TriCast: Triangulation with multicast support for P2P virtual environments

Eliya Buyukkaya, Maha Abdallah, Romain Cavagna

TRICAST: TRIANGULATION WITH MULTICAST SUPPORT FOR P2P VIRTUAL ENVIRONMENTS

Eliya Buyukkaya, Maha Abdallah and Romain Cavagna

LIP6, University of Paris 6
{Eliya.Buyukkaya, Maha.Abdallah, Romain.Cavagna}@lip6.fr

ABSTRACT

Peer-to-peer (P2P) architectures have recently become a popular design choice for building virtual environments (VEs). In P2P-based VEs, peer connectivity between the different VE users depends on their positions in the virtual world regardless of the underlying physical network, which is subject to redundant hops and delay. In this paper, we propose Tri-Cast, a fully-distributed P2P architecture to support virtual environment applications by combining a triangular logical network used for discovering the interest group of an event in the virtual world with an application layer multicast support, used for multicasting the event message to the group members. Simulation results show that the proposed physical multicast tree algorithm reduces the cost of message transmission to group members.

Keywords— Networked virtual environments, peer-to-peer systems, proactive ad-hoc routing protocol, triangulation, multicast tree

1. INTRODUCTION

Networked virtual environments (NVEs) are computer generated virtual worlds where users navigate and interact with their surroundings and each other through virtual representations called *avatars*. NVEs are traditionally supported by client/server architectures. The server (or a cluster of servers) keeps the world data, calculates the states and disseminates the state updates to users. A user informs the server about the events that she/he creates and is informed by the server about the states of other objects in the virtual world. However, centralized architectures can lead to high communication and computation loads at the servers, which quickly become a bottleneck point during peak operations. To overcome these problems inherent to centralized solutions, P2P overlay networks are emerging as a promising alternative for VEs [1, 2, 3]. In P2P systems, the overall system load is distributed among all participating users. By aggregating and sharing the users' resources, the system can achieve high scalability in a cost-effective manner.

The key issue of VE architectures is, in case of any event occurring in the virtual world (e.g., such as movement of an avatar), the diffusion of the event update message to the players interested in the event. Architectures deal with this issue based on a two-phase process. The first phase is the determination of the group of players (peers) interested in an event. The second is the transmission of the event update message to the group members. In order to realize the first phase, a logical network based on the position of players in the virtual world (e.g., game terrain) is built on top of a physical network. The logical network defines the way peers connect to other peers with whom they can interact and exchange messages in the virtual world. It is important to note that the logical connections between peers are constructed independently from peers' physical location, which may lead to logical neighbor peers far away from each other in the real world (i.e., physical network). The message transmission phase is realized by the physical network. How logical and physical networks can most efficiently be mapped to each other to support VE is thus of central concern to P2P VEs. A well-designed architecture should take into consideration the constraints due to both logical and physical networks.

Several P2P architectures dealing with event update message diffusion to interested peers have been proposed [1, 2, 3]. These architectures mainly address the problem of a logical network management, i.e., logical peers connection and interested peers' discovery. For the event update transmission, these solutions simply assume unicast message delivery, that is, sending one event update message per interested peer. However, the unicast message transmission solution is not efficient enough since on the physical path from the sender peer (i.e., event source peer) to destination peers (i.e., interested peers), there may be several peers receiving the same event update message more than once. Therefore, the message transmission scheme leads to such relay peers, which can also be destination peers, receiving the same event update only once is a more efficient solution compared to the unicast one, which is the contribution of this paper.

In this paper, we propose *TriCast*, a fully-distributed P2P architecture for VEs based on a triangular logical network on top of an ad-hoc network. The reason behind our decision to build the architecture on an ad-hoc network is to provide a flexible physical network that permits users to enter the system anywhere and anytime they want through laptops, PDAs or consoles, which is the trend of near future VE system. A target application can be a game application combining a number of friends using Sony Playstation Portables to play together.

The interest group of an event occurring in the virtual world

This work is supported by the "Département de Paris" in the context of the KEP1 project of the Cap Digital FEDER 2 program. We thank Arnaud Kaiser, Khaled Boussetta and Nadjib Achir for valuable and useful discussions.

