TCMM: Hybrid Overlay Strategy for P2P Live Streaming Services*

Hai Jin, Xuping Tu, Chao Zhang, Ke Liu, and Xiaofei Liao

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
Huazhong University of Science and Technology, Wuhan, 430074 China
hjin@hust.edu.cn

Abstract. This paper proposes an application level multicast approach called *Tree-Control-Mesh-Media* (TCMM) to distribute live media streams to a large number of users efficiently. In TCMM, transmissions of media data are controlled by two independent relay protocols in a collaborative manner. One protocol here is used to help a peer to identify its neighbor peers using the location information while the other one is used to deliver of media stream among the peers. The two protocols organize all peers into two graphs with different topologies that the communications can benefit a lot from the hybrid control topology. We have studied the performance of TCMM approach using different simulation cases. The experimental results have shown that the broadcasting performance of TCMM can achieve that of a well constructed mesh network while it can adapt more dynamic and irregular network environment. We also see that the penalty of introducing two protocols is rarely low which implies the high scalability of TCMM.

1 Introduction

Recent research works reveal the brilliant future to provide media streaming services based on the P2P substrates. Many papers discuss the important roles that peer nodes have played in distributing streaming media. Till now, many P2P media streaming systems have been developed. They can be divided into three catalogues: tree-based (or hierarchical-based) system [20], DHT-based system [22] and mesh-based system [6]. In tree-based system, all peer nodes are organized as a spanning tree over the existing IP network, and the streaming data are distributed along that tree. As the parent nodes should provide streams to child nodes, the total bandwidth of a parent node having n child nodes would be $bw \times (n+1)$, where the bw is the minimum bandwidth needed by a peer. One disadvantage of the distribution topology is that a parent node will require more in bandwidth to feed its child nodes. Also, this kind of systems which only have one root node will become unstable when peers join and leave frequently [19].

* This paper is supported by National Science Foundation of China under grant 60433040, and CNGI projects under grant CNGI-04-12-2A and CNGI-04-12-1D.

The second distribution system is DHT-based. In this kind of systems, peers are organized as a circle. Due to the ring-alike topology, one peer node just has to bypass the stream to its neighbor peer. However, it also suffers for the instability and usually lacks methods to optimize the communication. The systems belong to the third catalog are mesh-based. In these structures, every peer node provides data to and gets data from several other nodes. Although this kind of structures have no stability problem, it is also very difficult to do traffic optimization [14][26].

In this paper, we propose a hybrid communication scheme, *Tree-Control-Mesh-Media* (TCMM). We organize all peers into two graphs, one is the spanning tree and the other one is a pure mesh. In the spanning tree, only control messages can be transmitted, therefore all the peers can quickly find its neighbor peers and establish data links using the control messages. Then all the media data can be transmitted in a constructed mesh network as traditional mesh-based systems. Extensive simulations demonstrate that this kind of hybrid structure gives a better solution for the locality optimization and stability. Usually, in a non mesh-based system, it is critical to avoid high quantity of messages transmitted from the parent node to each child node. However, in TCMM, nodes can receive control data from different peers simultaneously, which can reduce the risk of suffering from a high transmission rate.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents TCMM scheme. Performance evaluation of the TCMM is presented in section 4. Finally, we conclude our work in section 5.

2 Related Works

Based on different network topology, application level multicast used in P2P media streaming systems can be divided into three categories: DHT-based, tree-based, and mesh-based.

The systems belonging to the first kind rely on those existing DHT network to optimize the paths according to certain metrics such as latency and bandwidth. For example, paper [18] is based on *content addressable network* (CAN) [17], and Bayeux [27] is based on Tapestry [4]. CoopNet [22] supports both live and on demand streaming. It employs *multi-description coding* (MDC) to construct multiple distribution trees (one tree for one strip). SplitStream [15] is based on Scribe [3] which is based on Pastry [2].

In tree-based systems (Yoid [9], ALMI [16], Nearcast [25], NICE [20], ZIGZAG [21], Anysee [11], and Chunkyspread [23]), peers are organized into a hierarchal structure. They just get streams from a single parent. The advantages of these systems include low overhead and can get optimal nearby nodes as data provider. However some peers usually have not enough bandwidth to support their children and it is difficult to resist the churn. Hence it limits the deployment of tree-based systems.

