OPT-UFR algorithm for distributed environment

Huijuan Wang, Quanbo Yuan*

North China Institute of Aerospace Engineering, Langfang, Hebei, China

Received 1 October 2014, www.cmnt.lv

Abstract

As a novel relevance filtering method, some experiments are done to illustrate the performance of UFR since this algorithm is proposed. However, there is no any comparison between UFR and some other relevance filtering mechanisms. This paper compares UFR with VON in scalability and efficiency. By changing the peer number and changing the AOI when setting the peer number fixed in experiments, it is proved that UFR is more efficient than VON in these two cases. Then some experiments on group based moving model prove the result more sensitive. To counter that the original "strip" algorithm to calculate the UFR border is not very efficient, we proposed OPT-UFR to reduce the useless update messages, and finally proposed a new algorithm to solve the heavy traffic problem for the joining node. Experimental results show that OPT-UFR always has better performance than both random based and group based moving models.

Keywords: filtering method, UFR, VON, OPT-UFR

1 Introduction

UFR is a novel relevance filtering method proposed by Makbily in 1999. The concept of UFR is very different from other relevance filtering technologies for distributed environment. Most of the existing relevance filtering technologies tries to maintain a neighbour list by using neighbour discovery mechanisms to keep connected and then implement the relevance filtering according to the neighbour list. Otherwise, Makbily maintains an update free region which is called UFR for each pair of peers. It means that if two peers are both in their UFRs referred to each other, they don't need send state update messages to each other. Because both of the peers will not enter each other's AOI if they don't go out of UFR. This can be ensured by the algorithms to calculate the UFR borders.

One recent application by using UFR is to detect the efficient proximity among mobile friends proposed by Arnon Amir. They use "strips algorithm" which is very similar to Makbily's original method to calculate the UFR borders. They also give out some mathematical analysis to show the efficiency of UFR algorithm. However, the condition of the analysis is much too idealized, the practical environment is not fully considered. Except for Amir's work, Makbily and Steed also have some experiments to illustrate the performance of UFR algorithm. But they all don't compare UFR with some other relevance filtering mechanisms.

The contribution of our work includes the following aspects. First, we compare UFR with VON to give an intuitive view of the scalability and efficiency of UFR. Second, based on the experiments, a regression analysis of UFR will be given. Third, we propose a kind of an optimization for "strip algorithms" to reduce the useless update messages. At last, we discuss the peer joining procedure and propose a "delayed joining" algorithm to solve the heavy traffic problem for the joining node.

2 Related works

2.1 STRIPS ALGORITHMS

As mentioned above, strips algorithms is used in the wireless mobile environment to reduce the location update messages between peers. Each peer in the world has its area of interest. The area of interest is assumed to be a circle with identical radius R for all the peers. Let a, b be two users whose Euclidean distance, denoted $|b-a|$, is larger than R. Let $\ell(a,b)$ denote the bisector of the line connecting a and b. (see Figure 1). Let S(a,b) denote the infinite strip of width R whose central axis is $\ell(a,b)$. Let ei denote the line bounding S(a,b) on the side closer to a. The idea behind this method is that as long as neither a or b enters S(a,bi), they do not need to exchange location update messages. The strip serves as a static buffer region between a and b and ensures that as long as they are on both sides neither one of them is in the vicinity of the other.

They also discuss the efficiency of strips algorithm by mathematical analysis. They point out the ε plays an important role on determining a trade-off between the desired distance and accuracy in generating alerts and the required number of location update messages. But actually in the practical environment, the one step distance of each peer often is not very small, so the limit case will not occur.

* ***Corresponding author's*** e-mail: 545682864@qq.com

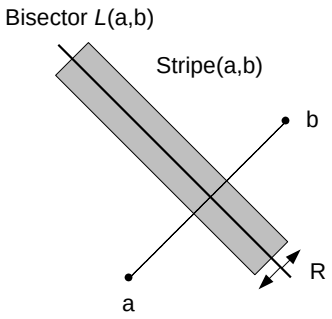

FIGURE 1 Concept of stripes algorithms

2.2 VON

VON is a recent proposed relevance filtering method for fully distributed environment. The Voronoi diagram [13] from computational geometry is used to maintain and discover the neighbours. VON has good scalability and consistency characteristics.