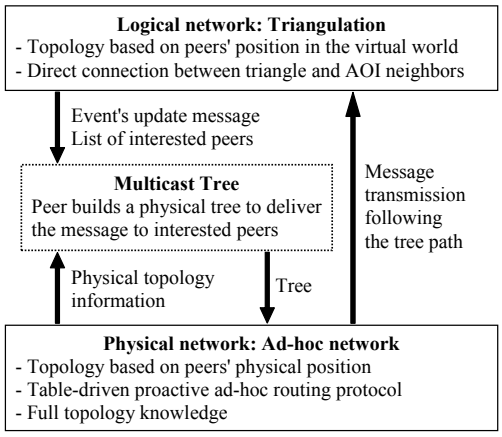


Fig. 1. System model

(i.e., the destination group of the message) is determined by a triangular logical network based on users' position in the virtual world. Our previously proposed overlay, called Triangulation [4], provides connectivity between peers based on virtual proximity, where peers have a limited number of neighbors independently from the size of the network. The major advantage of Triangulation is that it drastically reduces the number of maintenance messages due to peers' movements in the game world compared to other well known triangulation algorithms, such as Delaunay triangulation [5]. When a peer sends the message of an event to group members (i.e., peers interested in the event) determined by Triangulation, the peer builds a physical multicast routing tree spanning all group members to reduce the message transmission cost. The message is then forwarded following the tree path in the physical network. Our main contribution is thus the efficient transmission of a state change message through a multicast tree in a fully-distributed P2P architecture based on Triangulation on top of an ad-hoc network.

The remainder of the paper is organized as follows. Section 2 presents related works. Section 3 defines the underlying system model and gives some background on the ad-hoc routing protocol and Triangulation algorithm. Section 4 details the multicast scheme in our architecture. Performance evaluations are given in section 5. Finally, section 6 concludes this work.

2. RELATED WORK

A number of P2P designs have been recently proposed to support VEs. MOPAR [1] partitions the virtual world into fixed-size, continuous hexagonal cells, each of which is assigned a unique master peer within the cell. A cell's master keeps track of all other slave peers in its cell, and regularly exchanges this list with the master peers of neighboring cells. Slave peers are notified of new neighbors by their masters, while other message exchange is performed directly between the slaves.

VON [3] discusses a Voronoi-based [6] partitioning of space in the context of VEs. Each peer maintains a Voronoi diagram of its AOI neighbors, and keeps a direct connection to all of them. Neighbor discovery is achieved by neighbor cooperation and mutual notification. If crowding occurs, however, mechanisms that connect a peer to all its AOI neighbors might lead to computing and bandwidth requirements exceeding peers' capabilities. To deal with this issue (i.e., a growing number of users within the AOI), AOI-Cast [7] based on VON, proposes to build a spanning tree across all AOI neighbors of a peer, thus reducing bandwidth usage. However, the spanning tree is built based on peers' position in the virtual world without any consideration of the underlying physical network, which may lead redundant hops and unacceptable delay for the application.

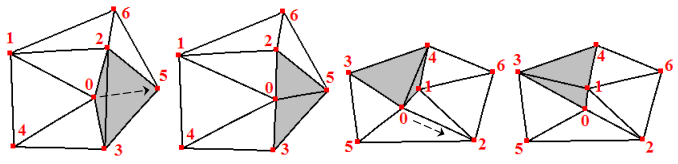
Loader [8] uses Pastry to distribute objects to peers, and Scribe on top of Pastry for object update message dissemination. In Loader, a multicast group is created for each object to disseminate the object update messages to interested peers. Similar to Loader, P2P propagation [9] creates a multicast group for each object in the world. In P2P propagation, for each multicast group, the peer having the minimum delay to other peers inside the group, is selected as relay to forward object update messages to other peers. Thus, a message from server passes through relays to reach to clients.

3. TRICAST: SYSTEM MODEL

Our system consists of two layers: 1) Physical network supported by a table-driven proactive ad-hoc routing protocol, 2) Logical network based on Triangulation [4] (see Figure 1). The transmission of an event message to interested peers is realized by the collaboration of two networks. The group of interested peers is determined by the logical network. With the physical topology information provided by the physical network, the peer creating the event builds a multicast tree spanning all interested peers. The message is then forwarded in the physical network following the tree path. A peer in the tree sends one message per child containing the event state information and the information of the subtree whose root is this child.

