Bandwidth- and Latency-Aware Peer-to-Peer Instant Friendcast for Online Social Networks

Jehn-Ruey Jiang, Chao-Wei Hung, and Jih-Wei Wu
Department of Computer Science and Information Engineering
National Central University,
Jhongli City, Taiwan, R.O.C.
jrjiang@csie.ncu.edu.tw, hzwei@msn.com, jihwei.wu@msa.hinet.net

***Abstract*—Online Social Networks (OSNs) are more and more popular recently; people may through them interact with each other for the purpose of social intercourse. The client/server OSN architecture brings about the bottleneck of bandwidth and computation. It leads to the scalability problem and the communication latency increases as users grows. This paper proposes a bandwidth- and latency-aware peer-to-peer (P2P) instant friendcast scheme for every user (or peer) in OSNs to construct a friendcast tree (FCT) to send instant messages to all of its friends. A lightweight server is responsible for only easy tasks, such as logining and maintaining peer information, to facilitate the tree construction. A peer logins to the server to obtain the list of friends and their Vivaldi coordinates, which are computed by every peer in a distributed way to estimate the latency between peers. The proposed scheme also uses Available Out-Degree Estimation (AODE) to evaluate the proper out-degree of a peer, and then uses Degree-Adapted Greedy Tree Algorithm (DATGA) to construct FCT. The scheme is simulated and compared with other relevant ones to show its advantages.**

Keywords*-friendcast; online social network; peer-to-peer; network coordinate; multicast tree*

I. INTRODUCTION

The rapid evolution of Internet has dramatically changed the communication style of people in recent years. In the Web 2.0 era of Internet, people can communicate, interact or exchange data with each other in real time via diverse types of messages such as text, voice and video. End users participate not only as passive consumers of content provided by web sites, but also as contributors creating content collaboratively with fellow users.

Online Social Networks (OSNs), such as ICQ, MSN Messenger, EtherPad, Facebook, MySpace, Twitter, and Plurk, are an important class of Web 2.0 applications, in which users can share content with others, especially *friends*. OSNs have become extremely popular nowadays. For example, Facebook has more than 500 million active users, which spend over 700 billion minutes per month on it to share more than 30 billion pieces of content (e.g., web links, news stories, blog posts, notes, and photo albums)[1].

The legacy client-server (C/S) architecture may be unscalable and restrain the blooming Internet applications, including OSNs, because of its centralized and limited supply of computation ability, network bandwidth, and data storage. On the other hand, the peer-to-peer (P2P) architecture has every participating entity as a resource provider and consumer; it is thus scalable. Besides scalability, the P2P architecture has the properties of decentralization, self-organization, fault resilience, ad hoc connectivity, and low construction costs. Therefore, many Internet applications and/or services are built on the basis of the P2P architecture.

Below, we focus on the P2P *instant friendcast* operation in OSNs, by which a user (or peer or node)[2] can send a message to all of its online friends in real time under the P2P architecture. Some useful services, such as *instant messaging* and *collaborative real-time editing*, can be built atop the service. Instant friendcast is related to the multicast service. Many P2P approaches [6][7][8][19][17][21] that construct efficient multicast trees for the multicast service may be used to provide the friendcast service. However, these approaches do not fully exert OSN characteristics in the sense that they do not integrate the friend relationships into the tree construction for better performance.

In this paper, we propose an efficient P2P friendcast scheme for OSNs. Note that the scheme can be built on top of C/S OSN systems (e.g., ICQ, Facebook, MySpace, Twitter, and Plurk) or P2P OSN systems (e.g., PeerSoN [17]) as long as the systems can provide each peer with the list of its friend peers and their relevant information like IP addresses and statuses, etc. In the scheme, each peer builds a *friendcast tree* (*FCT*) to cover all and only all of its friends (or *friend peers*) for the purpose of sending messages to them. The scheme has the advantages that a peer only needs to track friends' statuses for maintaining the tree. Furthermore, only friend peers of the message source peer are selected to forward the message. Thus, non-friend peers are exempt from participating the message forwarding and the risks of information eavesdropping are prospectively reduced.