The mesh-based systems are named for the reason that each peer has multiple data senders and receivers, e.g., Narada [6], ScatterCast [5], PROMISE [13], DagStream [10], RandPeer [12]. They overcome the difficulties in tree but lead to redundant traffic of underlying physical networks.

CoolStreaming [24] is one of the most famous mesh-based application level multicast systems. By using DONet protocol, each node first exchanges data available information with all the partners periodically, and then retrieves the data from one or more partners. Actually, the data transferring mesh in our proposed approach is similar to CoolStreaming to inherit the efficiency of data exchanging.

BULLET [7] is the most similar structure to TCMM. It uses RanSub [8] to build an overlay tree, one peer, if not fed enough, can receive data from multiple ancestors in the tree. But the tree participating the data transferring is different from TCMM. Since in TCMM, the tree is to organize the peers in a locality-aware overlay. The mesh overlay is used to exchange media data.

Different from these systems, TCMM is proposed for the streaming system that each receiver should have multiple senders. Here the tree topology is just used to identify nearby senders.

3 Design of TCMM

The main focus of this paper is the design and implementation of TCMM which is based on our previous work Nearcast [25]. First we will give a brief introduction of TCMM, and then details of the TCMM approach will be introduced.

3.1 Overview of TCMM

All peers in TCMM are involved to distribute media data. They are organized into two overlays – one is used as control tree and the other is used as media mesh. The control tree structure is used to make all nodes in the tree close to each other physically, it means there must be few routers between each pair node, or the *Round Trip Time* (RTT) should be small. Also, the messages transmitted over the tree should be lightweighted messages such as ping/pong messages. Because the out-degree of each node in the tree graph can be very large while the tree height (log$N$) is relatively low. Further, when no media data transmitted, the tree can be loosely maintained, that is, even if some peers have left, other peers still can postpone to update the tree information without breaking transmitting media data in a long period.

The second overlay in TCMM is a data mesh which is similar to CoolStreaming. It is used to transmit media data. Each peer first registers to the network to get a *Global Unique Identity* (GUID). On the other hand, at the beginning, it is at the tree root, the scheduling algorithm then guides it to route to a peer which has a relatively similar GUID. In the routing path, this peer can collect information about the visited peers to build its own candidate partner list. After that, it selects a group of nodes to connect to for more partner information. Finally, it can start the media data exchanging. Fig.1 gives an overview of the two-layer structure of TCMM.

3.2 Tree Management

The tree management is based on Nearcast protocol [25]. In this protocol, leaf peers in the overlay multicast tree are self-organized to form the H layer hierarchical structures.

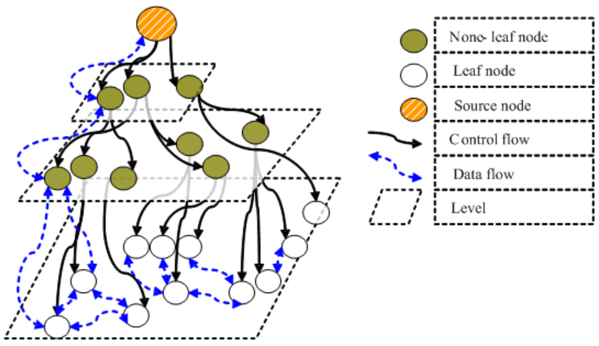

Fig. 1. Overview of TCMM, the dashed line stands for a data link, those of which construct a mesh, while thick line stands for control link for constructing a loosely maintained tree

Based on the network position coordinates of leaf peers, the intra-subtree structure is designed to be sensitive to the locality information. This strategy leads to that nearby leaf peers in the physical network are nearby with each other in the overlay. These two techniques help the overlay multicast tree to become a good represent of the underlying physical network, therefore the link stress and the total (or average) end-to-end delay can be effectively reduced.