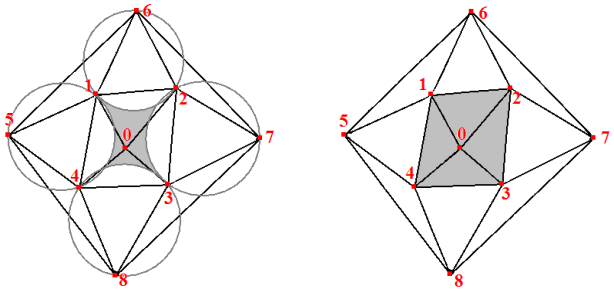
3.1. Proactive ad-hoc protocol - OLSR

The reason behind our choice of building the system on a proactive routing protocol is based on the fact that in this protocol, the routes to all destinations within the network are known and maintained before use, which is required by VE applications due to delay constraint. There exist various proactive ad-hoc protocols. In this paper, we focus on and design our solution in the context of the optimized link state routing protocol (OLSR) [10], which is a proactive protocol developed for mobile ad-hoc networks [11] but can also be used on other wireless ad-hoc networks. In OLSR, the global topology knowledge is obtained by partial topology information exchange between peers. Every peer sends periodically broadcast messages including the distance information of specific neighbors. Through this information, a peer builds a routing table and calculates the shortest route to every peer by using Dijkstra shortest-path algorithm. Thus, the route information to any peer is available when requested. Besides, the flooding of broadcast messages is optimized through multipoint relays (MPR) which are responsible



(a) (b)

Fig. 2. Flip during peer *0*'s movement



(a) Delaunay (b) Triangulation

Fig. 3. Peer *0*'s flip-free area

for forwarding control traffic. Each peer selects a set of its 1-hop neighbors as MPRs to cover all 2-hop neighbors. MPRs are used to avoid sending twice the information packet in the same region of the network and to reduce the size of broadcast messages (i.e., the broadcast message of a peer p includes only the neighbors that select p as one of their MPRs).

3.2. Triangulation

Triangulation [4] provides connectivity between peers depending on their positions in the virtual world. Triangulation for a set of vertices V in a 2-D plane is a triangulation $T(V)$ such that no vertex of V is inside any triangle in $T(V)$. Note that a position change of a vertex in V can violate the basic property of Triangulation. Let *0, 2, 3, 5* be vertices in V and *032* and *523* be triangles in $T(V)$ (see Figure 2a). If *0* moves towards peer *5* and passes across the base of *032*, triangles *032* and *523* violate the Triangulation property. To meet the Triangulation condition, an edge flip operation is performed by switching the common edge *32* for the common edge *05* resulting in two valid triangles *052* and *035*. In Figure 2b, when peer *0* moves towards peer *2*, peer *1* passes across the base of triangle *140*. As a consequence, the movement of peer *0* causes a flip operation between triangles *140* and *304* and creates triangles *143* and *130*.

In Triangulation, apart from flip operations occurring while crossing a triangle base, all other flip operations that might be triggered during a peer movement are eliminated. Therefore, compared to other triangulation algorithms such as Delaunay triangulation [5], Triangulation decreases maintenance cost by reducing the number of connection changes due to users' movement. To do so, Triangulation maximizes the area where a peer is allowed to move without triggering any flip operation. The gray area in Figure 3 shows the area where peer 0 can freely move without generating a flip operation for Delaunay and Triangulation algorithms.

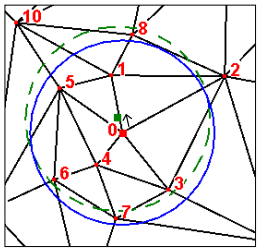


Fig. 4. Peer *0*'s movement towards peer *1*
Circles show *0*'s AOI boundary during its movement

When an event occurs in the virtual world (e.g., an avatar position change), the group of peers interested in the event (i.e., peers to whom the event is visible) is determined through Triangulation. Each peer, say p, keeps a peer list containing p's triangle neighbors (i.e., peers with whom p forms a triangle) and p's AOI neighbors (i.e., peers to whom p is visible). Each time p changes position, p informs the peers in its peer list. p's triangle neighbors inform p about p's new triangle neighbors due to eventual flip operations. p's boundary neighbors (i.e., p's outermost neighbors) inform p about p's new AOI neighbors. p reconstitutes its peer list by removing the peers from its peer list that are no longer triangle or AOI neighbors, and by adding new triangle and AOI neighbors into the peer list. Neighbor discovery is thus achieved by neighbor cooperation and mutual notification. In Figure 4, peer *0*'s triangle neighbors are *1, 2, 3, 4, 5* and peer *0*'s AOI neighbors are *1, 3, 4, 5, 6, 7*. When *0* moves towards peer *1, 0* informs all peers in its peer list (i.e., p's triangle and AOI neighbors). *0* removes peer *7* from its AOI neighbor list since *7* now resides outside its AOI. Peer *1* (or peer *2*) informs *0* about the existence of peer *8, 0* and *8* exchange message and *0* adds *8* into its peer list as a AOI neighbor.