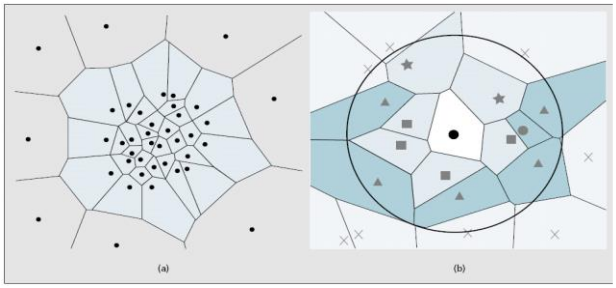

FIGURE 2 VON description

Each node in VON is represented as a site in the Voronoi diagram. For a given node, they define AOI neighbours as the nodes whose positions are within its AOI. Enclosing neighbours are nodes whose regions immediately surround the given node, and boundary neighbours are AOI neighbours whose enclosing neighbours may partially lie outside the AOI [1] (Figure 2). Each node maintains a Voronoi diagram of all AOI neighbours and directly connects them to minimize latency. As only a few neighbours are kept, the cost to maintain a Voronoi diagram at each node is low. To prevent overlay partition (i.e., groups of nodes become mutually unaware of each other), they also require each node to minimally keep its enclosing neighbours (which may be outside the AOI when neighbouring nodes are sparse).

When a node moves, position updates are sent to all connected neighbours (i.e., AOI neighbours plus any enclosing neighbours beyond AOI). Neighbour discovery is done via notifications from boundary neighbours, as they know both the moving node and other nodes beyond the AOI (which happen to be their enclosing neighbours). This way, potential AOI neighbours are discovered with mutual collaborations. As a node moves around, it will constantly discover new nodes and disconnect those that have left its AOI (unless they are enclosing neighbours). A node thus restricts communications with mostly the actual AOI neighbours, independent of the scale of the system. Keeping bandwidth consumption at each node bounded is the key to VON's scalability [2].

However, VON is not very efficient. There are two kinds of extra messages in VON. Firstly, some enclosing neighbours may be not in the peer's AOI, so there will be useless location update messages to be sent. Secondly, boundary neighbours should send back the enclosing neighbour lists which are also extra messages for neighbour discovering.

3 Comparisons between UFR and VON

Our objective is to compare UFR [3] with VON on the scalability and efficiency aspects. In order to give a clear view, we use PPP (perfect peer to peer) to make a comparison. PPP is an ideal case that a peer only sends location update messages to the neighbours within its own AOI. No relevance filtering method can be as efficient as PPP [10]. For VON, it should keep some "remote enclosing neighbours" [4] which are out of AOI for neighbour discovering. It also needs to send back neighbour list from the boundary neighbours to discover the new neighbours. All of these messages are "useless" messages. For UFR, when peers hit the UFR border, it should communicate to update the border. All of the above communications are "useless" messages. Thus, we adopt the extra messages as the metric to evaluate the efficiency of UFR and VON.

We implemented the strips algorithm to calculate the UFR border in a simulated distributed networked environment. Each peer in the world has an area of interest represented by AOI. For VON, we download the source code written by Tainwan Group. We use the random and group based moving model to do the simulation respectively.

3.1 PARAMETER TUNING

Firstly, we keep the world size and the AOI unchanged. We change the peer number to evaluate the scalability characteristic. We use the average update message number per peer per step to be sent including useful and extra messages to measure the performance. We get the following results which are obtained from the random moving model.

Figure 3 shows that as the peer number increases, the average messages per peer per step to be sent in PPP, UFR and VON approximately increase linearly. It implicates that both UFR and VON have good scalability as peer density increases.

An intuitive explanation for this figure may be like this: the peer number increases linearly while the world size is fixed means that the peer density in the world increases linearly. For the random based moving model, we consider

approximately that all the peers are distributed evenly. Thus, the average neighbours within the AOI for a peer will increase accordingly.

