided to rectangles the size of which is 40*30 unit^2. A player can move 1 unit each step.

2) There are 10 players and 10 objects evenly distributed in each zone.

3) A player moves once per 500ms, and the direction will be "up", "down", "left" or "right".

4) A player battles or takes food every 20 seconds, which will not change its position but change its state.

5) There is a player joining and a player leaving per second for each zone.

6) A player delivers its update through multicast trees every 150ms in SimMud, and the update will achieve in 2 hops.

Basic symbols: the latency of network is T; the network traffic is M; the time to transmit a message once is t, it means the time of a hop is t; the state message of a player is M_S; the address message of a peer is M_A; the action message of a peer is M_M; the route message of Pastry is M_R; the number of peers in the game world is N.

4.1.1 Common Actions of Players

In SimMud model, action messages of moving, taking food and battling need not to be sent. Peers can get changes of other player's states through periodical event delivery. So in SimMud, there are only messages which related to a peer joining, leaving and multicasting to be sent per second. At most time, vision coverage area relates to four zones, so a peer has to join four multicast trees at one time.

In Slide model, the messages which need to be sent per second contain action messages of a player moving, battling, taking food and system messages of a peer joining or leaving. When a player stands at the position of boundary point of four zones in Figure 6, it's the worst condition in Slide. In this condition, only a quarter of player's vision coverage area is still in the zone which the player belongs to, and player's computation area is 7/4 zones.

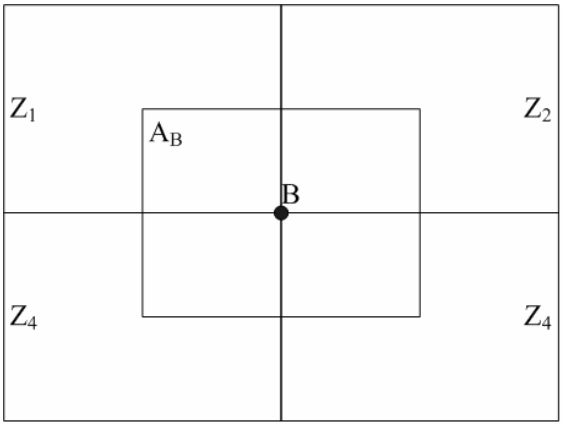

Figure 6: the Worst Condition in Slide

We analyze the traffic and delay of the two models under each player's common actions in the worst condition, and compare the total traffic of a zone with the same size per second in Table 1.

Table 1. The performance contrast of SimMud model and Slide model under player's common actions

	SimMud		Slide	
	traffic	**delay**	**traffic**	**delay**
Join	$480M_R$	2.5t	$26.5M_A+53.5M_S$	8t
Leave	$16M_R$	t	$17.5M_M$	t
Multicast	$26.67M_S$	3t	——	——
Battle or take food	——	——	$17.5M_M$	t
Move	——	——	$17.5M_M+M_S+0.33M_A$	5t
Total traffic	$496M_R+266.7M_S$	——	$27.16M_A+55.5M_S+53.375M_M$	——

Table 1 shows that the total traffic of a zone per second in Slide model is about 20.81% of the traffic in SimMud, because state message M_S is much larger than address message M_A and action message M_M. And the traffic in SimMud is related to routing of Pastry which is related to the number of peers in the whole game world. Along with increase of the number of peers, the traffic in SimMud increases too, which will affect scalability of the whole game.

In the condition of peer joining, the traffic in Slide is smaller than in SimMud, but the latency is higher. However, when a player just enters a game, the game has not gotten under way, so waiting for a short time is acceptable, and will not affect the other peers in the game. Here we use the supposition of Pastry in [5]: when there are 1000 peers, the average delay of a peer joining is 2.5 hops, and average traffic is 120 route messages.

In the condition of peer leaving, the traffic and delay in Slide is the same as in SimMud.

In the condition of player's other common actions in the game, traffic in Slide is smaller, but latency is two hops higher when players move. But it is the latency of resource discovery, not the delay of event delivery. In the process of resource discovery, other actions do not need to be blocked, so it will not affect consistency of the game. Besides, the area which needs to be discovered is at the border of player's vision coverage area, it means the border of screen might display slower than other place of screen. So we can make area to be displayed on the screen a few smaller than vision coverage area in order to solve the problem. After that, the display is seamless and will not affect current actions of players.