When a peer p moves, in order to inform the peers in its peer list (i.e., p's triangle and AOI neighbors) about p's state change, p builds a multicast tree spanning all peers in the list through the physical topology information in p's routing table. p then sends to each of its tree children one message containing p's state information and the related subtree information (i.e., the information of the subtree whose root is this child). A peer receiving p's message sends to each of its own tree children one message containing p's state information and the related subtree information. The message is thus forwarded in the physical network following the tree path.

4. MULTICAST TREE

In a system based on OLSR, if a peer p wants to send a multicast message to a group of peers, either the message is flooded over the network through multipoint relays (MPRs), or p sends one message per peer in the group through unicast. Even though the flooding is optimized by MPRs, the broadcast solution has a high cost if the group consists of a few peers. To disseminate a message to a group, we thus improve the unicast solution to build a multicast tree in order to reduce message transmission cost. The minimum cost path from p to any group member is

$routingTable(p, p_i).path : p \rightarrow p_5 \rightarrow p_6 \rightarrow {p_i}^*$
$routingTable(p, p_j).path :$
$p \rightarrow p_0 \rightarrow p_1 \rightarrow p_2 \rightarrow {p_m}^* \rightarrow p_3 \rightarrow {p_j}^*$
$routingTable(p, p_k).path : p \rightarrow p_0 \rightarrow p_1 \rightarrow p_4 \rightarrow {p_k}^*$
$routingTable(p, p_m).path : p \rightarrow p_0 \rightarrow p_1 \rightarrow p_2 \rightarrow {p_m}^*$

Fig. 5. Unicast

$$\begin{array}{c} \nearrow p_5 \rightarrow p_6 \rightarrow {p_i}^* \\ p \rightarrow p_0 \rightarrow p_1 \rightarrow p_2 \rightarrow {p_m}^* \rightarrow p_3 \rightarrow {p_j}^* \\ \searrow p_4 \rightarrow {p_k}^* \end{array}$$

Fig. 6. Basic Tree

extracted from p's routing table. A *basic tree* is built based on the minimum cost paths. To reduce the cost of a basic tree, peers change connections and attach to the tree from lower cost points. The resulting reduced-cost tree is *the relaxed tree* which determines the message routing path.

4.1. Unicast

The basic solution to disseminate the update message of a peer p to the peers in p's peer list is to send one unicast message per peer in the peer list. To unicast the message to a peer q in p's list, the path information from p to q is extracted from the entry of p's routing table corresponding to q. The message is thus forwarded along the minimum cost path from p to q. Let p_i, p_j, p_k, p_m be peers in p's list, $routingTable(p, q)$ be entry of p's routing table corresponding to q and $routingTable(p, q).path$ gives the minimum cost path from p to q. Figure 5 shows the unicast forwarding of p's update message from p to p_i, p_j, p_k, p_m, which requires a total of 17 message transmissions.

4.2. Basic tree

The tree of a peer p spans all peers in p's peer list. A peer in p's peer list is marked by a star (*) and called a *terminal* peer. Any peer which is not a terminal peer is called a *nonterminal* peer.

When p sends an update message to its terminal peers, the basic tree to determine the message forwarding path is built by finding common peers on the unicast forwarding paths from p to any terminal peer and then sending only one message to each common peer. Let r, s be two terminal peers and q be the last common peer on the path from p to r and from p to s. Since the path from p to q contains the same peers for any path, all peers from p to q are common for the path from p to r and from p to s. p thus routes one message to q. q then routes one message to r and one message to s.

The basic tree of p in Figure 5 is shown in Figure 6. p sends to p_5 one message containing p's update information (i.e., p's new logical coordinates) and the forwarding information related to p_5 (i.e., subtree information whose root is p_5). Similarly, p sends to p_0 one message containing p's update information and the forwarding information related to p_0. When a peer receives the message, the peer forwards to each of its tree children one message consisting of p's update information and the subtree information related to this child. p's update message dissemination through the basic tree requires a total of 11 messages compared to 17 in the unicast solution (Figure 5).