The TCMM tree is constructed based on GUID, which consists of the peers' location information. It encodes the following information into 16-bytes of string: network type (firewall or NAT or else), ISP (internet service provider), city, postcode, public IP, and private IP, see Fig.2. Here we introduce briefly only the basic operation of the tree maintenance: Join Process and Leave Process. For more details of how to maintain the tree, readers can refer to Nearcast [25].

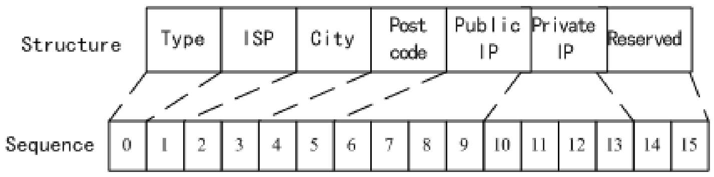

Fig. 2. The elements of GUID

Once an existing host *Y* receives the *"Join*" message from *X*, it uses the admission algorithm to compare the joiner's GUID with its own GUID, so as its children's GUIDs. Also, it tests the network bandwidth constraints to determine whether *Y* is the nearest host to *X*. If so, *X* should be admitted to be a child of *Y*. Otherwise, it is redirected to the nearest child of *Y*. This process will repeat until *X* finds its nearest parent. If a child receives the "*Leave*" message from a leaving peer, it should immediately response by sending a "*Join*" message to its original grandparent. The parent receives "*Join*" message, it will treat it as a new join process. Since in TCMM, the control tree only helps to find close peers without transmitting media data. It is unnecessary to absolutely maintain the tree structure.

3.3 Mesh Management

In TCMM, each peer maintains an *active partners* set and an *inactive partners* set. The *active partners* set is used to exchange media data while the *inactive partners* set is used to select active candidates. A peer also maintains a local window, which stores media data received from others and will be shared with others.

In this section, we mainly focus on partner management and window management techniques. As we know, in real internet environment, peers usually have different bandwidth as well as other network resources. Also, there always exist many partners which receive much more media data than they contribute. Based on this observation, we can classify active partners into two kinds, *provider partner* and *receiver partner*. Suppose node *A* has a partner *B*, whose sequence number of its window's first packet is bigger than that of *A* (usually close to the media source), Here, *B* is the provider partner of *A*, and *A* is the receiver partner of *B*. It is clear that each peer must maintain a minimum number of provider partners in order to maintain continuity. The classification of partners is illustrated in Fig.3. Fig.4 depicts the operations and algorithms applied between a peer and its partners, including a) how to produce inactive partners, b) how to select one from an inactive partners list to be an active partner, and c) how to schedule when more than one active partner possess the data to a peer.

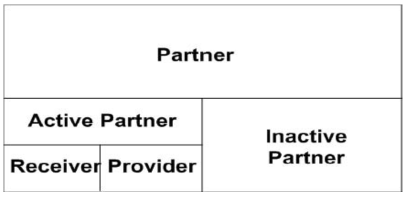

Fig. 3. Classification of partner

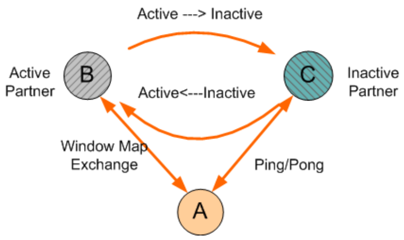

Fig. 4. Partner maintenance (origin node A), Ping/Pong with inactive partners and window map exchange with active partners

3.3.1 Inactive Partner Generation

There are three ways for a peer to get inactive partners to build up its inactive peer list: a) when a peer joins the overlay, it will receive a partner list as a piggyback message of "*OK Response*" message from its father node; b) send requests to its active partners when the peer's count of provider partner is less than a predefined minimum value. When a peer receives a *"Partner Request*" message, it responds by sending an active partner list to the requester. It is because a partner's active partners would proverbially to be active partners. On receiving the partners reply, if they do not exist in the inactive partners list, the peer will add them to the list to be candidates of active partners; c) a peer will periodically collect children and father information in the control tree to build local partners list. Because the tree is maintained by *"Alive"* message, the peers in the tree are very probable online and can perform data transmission well. Thus, each peer will periodically send ping message to those inactive partners to check whether they are still online. Suppose the number of members in inactive list is $N_{inactive}$, ping interval is $I_{inactive}$, packet size of Ping/Pong is $S_{inactive}$ the Ping/Pong overhead is $O_{inactive} = N_{inactive} \times I_{inactive} \times S_{inactive}$.