From above analysis we can see that Slide model can effectively reduce the traffic of the whole system under player's common actions.

4.1.2 Player's Frequent Moving across the Boundary

In SimMud model, if a player moves around the boundary of two zones, the most serious system jitter will happen when each step of player's moving caused the changing of multicast trees, and each step the player left two multicast trees and joined two new multicast trees. From the above supposes, a player moves twice each second, and will cause two times of changing multicast trees at most. Assume system jitter will happen when a player continuously has changed multicast trees ten times.

Assume that there are 5 units of buffer at each side of a zone in Slide model, so there are 10 units of buffer between two zones, and system jitter will happen when a player continuously has moved across the buffer ten times between two zones.

We analyze and compare the probability and traffic of the two models when system jitter happens in Table 2.

Table 2. The performance contrast of SimMud model and Slide model under player's frequently moving across the boundary

	Probability	Total network traffic	Average network traffic per second
SimMud	$1.0808*10^{-6}$	$9600M_R$	$1920M_R$
Slide	$1.1333*10^{-30}$	$265M_A+535M_S$	$5.3M_A+10.7M_S$

Table 2 shows that the probability of system jitter in Slide model is much smaller than in SimMud model, which can be thought as an impossible event from probability. Even if it did happen, the average traffic per second when system jitter happened in Slide is much smaller than in SimMud. It means that Slide model can achieve the aim of seamless switching zones.

4.2 Experiments and Analysis

We implement a prototype system based on the proposed method. To simplify the experiment, we have modified some suppositions which were used in performance analysis. We take every 20s as a time slice. All basic actions will be taken at least once in every 20 seconds.

4.2.1 Resource discovery and Event delivery

Our first experiment is about resource discovery and event delivery in Slide model and SimMud model. In our first experiment, we measure 10000 seconds of 1000 simulated game peers. We random select a peer as the observed peer, and take every 500ms as a sampling period. In every sampling period for the observed peer, the number of resource which needed to be discovered, the number of relevant peers and the total network traffic will be recorded, the average and variance of all these results will be computed in Table 3.

Table 3. Statistical result of the observed peer

	The number of resources to be discovered		The number of relevant peers		The total traffic(Kb/500ms)	
	Slide	SimMud	Slide	SimMud	Slide	SimMud
ave	0.948	0.868	15.08	38.954	1.987	7.659
var	2.544	5.795	4.254	6.799	0.616	1.414

Table 3 shows: the average number of resources which needed to be discovered in Slide per sampling period is similar with in SimMud, but the variance is 43.90% of in SimMud. It proves that the jittery from resource discovery in Slide is much smaller than in SimMud. It is because progressive resource discovery is taken in Slide. The load of resource discovery is distributed into each step, so when players changed zones, the resources which needed to be discovery at one time are reduced. But in SimMud, though players do not need to discover resources at most time, when players changed multicast trees, all resources in new zones will be discovered at one time.

The number of relevant peers in Slide is 38.71% of in SimMud, and the variance is 62.57% of in SimMud. It is because: Relevant peers in Slide are the peers in computation area, and the number is the number of resources in 1~7/4 zones. But in SimMud, relevant peers are the peers in relevant multicast trees which the player joined, and the number is the number of resources in 4 zones.

In Slide, the total network traffic of observed peer per sampling period contains traffic of resource discovery, traffic of updates delivered and received. In SimMud, the total traffic contains traffic of resource discovery, traffic of deliver update to coordinator, and traffic of receiving updates from multicast trees. Table 3 shows the total traffic of observed peer in Slide is 25.94% of in SimMud, which is coincident with the result of performance analyze. And the variance is 43.56% of in SimMud.

4.2.2 Buffers

The second experiment is about the effect of buffers. We implement our experiment on 1000 peers and take every 20s as a sampling period.

First, we implement our experiment without buffers around the boundaries of zones. A peer is made to move across the boundary repeatedly, called test peer. Each step, test peer will move from one zone to another. And we random select a normal peer for contrast.