$peerCost(p_6) = weight(p, p_5) + weight(p_5, p_6)$
$peerCost(p_3) = weight({p_m}^*, p_3)$
$peerCost({p_m}^*) = weight(p_1, p_2) + weight(p_2, {p_m}^*)$
$peerCost({p_k}^*) = weight(p_1, p_4) + weight(p_4, {p_k}^*)$

Fig. 7. Peer cost

$$\begin{array}{c} \nearrow p_5 \rightarrow p_6 \rightarrow {p_i}^* \rightarrow p_7 \rightarrow {p_m}^* \rightarrow p_3 \rightarrow {p_j}^* \\ p \rightarrow p_0 \rightarrow p_1 \rightarrow p_4 \rightarrow {p_k}^* \end{array}$$

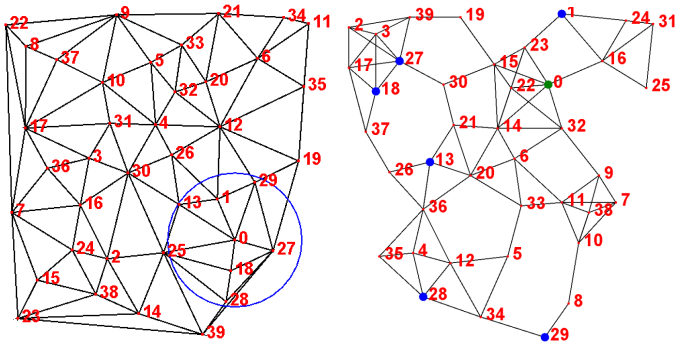
Fig. 8. Relaxed Tree

4.3. Relaxed tree

A relaxed tree is deduced from a basic tree by changing forwarding paths between peers to reduce the tree cost. The cost of a tree is defined as the total edge weight. The weight of an edge can be defined according to the characteristics of the underlying physical network. In a simple design, the same weight value is associated to all edges in the network. In an energy-constrained ad-hoc network, an edge weight can be defined in terms of energy level of two end peers of the edge between them.

A *branch* of a tree is a path of nonterminal peers having at most one child except the end peer. The end peer of a branch is either a terminal peer or a peer having at least two children. The branches of the tree in Figure 6 are $(p_5, p_6, {p_i}^*), (p_0, p_1), (p_2, {p_m}^*), (p_3, {p_j}^*)$ and $(p_4, {p_k}^*)$. The *peer cost* of a peer p in the tree is the total of the edge weight from the parent of the start peer of p's branch, to p. Figure 7 shows the peer cost value of peers $p_6, p_3, {p_m}^*, {p_k}^*$ in the tree shown in Figure 6.

A peer p in a basic tree seeks to reduce its peer cost by connecting to a peer q different from the peers in its branch and in its descendants where the cost of the path from q to p is the least and less than its peer cost, $routingTable(q, p).cost \ll peerCost(p)$. If such a q exists, then p is connected to the tree through q and the ascendant peers in p's branch are removed from the tree. In Figure 6, if ${p_m}^*$ finds that the cost of the path from ${p_i}^*$ to ${p_m}^*$ is less than ${p_m}^*$'s peer cost and ${p_i}^*$ is the peer through whom ${p_m}^*$ is connected to the tree with a minimum cost, then ${p_m}^*$ becomes the descendant of ${p_i}^*$ and p_2 is removed from the tree. The resulting tree is shown in Figure 8. This process continues until there are no more peers in the tree that can reduce its peer cost. The resulting tree is the *relaxed*



(a) Logical Network (b) Physical Network

Fig. 9. Circle shows peer *0*'s AOI. Pointed peers are peer *0*'s logical neighbors.

tree which has a lower cost than the basic tree. The update message is forwarded over the physical network following the relaxed tree path.

5. EXPERIMENTAL EVALUATIONS

In this section, we present the results of our simulation. The evaluation is performed with 40 peers. For the avatar movement, we use the traces of user movement collected by Liang et al. [12] in Second Life, a popular online virtual world. The virtual world in Second Life is made up of regions, each of which is 256m x 256m. Figure 9a shows an instant of the virtual world topology while Figure 9b shows the physical topology of the network. The circle in Figure 9a shows the AOI boundary of peer *0*. Peer *0*'s logical neighbors (i.e., triangle and AOI neighbors) are indicated by the pointed peers in Figure 9b.