In the proposed scheme, a peer calculates the average message traffic load caused by friend peers and then uses it along with the available bandwidth to evaluate the proper out-degree in the FCT. The scheme is thus bandwidth-aware and the message dropping rate is low. Furthermore, with the help of Vivaldi *network coordinate system* (*NCS*) [6], which can be used to estimate the transmission latency between two peers, the scheme is latency-aware and the transmission latency is short. In practice, the scheme utilizes a greedy algorithm to

[1] http://www.facebook.com/press/info.php?statistics

[2] The words "user" and "peer" and "node" are used interchangeably.

construct the FCT according peers' NCS coordinates and estimated out-degrees. We conduct simulation for the proposed scheme and other related multicast schemes and find that the proposed scheme outperforms others.

The rest of this paper is organized as follows. In Section 2, we review some related work. We elaborate the proposed scheme in Section 3 and show the simulation results of it and its related schemes in Section 4. And finally we conclude the paper with Section 5.

II. RELATED WORK

In this section, we review some related work regarding P2P OSNs, the NCS and P2P multicast schemes.

A. P2P OSNs

Most of popular OSN applications, such as ICQ, MSN Messenger, EtherPad, Facebook, MySpace, Twitter and Plurk are established on legacy C/S architecture. The paper [20] shows that existing centralized OSNs have some non-trivial limitations, such as limited bandwidth and computation resources, and that decentralized OSNs may hence be promising in the future. The P2P architecture is decentralized; the paper [1] advocates using the P2P architecture to implement OSNs so that users can store their data in a P2P manner to keep privacy and can use data even when Internet access is not available. The paper [17] suggests the implementation of a P2P OSN called PeerSon based on the *distributed hash table* (DHT), a well-known efficient structure of P2P networking. Shifting from the C/S to the P2P architecture has advantages; however, it also raises new problems. For example, system coordination becomes difficult when peers join and leave the system frequently.

The paper [19] proposes to use the hybrid architecture to overcome the problems. In such an architecture, servers are deployed in the system to assist in the bootstrapping and the tracking of the system, while other participating entities also assist with running the system in a P2P manner. The system coordination hence becomes easy and the system resources become abundant. In other words, the advantages of the C/S and P2P architectures coexist in the hybrid system. The paper [19] states that hybrid file-sharing systems have better performance than pure P2P systems because some tasks (e.g., searching) can be done much more efficiently in a centralized manner. We believe that OSNs could also benefit from the hybrid architecture.

B. NCS

The NCS assigns synthetic coordinates to Internet peers, so that the Euclidean distance between two peers' coordinates can be used to predict the network latency between them. Through the NCS, topology-aware applications can be established to achieve more responsive system behaviors, such as shorter data transmission latency and faster request response. Two classes of algorithms can be used to generate network coordinates for peers. They are *landmark-based algorithms* and *simulation-based algorithms* [6][13].

Landmark-based algorithms, such as GNP [14] and Lighthouse [15], involve centralized components and require global knowledge of peer latency measurements. Such algorithms use a set of infrastructure hosts called *landmarks* for calculating peers' coordinates. The key idea of GNP [14] is to model Internet as a geometric space. Some hosts specified as landmarks compute their own coordinates first and altogether serve as a frame of reference for any participating peers. Peers contact multiple landmarks to calculate their coordinates relative to those of landmarks. GNP is an absolute coordinate system; the accuracy of peers' coordinates depends on the choice of landmarks. Besides, landmarks are apt to become bottlenecks of networks and incur the single-point-of-failure problem [21]. Lighthouse [15] tries to eliminate the bottlenecks and the single-points-of-failure problem, and intends to be more scalable. It uses multiple sets of local *bases* with their own relative coordinate systems. Joining peers may select any of these local bases to contact with and determine their local coordinates. These local coordinates are then mapped into a global coordinates. Lighthouse is more scalable and accurate than GNP due to its localized coordinate design.

In simulation-based algorithms, every peer calculates its coordinate according to the network measurements between itself and its communication peers. Vivaldi [6] models peers as entities in a spring system. It determines peers' coordinates using spring relaxation simulation. The prediction error of the network distance measurement between a pair of peers is treated as the potential energy stored in a spring which connects these two peers. Peers attract and repel each other according to network distance measurements. In the other words, peers tune their coordinates to minimize the prediction error. The low-energy state of the spring system corresponds to the coordinates with the minimum error [6]. Vivaldi is widely used since it is fully distributed. Based on Vivaldi, Pharos [3] is fully decentralized and hierarchical. It seeks to eliminate the phenomenon that long distance scales of peers largely affect the prediction accuracy. In Pharos, each peer holds two sets of coordinate systems for long distance and short distance, respectively. According to the distance scale of interest, peers can choose the proper set of coordinate systems to achieve higher prediction accuracy.