3.3.2 Active Partner Generation

All active partners are inactive partners before they change their state, therefore, a peer will prepare to select some inactive partners to become active partner candidates when the number of local provider partner is less than a given threshold. Several factors are considered, including: a) the difference of GUID is lower; b) the RTT between is lower; c) more data that it needs is in the window. After choosing several candidates, the peers send "*Identity Request*" message to them. On the other hand, once the peer receives an *"Identity Request"* message, it will check whether this partner can be accepted. If it is ok, then "*Identity Agree Response*" will be sent. Otherwise a reject message will appear as a response. After that they begin to exchange window map at a given interval. At the same time, another task will compare their window maps independently and periodically. Also, a "*Data Request*" request will be sent for the missing data. As the window sliding and the window map changing, the data producing and consuming process continue until the end of the live streaming program. If being rejected, a peer will try the second peer in the candidate list and if accepted, the remote peer will become its *active partner* and be added into the active partner set.

3.3.3 Active Partner Schedule

Before discussing partner selection algorithm, some concepts about windows should be introduced. Each peer maintains a sliding window to store data availability information, including the sequence number of the first segment it is sliding to and the segment states in bytes. In these bytes, each bit stands for a segment's state, 1 is for available, 0 for unavailable. Because each peer's local window is limited, it has to discard the old data and fill new data.

A peer will periodically check its window to request the missing segment by sending a "*Data Request*" message to it. If multiple partners have the unavailable segment, it will schedule which partner acts as the provider. Here, we give a principal to the scheduler scheme, 1) MAX_REQ, which limits the maximum segment one *"Data Request"* message can convey. 2) Every segment of data will have a transmitting pending time $T_{pending}$, if a partner's last transaction has not been completed and does not encounter a timeout error during the transaction time, it should be added to current task this time. 3) If two video segments are available simultaneously, the one with bigger sequence number should have higher priority. This means, we always request the video segment with higher sequence number than that with lower sequence number. The third principal can strengthen the "enlarge ability" of the system. Having the three principals in mind, we implement our own algorithm in Fig.5.

```
Input:
Band[k]: bandwidth from partner k;
wm[k]: window map of partner k;
task[k]: assigned task of k ;
pending[k]: not completed task of k;
num_partners: number of partners of the node;
local_window[i]: segment i of local window map is
available or not;
Scheduling:
for segment i =size(local_window) do
   i i-1;
   if local_window[i]=1 then
     continue;//if segment i is available,schedule next
   end if
   for j to num_partners do
      n n +wm[j,i];//get potential suppliers for i;
   end for j;
   if  n =1 then
   k arg_r{ wm[r,i]=1};// only one potential supplier;
     if task[k]+pending[k]>MAX_REQ then
        continue;
     end if
     supplier[i] k; task[k] task[k]+1;
     continue;
   end if;
   for j =2 to n
     if task[k]+pending[k]> MAX_REQ or
task[k]+pending[k]>band[k] then
        continue;
     end if
     supplier[i] j; task[k] task[k]+1;
   end for j;
end for i;
Output: supplier[i]: supplier for unavailable segment i
```

Fig. 5. Scheduling algorithm at a TCMM node

4 Performance Evaluation

4.1 Simulation Setup

To evaluate performance of TCMM, we first propose a GUID-based delay and bandwidth simulation method instead of using traditional physical topology generation tools to generate physical topology, such as GT-ITM [1]. Because the communication between each pair of nodes is affected by delay and bandwidth, thus, if we try to simulate the two characteristics in internet, we need not generate the physical topology. In our simulation platform, we just generate a peer sets.