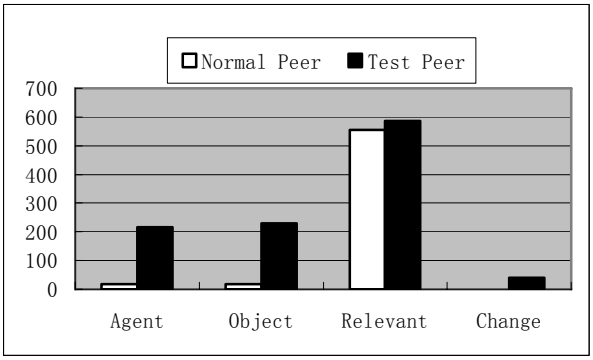

Figure 7: the Contrast of Normal Peer and Test Peer in the Condition without Buffers

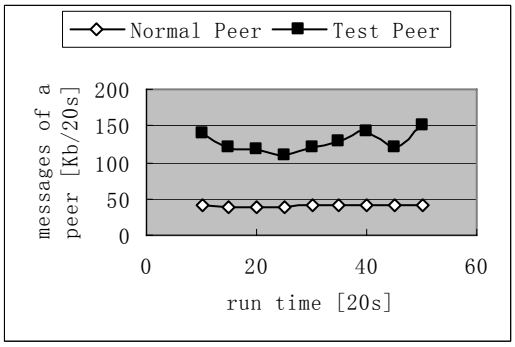

Figure 8: the Contrast of Normal Peer and Test Peer in the Condition without Buffers

Figure 7 and Figure 8 show: test peer's time of change zones is much large than normal peer's time, so the number of resource for

discovery is larger too. Though the number of relevant peers is similar, the test peer's average total traffic is about three times of normal peer's traffic.

Then we implement our experiment with buffers around the boundaries of zones. A peer called cross zone peer is just like test peer above. A peer called cross buffer peer is the peer move from one side of the buffer between two zones to the other side of the buffer repeatedly. A normal peer is random selected for contrast. Supposed the size of buffer in each side of a zone is 5 unites, so the cross buffer peer will change zone every 11 steps.

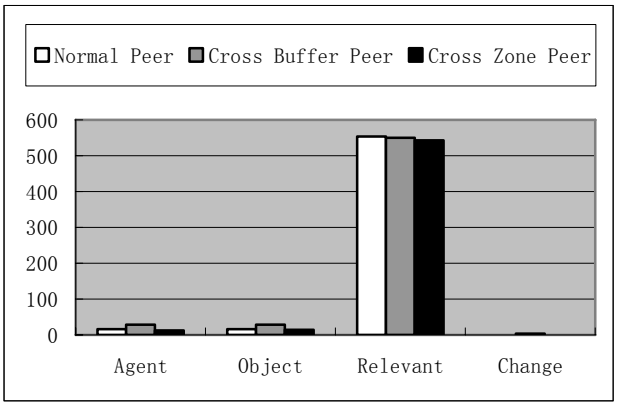

Figure 9: the Contrast of Normal Peer, Cross Buffer Peer and Cross Zone Peer in the Condition with Buffers

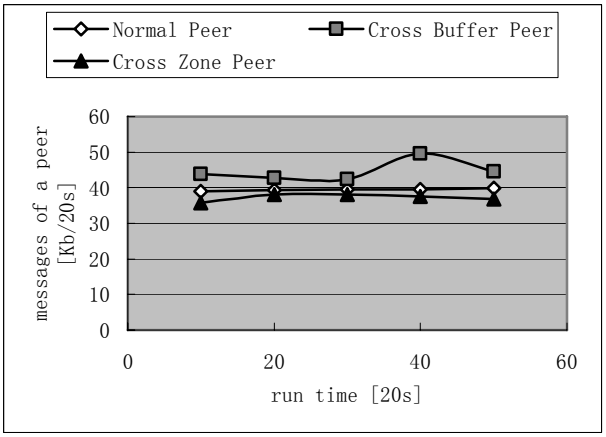

Figure 10: the Contrast of Normal Peer, Cross Buffer Peer and Cross Zone Peer in the Condition with Buffers

Figure 9 and Figure 10 show that the number of resources for discovery, the number of relevant peers and the total traffic of these three kinds of peers are similar. Cross buffer peer has the largest time of change zones, so its traffic per 20s is the largest. Cross zone peer has the smallest time of change zones, so its traffic per 20s is the smallest. But the difference of traffic between these three kinds of peers is small, about 10% of the normal peer's traffic.