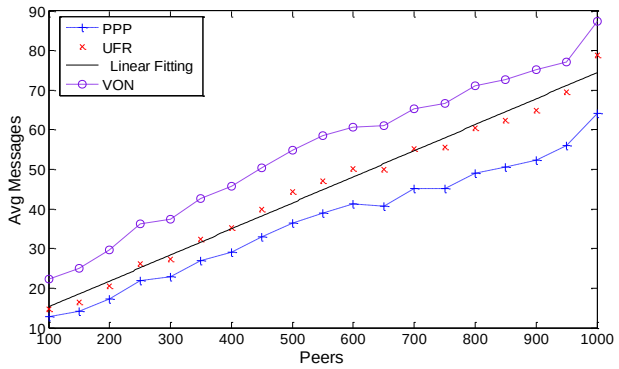

FIGURE 3 Performance of UFR and VON with random moving model

Additionally, another phenomenon we can see from Figure 3 is that the average message number of UFR is smaller than VON. UFR seems more efficient than VON in this case.

Secondly, we set the peer number fixed and change the AOI.

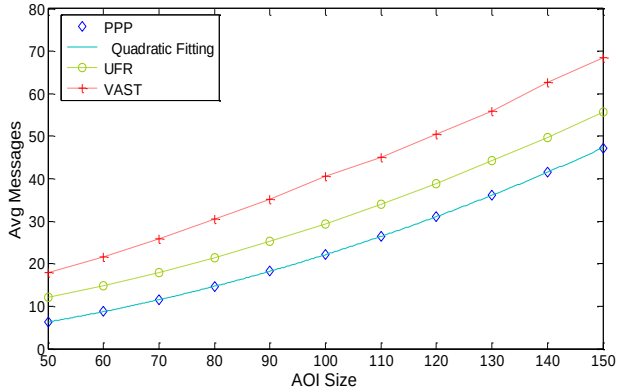

FIGURE 4 Performance of UFR and VON with different AOIs

Figure 4 shows the performance with AOI changed. We can see that, the message numbers of UFR and VON both have an approximate quadratic increase as AOI increases. Similar to the above explanation, when AOI increases linearly while keep peers density unchanged, the average neighbours within the AOI increase quadratic (AOI can be considered as the radius of the area of interest). Additionally, UFR still has better performance than VON in this case.

As for the other parameters, the velocity has the same effect with parameter AOI, the world size is also a relative value of AOI and the peer number, so we don't consider them.

From the above two groups experiments we can conclude that both VON and UFR scale well. UFR seems more efficient than VON in general cases.

3.2 REGRESSION ANALYSIS

In order to give a more accurate evaluation, we make a regression analysis for UFR. Different from the mathematical analysis of strip algorithms, we use the experimental data to give out an approximate model of the performance of UFR. The following are some definitions. We use d to represents the peer density, P_N represents the peer number, S represents the world size (area of the world), aoi represents the radius of the AOI. We can get d from the following expression.

$$d = \frac{P_N}{S} . \tag{1}$$

From the above intuitive experimental results, we can see that the average message number marked by N of UFR is just related to d and aoi. So we can get the target equation of N. a, b, c and d are the coefficients.

$$N_{UFR} = a \times d \times (aoi)^2 + b \times d + c \times (aoi)^2 + d . \tag{2}$$

By using least square analysis, we get the coefficients b and c are close to zero, so neglect them. Thus, we get the two equations for UFR and VON respectively as followed.

$$N_{UFR} = 6.04 + 4.36 \times d \times (aoi)^2 , \tag{3}$$

$$N_{PPP} = 1.98 + 4.1 \times d \times (aoi)^2 . \tag{4}$$

If we let e be equal to $d*(aoi)^2$. Actually, e is a reflection of the average neighbour number of a peer. We can get the following expression

$$(N_{UFR} - N_{ppp}) / N_{ppp} = (4.06 + 0.26e)/(1.98 + 4.1e) . \tag{5}$$

We define this as "useless rate" for UFR. Figure 5 shows the tendency of "useless rate". From Figure 5 we can see that as the average neighbour number increases, the "useless rate" seems to decrease. It is a little tricky. But actually we can assume the limit case: If the AOI is enlarged to the whole world, all the peers will be in the neighbour list for each peer, every location update messages will be "useful".