C. Multicast Algorithms

Paul and Raphavan indicate that "multicast" is an efficient scheme suitable to one-to-many or many-to-many communications paradigm [16]. They make a survey of multicast algorithms working in the network layer, whose overall features and constraints are quite different from those of P2P algorithms working in the application layer. However, the analysis of topologies and efficiency are still valuable for reference. In practice, most multicast algorithms rely on constructing multicast trees to improve transmission efficiency, since the trees minimize duplication of forwarding messages.

Three tree construction algorithms, namely MST, Modified ESM [5], and LGK [2], are used to examine the performance of the NCS in [3] and [22]. The MST (Minimum Spanning Tree)

algorithm is based on classical Prim's algorithm to construct the minimum spanning tree by greedily selecting the shortest links measured by NCS coordinates. The modified ESM (End System Multicast) algorithm is a variant of the broadcast ESM protocol. A new peer first obtains a randomly sampled partial list of on-tree nodes, and then selects the one with the smallest latency as its parent. In the simplified version of modified ESM, the link capacity or node degrees are ignored. The LGK (Location-Guided k-ary) algorithm constructs a k-ary tree by exploring node location information on a plane. An LGK tree is constructed as follows: (1) the root node selects the closest k nodes as its child nodes; (2) the remaining nodes are clustered to the k child node according to geometric proximity. As a result, each of the k child nodes is the root of a subtree consisting of the nodes closer to it than to other child nodes. The multicast tree is formed as each subtree repeats the two steps of child selection and clustering. It has been shown that k =2 gives the best tradeoff between the delivery delay and overhead of the node.

VoroCast [11] is a special multicast scheme derived from the VON project [9] for P2P *networked virtual environments* (*NVEs*), in which a peer assumes a virtual representative called *avatar* to interact with other avatars in a computer generated virtual scene through network connections. It is usual that an avatar would like to send a message to all AOI neighbors, the avatars within the *area of interest (AOI)*, which is the circle of a pre-specified radius centered at the avatar. The basic idea of VoroCast is to construct a multicast tree spanning all AOI neighbors for each node. Messages can then be sent along the tree edges without redundancy. Each node organizes its neighbors with a Voronoi diagram and separates them and itself into various Voronoi regions. In the child selecting procedure, the message sender assumes itself as the root node. And the *enclosing neighbors* of the root, whose regions are adjacent to the root's, select the root as their parent. The other nodes just select their enclosing neighbors that are nearest to the root as their parents. In this way, the multicast tree is constructed.

There exist many algorithms [7][18] for solving the NP-hard constrained spanning tree problems, in which the degrees of tree nodes are limited. For example, the heuristic genetic algorithm (HGA) [7] combines the heuristic search methods with genetic algorithms to construct degree-constrained MST by using the chromosome as heuristic information for searching globally. Degree-constrained MST can shed light on instant friendcast schemes. However, due to the high complexity, heavy overheads and the fixed-node-degree constraints, such algorithms are not suitable for the P2P instant friendcast operation in which timeliness is important and node degree constraints are changing dynamically.

III. The Proposed Scheme

This section elaborates the details of the proposed P2P instant friendcast scheme applied to OSNs, which is notably aware of network bandwidth and communication latency. The scheme is for a node to construct a FCT to cover all of the node's friends for forwarding messages efficiently. It uses Available Out-Degree Estimation (AODE) to estimate the proper out-degree of a tree node by examining the node's available bandwidth and the average traffic caused by the node's friends. It then applies Degree-Adapted Greedy Tree Algorithm (DAGTA) to select parent nodes for a node in FCT according to the node's estimated out-degree and all nodes' Vivaldi NCS coordinates. Below, we first describe the system architecture of the scheme and then describe the details of AODE and DAGTA.