We define the weight of an edge from p to q, such that p to q are direct neighbors in the physical network, as follows :

$$weight(p,q) = 1 - (m \times energy(p) + n \times energy(q))$$

where $energy(p)$ gives the percentage of p's residual energy level, and m and n are constants determined by the ratio of energy consumption in send mode to receive mode in message transmission [13]. Figure 10 shows the weight of the edges between p_1 and p_2 where $m = 0.80$ and $n = 0.20$, meaning that message transmission from consumes four times more energy than message reception. The reason behind our decision to use an energy based weight metric is to measure the network lifetime by taking into account the energy capacity of participating machines and energy consumption during message transmission.

Peers start the simulation with maximum residual energy level. When a peer moves, the peer sends its update message to the peers in its peer list following one of three routing algorithms : unicast, basic tree, relaxed tree. A message transmission reduces residual energy level of peers on the message forwarding path according to the energy consumption model presented in [13]. The simulation is ended when a peer in the system runs out of energy. Figure 11 compares peers' lifetime

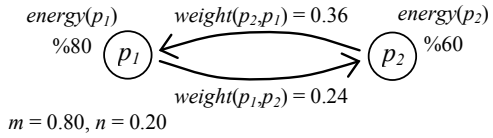


Fig. 10. Weight of the edges between p_1 and p_2

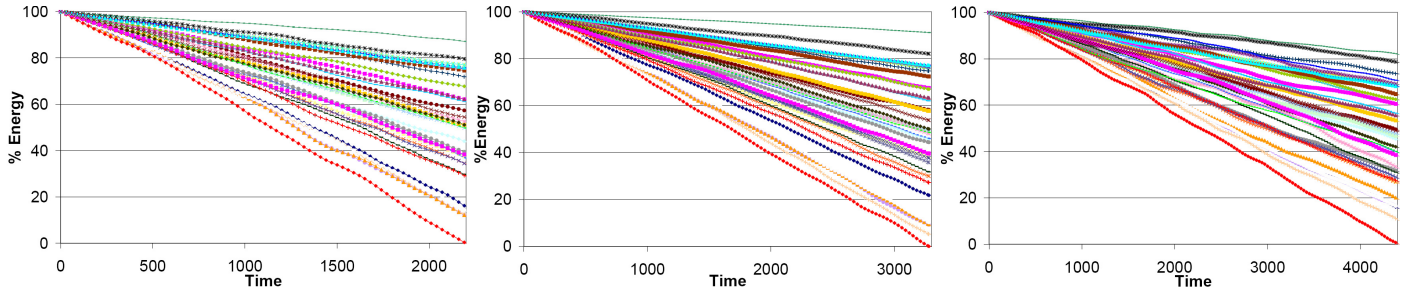
under the three routing algorithms. Basic tree routing (Figure 11b) yields less energy consumption and longer network lifetime compared to unicast routing (Figure 11a). Relaxed tree routing (Figure 11c) optimizes further and provides longer network lifetime compared to basic tree routing (Figure 11b).

Figure 12 shows the residual energy level of peers *36, 14, 2, 25* under three routing algorithms. The position of these peers within the physical network can be seen in Figure 9b. Peers *36* and *14* are located in the middle of the network while peers *2* and *25* are located at the corner. For all peers, the most energy consuming algorithm is unicast routing. Relaxed tree routing is the least energy consuming for peers *36* and *14* while for peers *2* and *25* basic tree routing is the best. This is because relaxed tree routing changes forwarding path to reduce the message transmission cost by replacing peer having low residual energy on the path with peer having high residual energy in order to increase overall network lifetime.

In Figure 13, we analyze the latency of unicast, basic tree and relaxed tree routing algorithms in terms of number of hops a message makes in the physical network from a source peer to a destination peer. Figure 13a shows the cumulative distribution function of peers receiving a message before X hops in the physical network under three routing algorithms. The performance of unicast is slightly better than basic tree routing (i.e., $1/2$ hop less on average). Relaxed tree algorithm produces on average one more hop compared to basic tree algorithm (Figure 13b). It is important to note that we do not restrict the maximum depth of a relaxed tree in our evaluations, however, the maximum depth can be limited according to the VE application requirements.