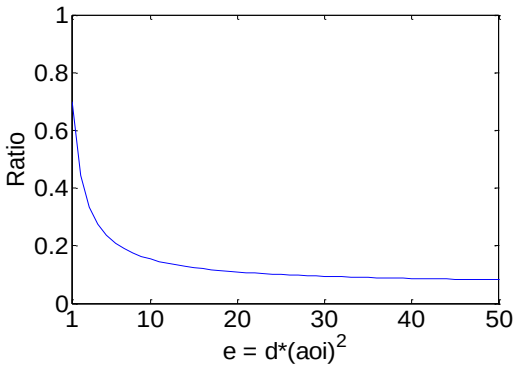

FIGURE 5 Useless rate for UFR with random moving model

3.3 GROUP BASED MOVING MODEL

We also do some experiments on group based moving model. Different from the random moving model, group based moving model organizes peers in some groups. All the peers in the same group go toward the same destination during a period. The following figures show the performance of UFR and VON in group based model. The performance of UFR and VON with group based moving model fluctuates much more than with random moving model. UFR also has better performance than VON (see Figure 6).

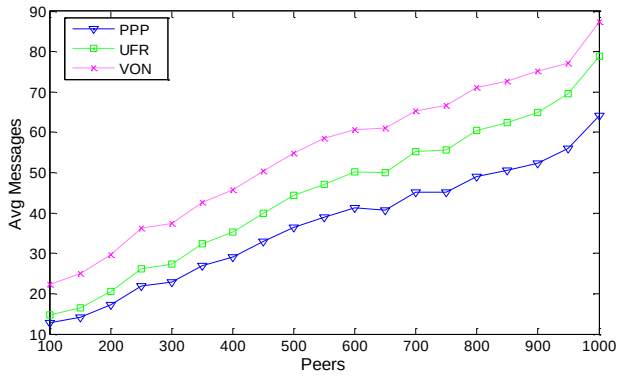

FIGURE 6 Performance of UFR and VON with group based moving model

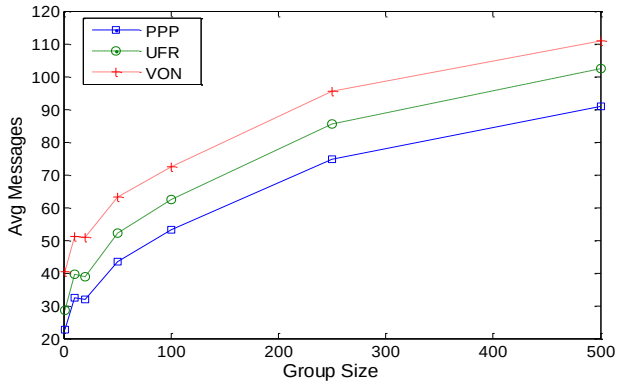

FIGURE 7 Performance variety with different group sizes

4 Optimization on stripes algorithms

The original "strip" algorithm to calculate the UFR border is not very efficient. When the two peers are going closer, the updates messages will be increased obviously. In addition, the "strip" only calculates the UFR border by using the positions of peers but not considers the speed and trajectories of the peers. In order to reduce useless update messages and relieve the increasing update messages when peers going closer, OPT-UFR is proposed.

Figure 8 shows the distribution of the border update messages. The AOI is set to be 100. All the border update messages are classified by the distance between the pair of peers when updates occur. We can see clearly that most of the update messages are occurred within 200.

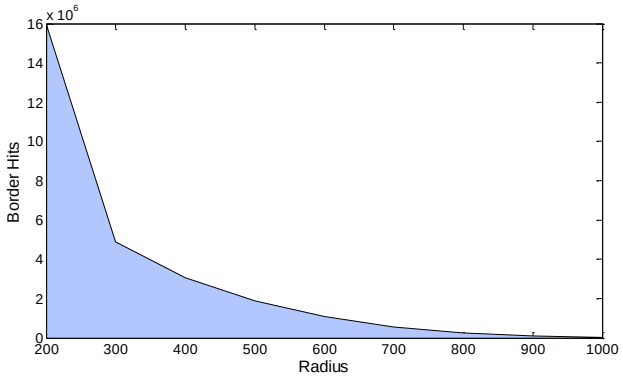

FIGURE 8 Update messages distribution in UFR with random moving model

OPT-UFR algorithm considers the predicted trajectory of the two nodes. Thus, when the UFR borders are calculated, we can set the border more reasonably. We only consider some simple cased now, but from a different point of view, the easier the better. The simulation experimental results show that OPT-UFR has better performance than the original "strip" algorithm.