We suppose that the delay and bandwidth between two peers can be determined by their GUIDs. In the sending queue, a packet can be sent when the previous sending operation has been finished. The communication delay between two logical neighbors is calculated according to formula 1. From formula 1, we can see that the delay will affect the bandwidth. Also, using GUID-based methods, we generate 5 physical peer sets each with 2000 nodes. The logical topologies are generated with a number of peers (nodes) ranging from 100 to 1,024. Suppose *N* is the number of the total peers, *N*/10 cities and *N*/5 postcodes are generated and randomly assign all the nodes to them. The expected number of inactive partner is 20, and the minimum number of each peer's provider partners is 3, the maximum number of active partner numbers is 15. We start the broadcaster and let 2 randomly selected peers join the system every second. The lifetime of each peer is set to 600 seconds. We collect the log to analyze the performance of our TCMM system.

$$\mathrm{Delay(i,j)}=\mathrm{ISP}_i \oplus \mathrm{ISP}_j \times W_{\mathrm{ISP}}+\mathrm{City}_i \oplus \mathrm{City}_j \times W_{\mathrm{city}}+\mathrm{postcode}_i \oplus \mathrm{postcode}_j \times W_{\mathrm{postcode}}+\mathrm{IP1}_i \oplus \mathrm{IP1}_j \times W_{\mathrm{IP1}}+\mathrm{IP2}_i \oplus \mathrm{IP2}_j \times W_{\mathrm{IP2}}+\mathrm{IP3}_i \oplus \mathrm{IP3}_j \times W_{\mathrm{IP3}}+\mathrm{IP4}_i \oplus \mathrm{IP4}_j \times W_{\mathrm{IP4}} \quad (1)$$

$$TotalDelay(i,j)= Delay(i,j)(1+L/2048) \quad (2)$$

In formula 1, $\oplus$ means exclusive OR operation. If ISP_i is equal to ISP_j, then $ISP_i \oplus ISP_j$ is 0, otherwise 1. W_{ISP} means the weight of *ISP* to the delay. It means that only nodes from different ISP can affect the delay in ISP item, so does other factors in this formula. Let the first byte of internet address of peer *i* is $IP1_i$, $IP1_i \oplus IP1_j$ compares the first byte of two addresses. Then *IP2, IP3* and *IP4* compare the second, third, fourth byte of the two peers' IP address, respectively. We set the weight of each factor as W_{ISP}=500, W_{city}=200, $W_{postcode}$=100, W_{IP1}=100, W_{IP2}=100, W_{IP3}=100, W_{IP4}=50. Because we send a message after its previous message has been sent, suppose we get a delay 50ms through formula 1, and formula 2 adds the effect of messages length to the delay, if the sending queue consists of 3 messages with the size 50, 10240, 10240 bytes, we get the total delay 50ms, 100ms, 100ms according to formula 2, then the completion sending time of the 3 messages are 50ms, 150ms and 250ms, respectively.

There are already some metrics to evaluate a peer to peer live streaming system, such as link stress method [6], and data path quality method [20]. Because in TCMM, there is no physical topology to evaluate the link stress, on the other hand, TCMM does not transmit media data through a multicast tree, thus avoids evaluating the data path quality either, therefore, in this paper, we use other metrics, such as starting delay, dynamic resistance, and overhead to evaluate the performance of TCMM. Each experiment result is got by averaging 5 tests cases.

4.2 Control Overhead

This index is categorized by tree overhead and mesh overhead. Tree overhead is defined as the ratio of the bytes that a peer received to maintain the tree structure over the total bytes a peer received. Mesh overhead is defined as the ratio of the bytes that a peer received to exchange window map over the total bytes the peer received. The tree overhead mainly includes alive messages cost happened in a peer periodically sends this message to its children and receives them from its parents. Fig.6 presents the average tree overhead of TCMM. The data are collected when sending an *"Alive"* message every 5 seconds. This figure implies that the tree overhead is nearly independent of the community size. That is because the alive messages are only sent to children by father, and children have no responsibility to answer them. So, when a peer can accept more children, its own overhead increases, but its children's overhead will decrease. This will cause the average overhead changes a little. Fig.6 also depicts that the tree maintain overhead is less than 0.5% of the total traffic.