A. System Architecture

As we have mentioned, the proposed scheme can be integrated into C/S OSNs (e.g., ICQ, Facebook, MySpace, Twitter, and Plurk) or P2P OSNs (e.g., PeerSoN [17]), as long as the OSNs can provide each peer with the list of its friend peers and their relevant information like IP addresses, communication statuses, and NCS coordinates. For the sake of simplicity, we assume there is a lightweight server to take the housekeeping tasks. It is noted that the server is actually of light load since it is only responsible of logining, providing friend peer information and managing peer statuses.

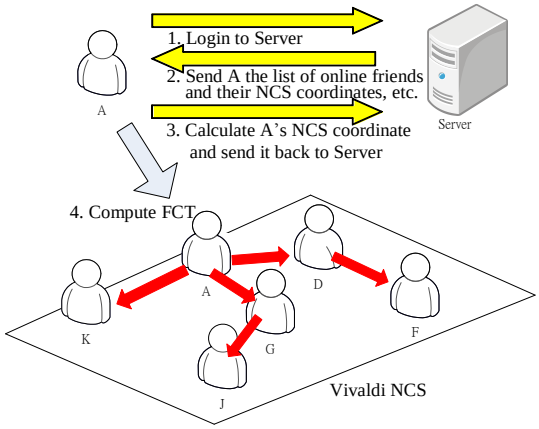

Fig 1. System architecture of the proposed scheme

As shown in Fig. 1, when a peer joins the system, it first logins to the server to get its ID and the list of online friend peers along with their IDs, IP addresses and NCS coordinates. If the peer has not got a NCS coordinate yet, it computes its own Vivaldi NCS coordinate and sends it back to the server. Afterwards, the peer constructs the FCT covering all friend peers according to the information obtained to achieve the goal of friendcast: to send a message to all online friend peers in a timely manner.

B. FCT Construction

In the proposed scheme, the FCT plays the role of delivering messages from a peer to all its friend peers with as short as possible latency. Below, we define the total transmission latency, TL, as the measurements of a friendcast scheme. Assume peer n_0 has m friend peers $n_1,\ldots,n_m$ and let l_i , $1\le i\le m$, be the latency that peer n_0 transmits a message to friend peer n_i. We have

$$\mathrm{TL} = \sum_{1 \le i \le m} l_i$$

TL is the sum of the latency that n_0 transmits a message to all of its friend peers. The goal of the proposed scheme is to keep TL as small as possible. The scheme first maps friend peers into Vivaldi NCS so that the transmission latency of any two peers can be estimated by the distance of their coordinates. It then tries to construct a FCT to span all friend peers such that the summation of the accumulated distances from all friend peers' coordinates to n_0's coordinate is minimized. At the first glance, if n_0 sends the message to all friend peers directly, TL is minimum due to the triangle inequality of NCS which is based on to estimate the latency between peers. However, when the bandwidth of n_0 is very limited and the message consumes a lot of bandwidth (e.g., messages for instant voice or video messaging), the directly sending approach will use up the peer's bandwidth and is thus not feasible. We therefore should take the available outgoing bandwidth of peers into consideration when constructing the FCT.

In the proposed scheme, Available Out-Degree Estimation (AODE) is used to evaluate the proper out-degree of each node in the FCT. AODE of a peer n_i is denoted as $AODE_i$, which is defined as follows.

$$AODE_i = \begin{cases} \left\lfloor \frac{C_i}{p_i \times f_i \times S} \right\rfloor, & \text{if } C_i \ge (p_i \times f_i \times S) \\ 1, & \text{if } C_i < (p_i \times f_i \times S) \end{cases} \quad (1)$$

In Eq. (1), S is the size of the message, C_i is the outgoing bandwidth of n_i, f_i is the current number of friend peers of n_i, and p_i is the estimated probability that n_i is asked by its friend peers to forward messages (refer to Eq. 2 for p_i). Hence $p_i \times f_i$ denotes the estimated number of forwarding requests that n_i will receive from its friend peers in the meanwhile. Therefore, $p_i \times f_i \times S$ means the current estimated traffic load shared by n_i. The out-degree of n_i is set to 1 if C_i is less than $p_i \times f_i \times S$; otherwise, it is set to $\left\lfloor \frac{C_i}{p_i \times f_i \times S} \right\rfloor$.