4.1 OPT-UFR DESCRIPTION

Firstly, when peers hit the border, we should calculate the new borders by using "strip" method. Secondly, we calculate out the bisector and the predicted trajectories of this pair of peers. If the borders should be adjusted according to the OPT-UFR regulations, we will adjust the borders according to the corresponding cases. Third, three different cased are listed here.

```
//get bisector
line bisector=GetBisector(peer1, peer2);
//get predicted trajectory
line traj1=GetTraj(peer1);
line traj2=GetTraj(peer2);

if(Intersect(traj1, bisector) && Intersect(traj2, bisector)) {
        double dist1=GetDistance (peer1, traj1, bisector);
        double dist2=GetDistance (peer2, traj2, bisector);
        SetBorderInverseProp (peer1,peer2, dist1, dist2,
        bisector);
} else if (Intersect (traj1, bisector) | | Intersect(traj2,
bisector)) {
        if(Intersect(traj1, bisector) {
                SetBorderNear(peer2, peer1, bisector);
        }else {
                SetBorderNear(peer1, peer2, bisector);
        }
} else {
        //do nothing
}
```

FIGURE 9 Pseudo code of OPT-UFR

1) If the two trajectories both have intersecting points with the bisector, we try to let the two peers arrive the border at the same time. In order to do this, we let the distance between the peer and its UFR border be inverse proportion to the distance between this peer and the intersecting point of its trajectory and the bisector.

2) If there is only one intersecting point, it implicates that one peer is leaving the UFR border, so we can set the border very near to this leaving peer and this can let another peer goes further without hitting the border.

3) If there is no intersecting node, it means that two nodes are both leaving each other. Thus, we don't need to adjust them anyway.

4.2 EXPERIMENTAL RESULTS

We test OPT-UFR algorithm with simulation programs. Random moving model and group based moving model are used as the moving model. The experimental settings are listed by following: world size (1000×1000), velocity (5), AOI (100), time step (500). We change the peer size to test different performance with different densities and collect the number of border update messages as our metric.

We get the following results. Figure 10, Figure 11 and Figure 12 shows the comparison of the OPT-UFR and "strip" algorithm with random moving model.

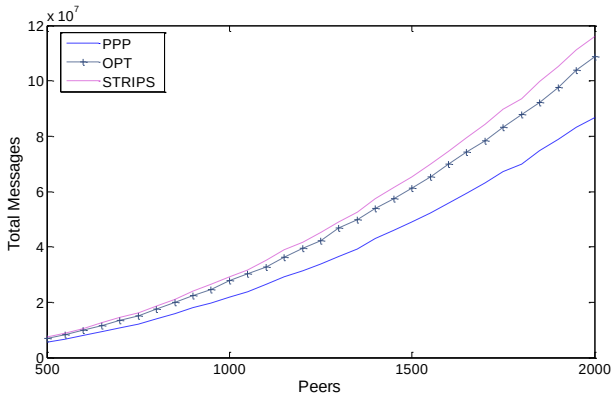

FIGURE 10 Comparison between OPT-UFR and Strips algorithm with random moving model (a)

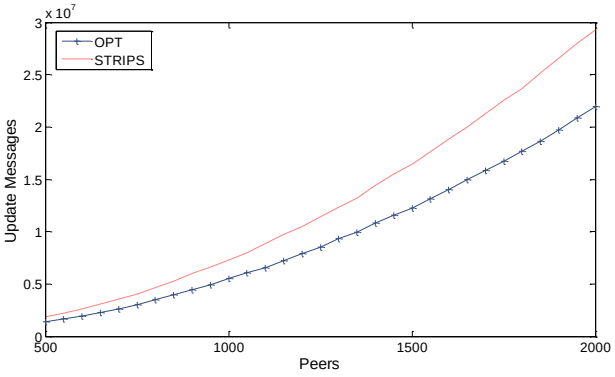

FIGURE 11 Comparison of OPT-UFR and Stripes algorithm with random moving model (b)