Every peer also exchanges ping/pong messages with its inactive partners to declare its aliveness and exchanges their window map messages with active partners as well. Fig.7 shows that when the Ping/Pong interval is 9 seconds and window map exchange interval is 2 seconds, the total mesh overhead is less than 2% when number of minimum providers less than or equal to 6. Considering the mesh overhead increases with more partners, we believe that minimum provider partner equal to 4 is a good practical choice. So it is adopted in the following experiments, and this result also meets the point got from [24]. However, an important fact here is that number 4 is just for provider partner not for the total active partner.

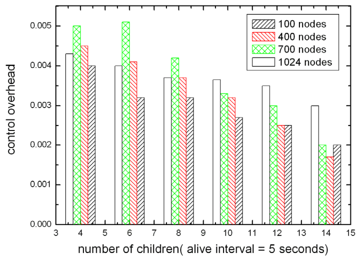

Fig. 6. Overhead of tree maintaining

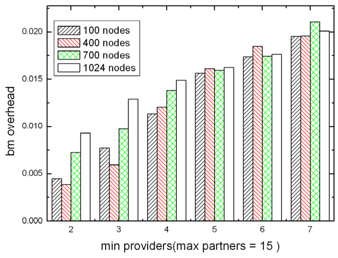

Fig. 7. Overhead of mesh maintaining

4.3 Starting Delay

This index is defined as the time period from a peer joins the multicast system to a peer starts to play back the media. This index describes how fast the system can provide service to a newcomer. Fig.8 presents the comparison of starting delay between TCMM and mesh-based scheme, this figure is for 1024 nodes. Actually the starting delay is almost independent of the system size.

We have ever thought that TCMM will have less starting delay than pure mesh-based structure, because peers in pure mesh-based overlay need much time to optimize their service providers, and this will increase the starting delay. However, data in Fig.8 proves it wrong. This data leads to a conclusion that although the TCMM provides a quick way to identify those nearby nodes, it has a little longer starting delay to build the control tree before starting to get media data, which causes about additional 4s-10s delay than pure mesh-based structure.

4.4 Dynamic Resistance

Because P2P environment is a dynamic environment, many peers' frequently joining and leaving will cause the source of each peer to become dynamic, therefore, a peer should have the ability to change at least part of its service providers at any time. We let the overlay with 1024 peers runs stably for 5 minutes, then we let a randomly produced 2 new peers join the overlay and another randomly selected 2 peers leave the overlay each second within 200 seconds.

In Fig.9, the *y* axis is sampling times of the window size of peers, *x* axis is the window size. We observe that the TCMM's window is fuller than the pure mesh-based method in most times. We set the dynamical peers ranging from 10 to 50, TCMM scheme produces a better average window size as shown in Fig.10. There are two reasons for this phenomenon. 1) Although the peers frequently join and leave, the peers in TCMM always fetch and transmit new segments before old segments. Definitely, this will accelerate the distribution of new segments (since most of peers are lacking new segments not old segments) and speeds up the data distribution dramatically. Also, this strategy strengthens the collaboration among peers. 2) The peers in TCMM can get provider partners efficiently from the control tree and reduce the effects of dynamics of peers.