Eq. (2) shows how to derive the estimated probability p_i of peer n_i to be asked by its friend peers to forward messages.

$$p_i = \frac{R_i}{F_i} \quad (2)$$

In Eq. (2), R_i is the accumulated number of forwarding requests that n_i receives from its friend peers, and F_i is the accumulated number of friend peers during the last specified estimation period. It is noted that R_i is increased by 1 and F_i is increased by the number of friend peers of n_i every time when peer n_i receives a request during the last specified estimation period. The value of p_i is the ratio of R_i to F_i.

The proposed scheme uses the greedy algorithm DAGTA to construct FCT. Since a FCT with a small height is preferred, the peer with the highest AODE is first selected to be the child node of the root. If there are many nodes with the same AODE, then the one with the shortest Euclidean distance in NCS is selected.

We show the pseudo code of DAGTA in Fig2. Given the friendcast source peer (node) n_0 and its m friend peers $n_1,\ldots,n_m$, the algorithm can generate the FCT rooted at n_0 to span $n_1,\ldots,n_m$. Without loss of generality, we suppose that $n_1,\ldots,n_m$ are listed in the order of their AODE values. In the algorithm, OD_i keeps the number of child peers of n_i; l_i stores the current accumulated latency that n_0 transmits a message to n_i directly or indirectly; $d_{k,i}$ stands for the latency measured by the distance of NCS coordinates of peers n_k and n_i. The algorithm selects n_k which satisfies $OD_k < AODE_k$ and has the minimum $l_k+d_{k,i}$ for $0 \le k \le i-1$ as the parent node of n_i in the FCT. However, if $OD_k \ge AODE_k$ holds for $0 \le k \le i-1$, the algorithm randomly selects a peer from $n_0, n_1, \ldots, n_{i-1}$ as the parent node of n_i.

Algorithm DAGTA
Input: node n_0 and its m friend peers $n_1,\ldots,n_m$ sorted according to their AODEs
Output: FCT rooted at n_0
$l_0=0$; //the latency from n_0 to n_0 is 0
$OD_i=0$ for $0 \le i \le m$; //OD_i stores the current out-degree of peer i
FOR $i = 1 \,..\, m$ // for peer n_i to select its parent node
  $l_i=\infty$; //l_i records the latency from n_0 to n_i, which is initially ∞
  p_i=null; // p_i is the parent node of n_i,
  FOR $k = 0 \,..\, i\text{-}1$ // parent node selection from $n_0, n_1, n_2 \ldots, n_{i-1}$
    IF $OD_k < AODE_k$ **THEN**
      IF $l_i > l_k + d_{i,k}$ **THEN**
        $l_i = l_k + d_{i,k}$
        $p_i = n_k$
  IF p_i = = null **THEN** // if no nodes satisfy $OD_k < AODE_k$
    Select an arbitrary peer from $\{n_0, n_1, \ldots, n_{i-1}\}$ and assign it to p_i
RETURN FCT rooted at n_0 and spanning $n_1,\ldots,n_m$ with parent pointers $p_1,\ldots,p_m$

Fig 2. The pseudo code of DAGTA

A parent selection example of DAGTA is illustrated in Figure 3, where five nodes $n_0,\ldots,n_4$ are depicted and the parameters of a peer n_i, $0 \le i \le 4$, are represented as a 4-tuple: ($AODE_i$, $d_{0,i}$, OD_i, l_i) after n_1, n_2 and n_3 have selected their parent nodes. Peer n_4 will takes n_1 as its parent peer since n_1, n_2 and n_3 all meet the requirement of $OD_i < AODE_i$ ($1 \le i \le 3$), and $l_1+d_{1,4}$ is the smallest among $l_1+d_{1,4}$, $l_2+d_{2,4}$ and $l_3+d_{3,4}$.