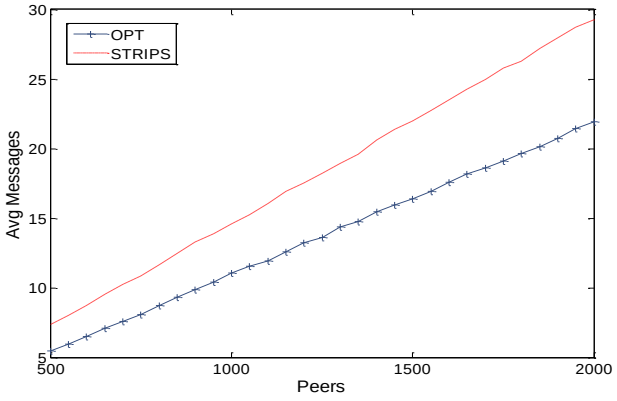

FIGURE 12 Comparison of OPT-UFR and Stripes algorithm with random moving model (c)

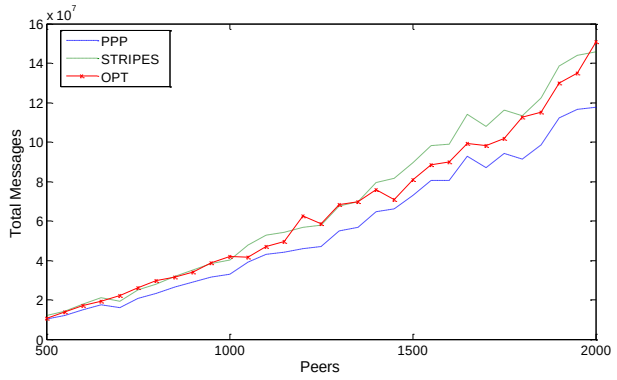

FIGURE 13 Comparison between OPT-UFR and Strips algorithm with group based moving model (a)

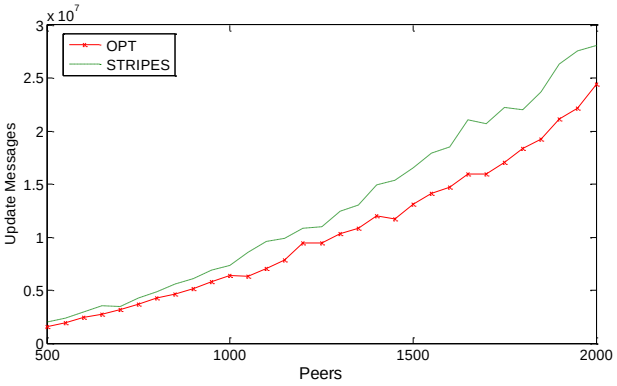

FIGURE 14 Comparison between OPT-UFR and Strips algorithm with group based moving model (b)

Figure 13, Figure 14 and Figure 15 show the comparison of the OPT-UFR and "strip" algorithm with group based moving and group based moving model respectively. The group size is set to be 50. We can see that OPT-UFR always has better performance with both two moving models.

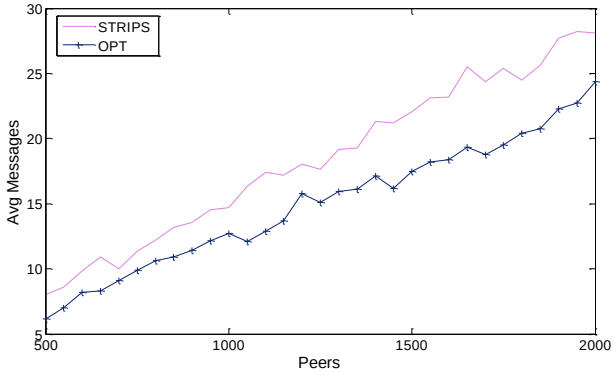

FIGURE 15 Comparison between OPT-UFR and Strips algorithm with group based moving model (c)

5 Peer joining

One of the key problems is the peer joining procedure for UFR scheme. In order to run the UFR algorithm, we should know all the existing peers to calculate the UFR borders. There will be a heavy traffic load for the joining peer when there are so many existing peers.

In order to solve this problem, we propose a feasible peer joining strategy which is called "joining by step". The main idea of the method is to delay some UFR border calculations for the un-urgent peers. We only calculate UFR borders for the "nearer peers" for the joining peer.

We first select the nearest peer to the joining peer as its proxy and get the UFR border list from the proxy. The objective is to get the "at least" distance from the joining peer to each of the existing peer. What we can only use is the UFR border list from the proxy. Then, we divide these UFR border into two groups. Group one is that both the joining peer and the proxy are at the same side of the border. The other group is that the joining peer and the proxy are at the opposite side of the border. We only calculate the distances to the borders from group one. For the borders from group two, we are not sure the "at least" distance from the joining peer to the corresponding peer related to this border, because, the joining peer is not in the safe area for the corresponding peer related to this border. At this time, we set the distance between these two peers to be 0.