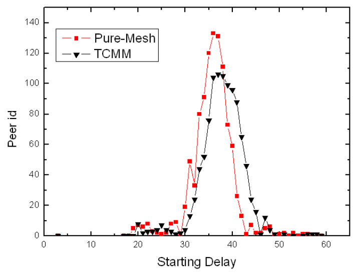

Fig. 8. Comparison of startup delay

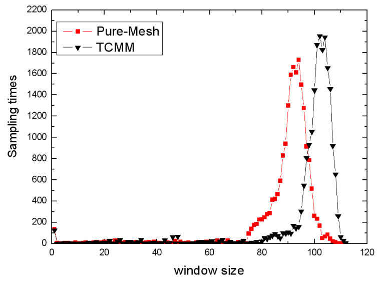

Fig. 9. Comparison of continuity

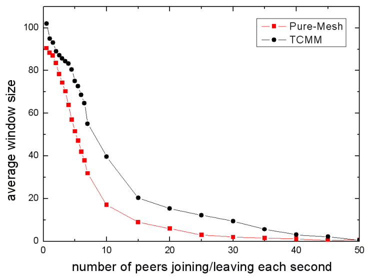

Fig. 10. Comparison of resistance to dynamics

5 Conclusions

In this paper, we have presented TCMM approach, which can support large scale live streaming service. TCMM just integrates two overlays, a tree based on GUID to overcome the mismatch problem between logical overlay and underlying physical networks, and a mesh to resist peer dynamics, instead of excluding any of them. The simulation results show that this approach not only benefits the overlay efficiently, decreases the time used to find close nodes, which is very important in reducing the redundancy of the P2P traffic, but also it strengthens the stability in a rigorous dynamic environment just by introducing additional slight starting delay.

References

[1] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems", In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.

[3] A. Rowstron, A. M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The Design of a Large Scale Event Notification Infrastructure", In *Proc. of 3rd International Workshop on Networked Group Communication*, Nov. 2001.

[4] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: an Infrastructure for Fault-Tolerant Wide-Area Location and Routing", Technical Report, UCB/CSD-01-1141, University of California, Berkeley, CA. USA, Apr. 2001.

[5] Y. Chawathe, "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service", Ph.D. Thesis, University of California, Berkeley, Dec. 2000.

[6] Y. H. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast", In *Proc. of ACM SIGMETRICS 2000*.

[7] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh", In *Proceedings of SOSP 2003*.

[8] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, "Using Random Subsets to Build Scalable Network Services", In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, March 2003.

[9] P. Francis, "Yoid: Extending the Multicast Internet Architecture", White paper, http://www.aciri.org/yoid/, 1999.

[10] J. Liang and K. Nahrstedt, "DagStream: Locality Aware and Failure Resilient Peer-to-Peer Streaming", In *Proc. of SPIE MMCN 2006*.

[11] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-Peer Live Streaming Service", In *Proc. of IEEE INFOCOM 2006*.

[12] J. Liang and K. Nahrstedt, "Randpeer: Membership Management for QoS Sensitive Peer to Peer Applications", In *Proceedings of IEEE INFOCOM 2006*.

[13] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer to Peer Media Streaming Using CollectCast", In *Proc. of ACM Multimedia 2003*.

[14] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network", *IEEE Internet Computing*, 2002.

[15] M. Castro, P. Druschel, A M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth Multicast in a Cooperative Environment", In *Proc. of SOSP 2003*.

[16] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", In *Proc. of 3rd Usenix Symposium on Internet Technologies & Systems*, March 2001.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network", In *Proc. of ACM SIGCOM 2001*.

[18] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content Addressable Networks", In *Proc. of 3rd International Workshop on Networked Group Communication*, Nov. 2001.

[19] S. Saroiu, P. Gummadi, and S. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”, In *Proc. of MMCN 2002*.
[20] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable Application Layer Multicast”, In *Proc. of ACM SIGCOMM*, 2002.
[21] D. A. Tran, K. A. Hua, and T. T. Do, “ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming”, In *Proceedings of IEEE INFOCOM 2003*.
[22] V. N. Padamanabhan, H. J. Wang, P. A. Chou, and K. Scripanijkuichai, “Distributing Streaming Media Content Using Cooperative Networking”, In *Proc. of ACM NOSSDAV 2002*.
[23] V. Venkataraman, P. Francis, and J. Calandrino, “Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast”, In *Proc. of IEEE IPTPS 2006*.
[24] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, “CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming”, In *Proc. of INFOCOM 2005*.
[25] X. Tu, H. Jin, X. Liao, and J. Cao, “Nearcast: A Locality-Aware Application Level Multicast for Peer-to-Peer Live Streaming Service”, To appear in *ACM Transactions on Internet Technology*, 2007.
[26] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, “Location-aware Topology Matching in Unstructured P2P Systems”, In *Proc. of INFOCOM 2004*.
[27] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz, “Bayeux: An Architecture for Scalable and Fault Tolerant Wide-area Data Dissemination”, In *Proc. of NOSSDAV 2001*.