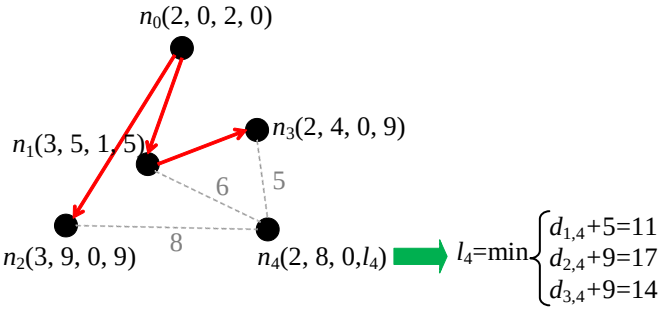

Fig 3. An example of DAGTA for the parent node selection

IV. Performance Evaluation

A. Simulation settings

We evaluate the performance of the proposed friendcast scheme by simulation experiments using MIT King [8] data set as the reference of network latency between arbitrary peers.

The data set contains measurements of the latencies between a set of 1,740 DNS servers collected using the King method [8].

The P2P network is assumed to have 300 peers, each of which is assumed to have 20, 30, or 40 friend peers. We use a freeware WireShark [10] to monitor the packets traffic, and take the Vivaldi NCS. Within a P2P network, the *churn rate* refers to the proportion of peers leaving the system during a given period (100 seconds in this evaluation). The churn rate is set as 0 and 20%. The parameters c_c and c_e are for Vivaldi NCS. They are set to 0.25 as suggested in [3]; the former is the constant fraction of the estimation error, and the latter is the local estimation error between the practical latency and the latency estimated by Vivaldi NCS. Each experiment is run for 1000 steps and each peer is assumed to generate 2 messages per 10 seconds. Each message is assumed to have 30 ms of processing delay when passing through one intermediate forwarding peer. Table 1 shows all the simulation parameters. We also assume peers have the upload bandwidth distribution proposed in [2], which is shown in Table 2.

Table 1. Simulation parameters

Network Size	300 peers
Simulation Steps	1000
Average Number of Friends (ANF)	20, 30, or 40
Churn Rate	0% or 20%
Message Load	2 /10s
c_c and c_e	0.25
Message Size (MS)	1500 bytes
Buffer Size	ANF*MS (bytes)
Processing Delay	30 ms

Table 2. Upload bandwidth distribution of peers.

Uplink (KB/sec)	Fraction of peers
10	0.05
30	0.45
100	0.40
625	0.10

We compare the performance of our friendcast scheme, which employs NCS and multicast tree technologies, with those of degree-constrained Prim's MST **(**DCPrim**)**, modified ESM (mESM), VoroCast, Dijkstra one-to-all shortest path, and LGK algorithms. DCPrim uses Prim's algorithm to construct the multicast tree and uses AODE to limit the out-degree of every peer. We set k=2 and k=15 for LGK to cover both the low out-degree and high out-degree situations. For Dijkstra's algorithm, we use the real latencies measured by "ping" as the distance between nodes since latency estimated by the NCS coordinates satisfies the triangle inequality and thus is not suitable for Dijkstra algorithm. A basic scheme, called STAR, in which a peer directly sends messages to all of its friend peers is also evaluated for the sake of comparison.

B. Simulation Results

Average latency and average reachability are the two performance metrics about which are concerned mostly. They are defined as follows.

$$\textit{Average latency} = \frac{\text{The total latency of friend peers receiving messages}}{\text{The number of friend peers receiving messages}}$$

$$\textit{Average reachability} = \frac{\text{The number of friend peers receiving messages}}{\text{The number of friend peers}}$$

The simulation experiments measure the two metrics under peer uplink bandwidth limitation, as defined in Fig. 2. Below, we show the simulation results under the churn rates of 0% and 20%.