After the distance calculations, we get the "at least" distance from the joining peer to each of the existing peers in the world. We use variable dis to represent this distance. We then calculate out the steps can be delayed for each peer in the world by the following equation.

$$steps = \begin{cases} 0, dis < aoi + 2v \\ \dfrac{dis - aoi}{2v}, dis > aoi + 2v \end{cases} . \quad (4)$$

The variable *steps* represents the time steps of this connection refers to this peer can be delayed. *aoi* is the AOI radius for this pair of peers. *V* represents the velocity of the two peers. We assume all the peers have the same AOI and velocity.

Thus, the joining peer can firstly connect the existing peers which the delay steps are 0. After a time step, it can continually connect the peers which the delay steps are 1. So on, we can divide the peer joining procedure into many steps. Additionally, during the peer joining period, though many peers may not know the new joining peer, the consistency actually is kept all the time. Because those peers who don't know the joining peers won't have any interaction with the joining node.

5.1 FEASIBILITY OF DELAYED JOINING

Finally, we discuss the feasibility of joining by step. The most important feature should be satisfied for the method is the appropriate group division. The distribution of the peers in each group should be average. We do some experiments to get the distribution of the peers according to the steps can be delayed and get the following Figures 16.

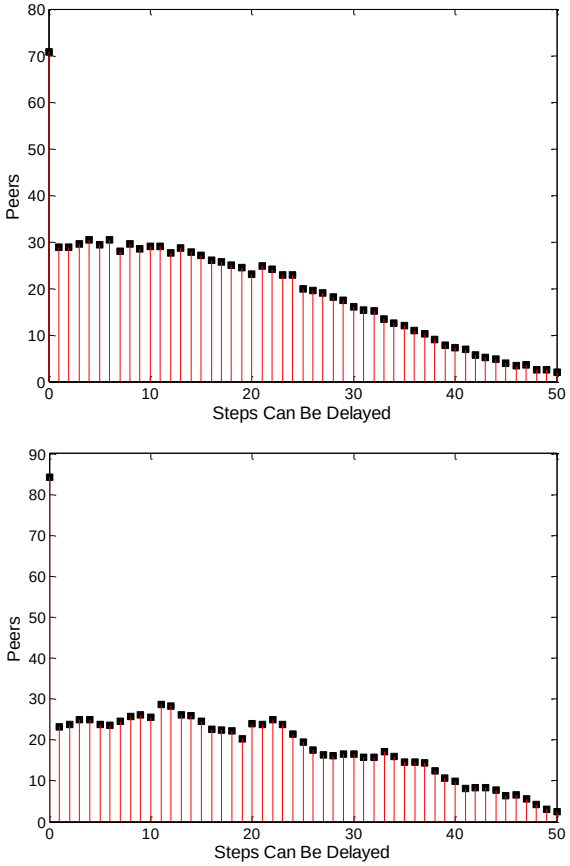

FIGURE 16 Peer distribution according to steps can be delayed

The experimental settings are like these, world size (1000×1000), AOI (100), velocity (5), peers (1000). These two figures are separately for the random moving model and the group based moving model. We let a new peer join the world several times and get the average peer

distributions. From the Figure 16 we can see that the number of peers in each group is average. The red point represents the average neighbours in this world for each peer. Thus, we can conclude that only about average neighbours number peers should be connect immediately for the peer joining. So the "joining by step" strategy seems to be feasible.

6 Other issues

Another issue should be considered is the computational complexity of UFR. Different from VON, each peer in UFR should keep strips of all the other peers in the world. Thus, each step one peer should find out which UFR borders it goes across and should be updated. When the peer number is very large, the computational complexity will be a problem. Fortunately, some data structures and algorithms focus on this problem have been proposed and the computational complexity can be controlled under O(nlgn).

7 Conclusions

We have proposed an optimization algorithm In order to reduce useless update messages and relieve the increasing update messages when peers going closer. The algorithm is based on UFR. When peers hit the border, we should calculate the new borders by using "strip" method, then calculate out the bisector and the predicted trajectories of this pair of peers. we will adjust the borders according to the corresponding cases when adjustment needed according to the OPT-UFR regulations.