The performances of normal peer in these two kinds of experiments are similar, so we can take it as the datum mark. We contrast the ratio of Test Peer to Normal Peer in Figure 7 and the ratio of Cross Buffer Peer to Normal Peer in Figure 9 about the number of players and objects for discovery, the number of relevant peers, the time of change zones and the average messages per 20s in Table 4. Result shows system jittery by changing zones can be avoided by setting buffers around the boundaries of zones.

Table 4. The contrast of Test Peer / Normal Peer in Figure 7 and of Cross Buffer Peer / Normal Peer in Figure 9

	TestPeer / *NormalPeer* **in Figure 7**	*CrossBufferPeer* / *NormalPeer* **in Figure 9**
Agent	12.50	1.78
Object	13.13	1.75
Relevant Peers	1.05	0.99
Change Zones	37.96	4.73
Average Messages	3.21	1.13

From these experiments above we find that the progressive resource discovery method and event delivery method in Slide can reduce the network traffic, so the possibility of system jittery will be reduced too; and the buffers in Slide can avoid system jittery which happened under some special condition such as frequently changing zones.

5. Conclusion

We have proposed a game model based on P2P called Slide, which can achieve seamless switching zones. The model uses normal peers to help each other for resource discovery. When there is no normal peer to give a help, peers will discover resource under the aid of super peers. With the change of vision coverage area, progressive resource discovery is distributed to each step, in order to avoid sudden massive resource discovery. Players send action messages as updates to relevant peers directly, to reduce the dependence of super peers. The new states of resources are computed by all peers in computation area, so the chance of cheat is reduced. Action messages are sent when states of resources changed, to avoid sending unnecessary messages and reduce the traffic. The buffers are used to reduce frequent traffic when players move across the boundaries of zones, to switch zones seamlessly. The size of buffer should be adjusted to different demands. The state of a resource is computed by several peers at one time, so the results from different peers may be different for lost of package, wrong event orders and so on. It is likely to cause the issue of inconsistency, and we will study this issue in future.

6. ACKNOWLEDGMENTS

Our thanks to Jin Zhang and Ping Zhu for their useful comments and valuable feedbacks, and give special thanks to Jin Zhang for proofreading.

7. REFERENCES

[1] C.GauthierDickey，D. Zappala, et al. (2004). "A Distributed Architecture for Massively Multiplayer Online Games", Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, ACM.

[2] Chris GauthierDickey，Virginia Lo，Daniel Zappala （2005）. "Using N-Trees for Scalable Event Ordering in Peer-to-Peer Games", Proceedings of the international

workshop on Network and operating systems support for digital audio and video: 87 - 92.

[3] Anthony, Y. and T. V. Son. “MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games”. Stevenson, Washington, USA, ACM Press.2005

[4] Iimura. T., H. Hazeyama, et al. (2004). "A Peer-to-peer Approach to Scalable Multiplayer Online Games." Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games Game infrastructure: 116 - 120.

[5] Knutsson, B., H. Lu, et al. (2004). "Peer-to-peer support for massively multiplayer games"，INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies 1: 107

[6] Antony, I. T. R. and D. Peter (2001). “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems”, Springer-Verlag

[7] Castro, M., M. B. Jones, et al. (2003). "An evaluation of scalable application-level multicast built using peer-to-peer overlays"，INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE 2: 1510- 1520 vol.2

[8] Castro, M., P. Druschel, et al. (2002). "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure." IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS VOL. 20, NO. 8: 1489-1499.

[9] He, Y., Y. Zhang, et al. (2005). "On Mitigating Network Partitioning in Peer-to-Peer Massively Multiplayer Games", Networking and Mobile Computing Volume 3619/2005: 481-490.

[10] Morgan, G. and K. Storey. “Scalable collision detection for massively multiplayer online games”. Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on 2005.

[11] Stefano, F. and R. Marco, “Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games”. Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. 2005, Valencia, Spain: ACM Press 405-412.

[12] Kutten, S. and D. Peleg, “Asynchronous resource discovery in peer-to-peer networks”. The 21st IEEE Symposium on Reliable Distributed Systems, 2002: p. 224- 231.