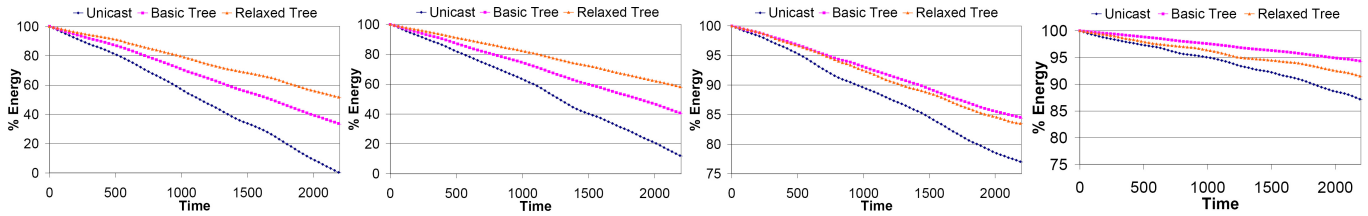
6. CONCLUSION

In this paper, we propose TriCast, a fully-distributed P2P architecture to support VE applications on an ad-hoc network. Triangulation [4] is used to support peer connectivity in the virtual world in order to allow peers to immediately determine the multicast group of a message when necessary. With the physical topology information provided by the physical network, peers generating an event build a multicast tree spanning all peers interested to the event. The message is then forwarded over the physical network following the tree path. We compare the diffusion of a message to its interested group under three routing schemes : unicast, basic tree and relaxed tree. Relaxed tree routing algorithm reduces the cost of message diffusion to the multicast group and thus extends the entire network lifetime.



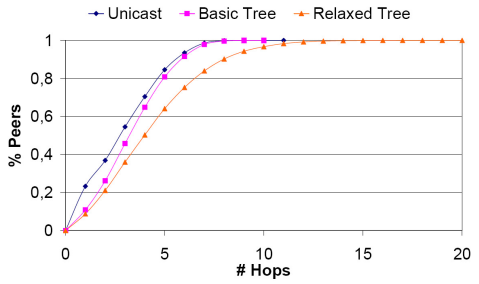
(a) Unicast (b) Basic Tree (c) Relaxed Tree

Fig. 11. Peers' lifetime



(a) Peer 36 (b) Peer 14 (c) Peer 2 (d) Peer 25

Fig. 12. Energy level of peers under three routing algorithms



	Average # Hops
Unicast	3,384
Basic Tree	3,827
Relaxed Tree	4,826

(a) (b)

Fig. 13. Cumulative distribution function of peers receiving a message before X hops in the physical network

7. REFERENCES

[1] A. P. Yu and S. T. Vuong, "MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *Proc. NOSSDAV*, June 2005, pp. 99–104.

[2] J. Keller and G. Simon, "Solipsis: a massively multi-participant virtual world," in *Proc. PDPTA*, 2003, vol. 1, pp. 262–268.

[3] S.Y. Hu, J.F. Chen, and T.H. Chen, "VON: A scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, July/August 2006.

[4] E. Buyukkaya and M. Abdallah, "Efficient triangulation for p2p networked virtual environments," *Multimedia Tools Appl.*, vol. 45, no. 1-3, pp. 291–312, 2009.

[5] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, January 2000.

[6] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, September 1991.

[7] J.R. Jiang, Y.L. Huang, and S.Y. Hu, "Scalable AOI-cast for peer-to-peer networked virtual environments," in *ICDCS Workshops*, 2008, pp. 447–452.

[8] Behnoosh Hariri, Shervin Shirmohammadi, and Mohammad Reza Pakravan, "LOADER: A Location-Aware Distributed Virtual Environment Architecture," in *Proc. VECIMS*, july 2008, pp. 97 –101.

[9] S. Ito, H. Saito, H. Sogawa, and Y. Tobe, "A propagation of virtual space information using a peer-to-peer architecture for massively multiplayer online games," in *ICDCS Workshops*, july 2006, pp. 44 – 44.

[10] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," *Request for Comments 3626*, October 2003.

[11] T. Krag and S. Büettrich, *Wireless Mesh Networking*, O'Reilly Wireless Devcenter, January 2004.

[12] H. Liang, R. N. De Silva, W. T. Ooi, and M. Motani, "Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications," *Multimedia Tools Appl.*, vol. 45, no. 1-3, pp. 163–190, 2009.

[13] L. M. Feeney, "An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mob. Netw. Appl.*, vol. 6, no. 3, pp. 239–249, 2001.