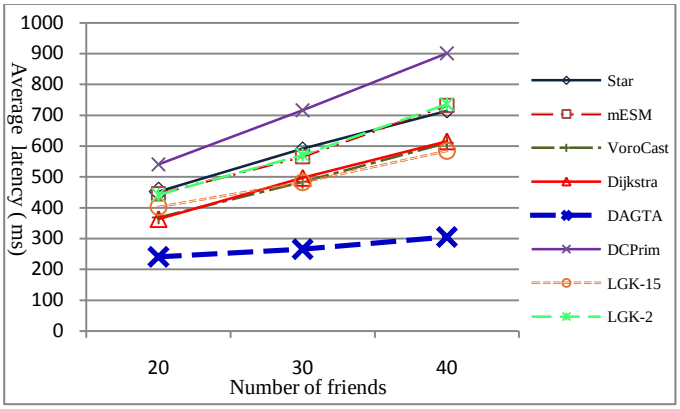

Fig 4. Average latency for churn rate=0%

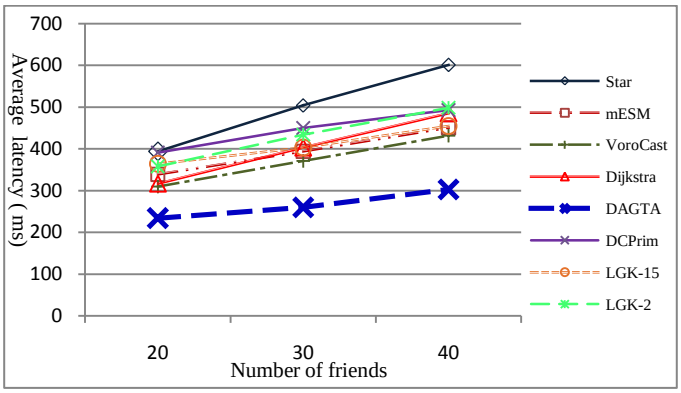

Fig 5. Average latency for churn rate=20%

As shown in Figures 4 to 7, for the churn rates of 0% and 20%, DAGTA outperforms others in terms of the average latency and average reachability. This is because DAGTA tries to keep FCT as low as possible under the consideration of setting proper peer out-degrees by AODE. If peers' out-degrees are not set properly, the outgoing bandwidth may be exhausted and the messages waiting to transmit are stored in the message buffer temporarily, which lengthens the latency. In case buffer is full and some incoming messages arrive in the mean while, the peer must drop some messages. This also counts for the reason why some algorithms with low tree heights (e.g., STAR and Dijkstra's algorithms) have worse performance than DAGTA. When churn rates go up, the leaving peers prevent messages from reaching some peers deeper in the tree. This leads to lower reachability and accounts for the reason why some algorithms with high tree heights (e.g., DCPrim, modified ESM and LGK-2) have worse performance than DAGTA. It is worth mentioning that the performance degrades with the number of friends for all algorithms, since more friends lead to more loads.

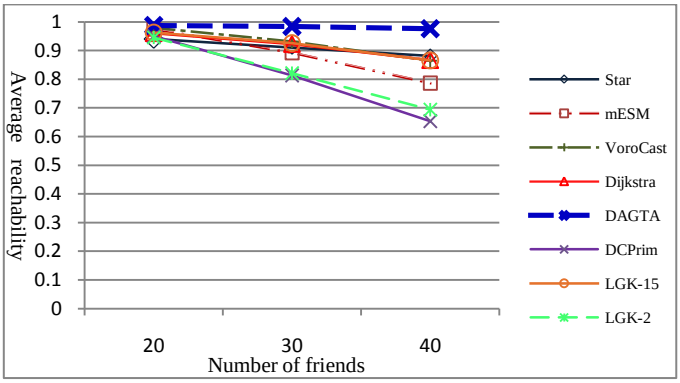

Fig 6. Average reachability for churn rate=0%

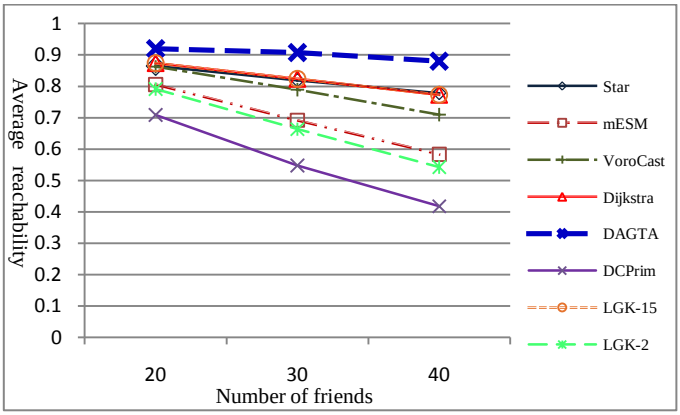

Fig 7. Average reachability for churn rate=20%

V. CONCLUSION

This paper proposes a new bandwidth- and latency-aware P2P instant friendcast scheme for OSNs. This scheme constructs an efficient FCT with a greedy DAGTA algorithm on the basis of Vivaldi NCS coordinates and AODE of out-degree estimation. All in all, the scheme takes the physical network topology and the capacity of the underlying network into account for the purpose of shortening the latency and increasing the reachability of message delivery. As shown by the simulation results, the proposed scheme outperforms other related schemes in terms of the latency and reachability.