By testing OPT-UFR algorithm with simulation programs, results show that OPT-UFR has better performance than the original "strip" algorithm. Otherwise, "delayed joining" algorithm can solve the heavy traffic problem for the joining node.

OPT-UFR has been proposed to relieve increasing useless update messages when peers going closer. The UFR algorithm, as described above, is applied only to a pair of agents. In multi-peers environment, a significant computation burden will appear when we use the UFR in every peer. This shows that a fully suitable algorithm for the entire system needs further research.

References

[1] Oliveira J C, Georganas N D 2003 VELVET: An Adaptive Hybrid Archi-tecture for Very Large Virtual Environments *Presence* **12**(6) 555-80

[2] Hu S Y, Chen J F, Chen T S 2006 VON: A Scalable Peer-to-Peer Networ-k for Virtual Environments *IEEE Network*

[3] Makbily Y, Gotsman C, Bar-Yehuda R 1999 Geometric Algorithms for Message Filtering in Decentralized Virtual Environments 39-46

[4] Keller J, Simon G 2002 Towards a Peer-to-Peer Shared virtual Reality *Proc.22nd ICDCS(Wksps.)* http://solipsis.netofpeers.net

[5] Benford S, Fahlen L, Greenhalge C, Bowers J 1994 Managing mutual awareness in collaborative virtual environments *Proceedings of ACM SIGCHI Conference on Virtual Reality and Technology (VRST '94)*

[6] Capps M, Teller S 1997 Communication visibility in shared virtual worlds *Proceedings of the 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Environments Cambridge MA*

[7] Clarson C, Hagsand O 1993 DIVE - a platform for multi-user virtual environments *Computers & Graphics* **17**(6) 663-9

[8] Earnshaw R A, Chilton N, Palmer I J 1995 Visualization and virtual reality on the Internet *Proceedings of the Visualization Conference Jerusalem Israel*

[9] Lea R, Honda Y, Matsuda K, Matsuda S 1997 Community Place: Architecture and performance *Proceedings of Second Symposium on the Virtual Reality Modeling Language (VRML '97)* 41-50

[10] Hu S Y, Liao G M 2004 Scalable Peer-to-Peer Networked Virtual Environment *Proc. ACM SIGCOMM Wksp. on NetGames* 129–33

[11] Macedonia M R, Brutzman D P, Zyda M J, Pratt D R, Barham P T, Falby J, Locke J 1995 NPSNET: A multi-player 3D virtual environment over the Internet *Proceedings of the 1995 ACM Symposium on Interactive 30 Graphics* 93-4

[12] Singhal S, Zyda M 1999 Networked Virtual Environments: Design and Implementation *New York: ACM Press*

[13] Alexander T 2003 Editor, Massively Multiplayer Game Development *Charles River Media*

[14] Stoica I 2003 Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications *IEEE/ACM Trans Net* **11**(1) 17-32

[15] Knutsson B 2004 Peer-to-Peer Support for Massively Multiplayer Games *Proc. INFOCOM* 96-107

[16] Chen K T, Huang P, Lei C L 2007 Game Traffic Analysis: An MMORPG Perspective to appear *Comp Networks* **51**(3)

[17] Funkhouser T A 1995 RING: A Client-Server System for Multi-User Virtual Environments *Proc Symp Interactive 3D Graphics* 85-92

[18] Singhal S K, Cheriton D R 1996 Using Projection Aggregations to Support Scalability in Distributed Simulation *Proc ICDCS* 196-206

Authors

Huijuan Wang, 3/23/1982, Langfang, Hebei, China.

Current position, grades: lecturer.
University studies: Bachelor degree and Master degree of Engineering in computer science and technology in Nankai University.
Scientific interest: parallel and distributed algorithms and embedded system.
Publications: more than a dozen papers.

Quanbo Yuan, 5/12/1984, Handan, Hebei, China.

Current position, grades: lecturer in North China Institute of Aerospace Engineering.
University studies: Bachelor degree in North China Institute of Aerospace Engineering, Master degree study in Tianjin Polytechnic University.
Scientific interest: computer science.
Publications: more than a dozen papers.