This paper focuses on the bandwidth and latency aspects of the instant friendcast scheme. We plan to discuss the consistency and fault-tolerance issues about the scheme in the future to deal with peers frequently leaving and joining the system. Furthermore, we have found that the proposed scheme is also suitable for bandwidth-hungry and time-constrained P2P applications, such as P2P audio streaming, video streaming and 3D streaming [4]. In practice, we have been planning to apply the proposed scheme to the above-mentioned applications.

REFERENCES

[1] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks – Opportunities and Challenges," Proc. 6th International Conference on Wireless On-demand Network Systems and Services (WONS 2009), Snowbird, Utah, USA, February 2009.

[2] K. Chen, K. Nahrstedt, "Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET," Proc. INFOCOM 2002, 2002.

[3] Y. Chen, Y. Xiong, X. Shi, J. Zhu, B. Deng, X. Li, "Pharos: Accurate and Decentralized Network Coordinate System," IET Comm., Vol. 3, Iss. 4, pp. 539-548, 2009.

[4] C. H. Chien, S. Y. Hu, J. R. Jiang, "Bandwidth-Aware Peer-to-Peer 3D Streaming," Proc. 8th ACM NetGames Workshop, Paris, 2009. (also International Journal of Advanced Media and Communication, to appear)

[5] Y. H. Chu, A. Ganjam, T. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early Experience with an Internet Broadcast System Based on Overlay Multicast," Proc. of USENIX, June-July 2004.

[6] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," Proc. ACM SIGCOMM, August 2004.

[7] Y. Feng, Z. Yu, Y. Pan, "Heuristic Genetic Algorithm for Degree Constrained Multicast Routing Problem," Proc. 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, pp. 2448-2452, 2004.

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," Proc. of SIGCOMM IMW 2002, pp. 5-18, November 2002.

[9] 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, Jul.-Aug. 2006.

[10] http://www.wireshark.org/

[11] J. R. Jiang, Y. L. Huang, and S. Y. Hu, "Scalable AOI-Cast for Peer-to-Peer Networked Virtual Environments," Proc. IEEE ICDCS Workshop on Cooperative Distributed Systems (CDS), Jun. 2008.

[12] S. Keshav, S. Paul, "Centralized Multicast," Computer Science Department, Cornell University, USA, Technical Report TR98-1688, 1998.

[13] J. Ledlie, P. Gardner, and M. Seltzer, "Network Coordinates in the Wild," Proc. NSDI, April 2007.

[14]T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-based Approaches," Proc. INFOCOM, June 2002.

[15] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for Scalable Distributed Location," Proc. IPTPS, February 2003.

[16] P. Paul and S. V. Raghavan, "Survey of Multicast Routing Algorithms and Protocols," Proc. 5th International Conference on Computer Communication (ICCC 2002), pp. 90-102, 2002.

[17] D. Schi¨oberg, "A Peer-to-Peer Infrastructure for Social Networks," Diploma thesis, TU Berlin, Berlin, Germany, December 17, 2008.

[18] A. Tawfig, and Z. Taieb, "Delay-Constrained, Low-Cost Multicast Routing in Multimedia Networks," Journal of Parallel and Distributed Computing, Vol. 61, No. 9, pp. 1307-1336, 2001.

[19]Beverly Yang and Hector Garcia-Molina, "Comparing Hybrid Peer-to-Peer Systems," Proc. 27th International Conference on Very Large Data Bases, pp. 561-570, 2001.

[20] Ching-man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee, "Decentralization: The Future of Online Social Networking," Proc. Workshop on the Future of Social Networking, Barcelona, Jan 2009.

[21] R. Zhang, Y. C. Hu, X. Lin, and S. Fahmy, "A Hierarchical Approach to Internet Distance Prediction," Proc. IEEE ICDCS, July 2006.

[22] R. Zhang, C. Tang, Y. C. Hu, S. Fahmy, and X. Lin, "Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications – an Analytical and Comparative Study," Proc. IEEE INFOCOM, May 2006.