http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (5): 728-736, 2008
ISSN 1812-5638

TapMulti: A Scalable and Low-Delay Application-Layer Multicast Protocol on Tapestry Overlay Network

Qing-hua Zheng, S. Jiang, F. Zhang, T. Peng and C. Chen
Laboratory of MOE KLINNS and Laboratory of SKLMS, Department of Computer Science and Technology, Xi'an Jiaotong University, 710049, Shaanxi, People's Republic of China

Abstract: In this study, we propose a low-delay ALM protocol named TapMulti on Tapestry to achieve scalability and low-delay, . In TapMulti protocol, a delivery tree is designed to reduce end-to-end delay and improve stability of multicast system. This low-delay delivery tree is constructed on Tapestry and it can guarantee a tradeoff between delay and network-traffic load of multicast system by constraining width (maximal out-degree of node in the delivery tree) and depth of the delivery tree. Moreover, the efficient and proportional route mechanism of Tapestry is exploited to decrease the control cost to maintain multicast delivery tree. Simulated results indicate that, compared with other existing ALM approaches on Tapestry, TapMulti is of distinct advantages in aspects of end-to-end delay and control cost.

Key words: Application-layer multicast, low-delay, tapestry

INTRODUCTION

With the widespread availability of inexpensive broadband Internet connections for home users, a large number of bandwidth-intensive applications have now become practical. Typically, video broadcast is one popular kind of such applications (Guo and Lo, 2008). Video broadcast applications require two main performances: scalability (Tian *et al*., 2008) and low-delay (Wu *et al*., 2000). IP multicast seems to be the ideal solution for content distribution over the internet. However, IP multicast was never widely deployed in the Internet (Zhu *et al*., 2007; Zhong *et al*., 2005). To achieve scalability and low-delay, at first, many Application-Layer Multicast (ALM) approaches over unstructured overlay networks have been proposed and applied for video broadcast system (Sonia and Minseok, 2007), such as ESM (Chu *et al*., 2002), ALMI (Pendakaris and Shi, 2001) and CoopNet (Padmanabhan *et al*., 2002). These ALM approaches are designed on unstructured overlay networks. In unstructured overlay networks, a node connects with a group of other nodes randomly and the nodes construct an irregular network topology. Therefore, the control messages of multicast system are exchanged inefficiently between nodes. These factors limit scalability of ALM system. As a result, some approaches of application-layer multicast over structured overlay networks, called the second generation of P2P systems (Leslie and Zimmermann, 2006), have been developed. Compared with unstructured overlay networks, these overlays networks, such as Kademlia (Maymounkov and Mazieres, 2002), Chord (Stoica *et al*., 2001), Pastry (Antony and Druschel, 2001), CAN (Ratnasamy *et al*., 2001), Delaunay (Jörg *et al*., 2002) and Tapestry (Zhao *et al*., 2004), implement a basic key-based routing (KBR) interface that supports deterministic routing of messages to a live node that has responsibility for the destination key. They can also support higher level interfaces such as a distributed hash table or a DOLR layer (Vishnevsky *et al*., 2008) . These systems scale well and guarantee that queries find existing objects under nonfailure conditions (Hai and Feang, 2008). So, structured overlay networks are more suitable than unstructured overlay networks for large-scale multicast system applications (Garces-Erice and Biersack, 2005).

In this study, we propose TapMulti, a low-cost, low-delay, application-layer multicast protocol on Tapestry that has already enabled the deployment of global-scale storage applications such as OceanStore (Kubiatowicz *et al*., 2000). TapMulti contributes solutions to three important problems of application-layer multicast in P2P environments: high delay, tree maintenance inefficiency and single-bottleneck.

Corresponding Author: Shan Jiang, Laboratory of MOE KLINNS and Laboratory of SKLMS, Department of Computer Science and Technology, Xi'an Jiaotong University, 710049, Shaanxi, People's Republic of China

High delay: For application-layer multicast, end-to-end delay lies on two factors: the sum of logical links from sender to receiver and the transmission speed of each link (Yuval and Tomer, 2008). The structure of multicast delivery tree determines these two factors. In general, smaller the depth of the delivery tree is, lower the multicast delay is. But for a certain amount of peers, decreasing the depth lead to increasing out-degree of interior nodes, that increase network traffic load so that exceed available bandwidth of network-layer connection. It causes a lot of loss packet and this bad case should be avoided in practical system (Jin *et al*., 2008). Decreasing the depth of the delivery tree is a challenge without a lot of loss packet or with a few (Hsiao and He, 2008).

Bayeux, a classic ALM approach over Tapestry, takes advantage of the nature of Tapestry unicast routing to build an application-layer multicast system. However unlike most existing application-layer multicast systems, in Bayeux, a delivery tree consists of non-receiver interior nodes. This is a unique feature of the Bayeux multicast system. This feature increases the amount of logical links between the source node to receiver nodes.

In TapMulti, the architecture of multicast delivery trees is designed according to the rule of Node ID prefix-layered matching. This architecture can constrain width and depth of delivery tree by the node ID structure of Tapestry. Therefore, the multicast delivery tree guarantees a tradeoff between delay and network traffic load of multicast system. Superior to the high-delay delivery tree of Bayeux, the delivery tree of TapMulti consists of receiver nodes only. This feature makes the amount of logical links smaller and decrease the delay from source node to receiver nodes.

Tree maintenance inefficiency: To maintain a delivery tree, a lot of control messages are produced for exchanging information between a node and another node in P2P environment, when a node is joining, leaving or is in failure. The overhead control messages bring complexity and communication cost to multicast system. It can increase response latency of the system and even decrease stability of the system.

In DoMulti, another ALM approach over Tapestry, the process of node joining phase is so primitive and recursive that it produces a large number of control messages. In TapMulti, a message of a new joining group phase is handled previously by each node where the message past along Tapestry route path, before it route to the destination node. This pre-handle, determining the relationship between the new node and routing nodes, can help to join multicast tree successfully halfway, if the relationship comply with some rules. So, it reduces the routing hops of control message on overlays.

Single-bottleneck: To join multicast delivery tree, the JOIN message of each new node is sent to the root node (the source node of the multicast system) and is handled by it in Bayeux and DoMulti. As a number of new nodes join the tree concurrently, the workload and network traffic load of the root will be enormous so that exceed the computing capability of the root node. If this case happens, the whole multicast system will break down. So, the root node is single-bottleneck node of the multicast system. In TapMulti, when the JOIN messages are sent to the root, the nodes which them past by along Tapestry routing paths pre-handle the messages. In the pre-handling process, some nodes can handle the JOIN messages and make the new node joining the tree successfully. Thereby, the method can reduce the workload of the root node effectively and possibility of single-bottleneck.

This study describes the techniques TapMulti use to address each of these problems, providing a complete solution for real-time video broadcast system. Simulated results indicate that, compared with other existing ALM approaches on Tapestry, TapMulti is of distinct advantages in aspects of end-to-end delay and control cost.

TAPESTRY OVERLAY NETWORK AND BAYEUX MULTICAST

Tapestry routing and location: Tapestry, a peer-to-peer overlay routing infrastructure, offers efficient, scalable, location-independent routing of messages directly to nearby copies of an object or service using only localized resources (Zhao *et al*., 2004). Tapestry nodes participate in the overlay and are assigned node ID uniformly at random from a large identifier space. Tapestry assumes node ID and GUID are roughly evenly distributed in the namespace, which can be achieved by using a secure hashing algorithm like SHA-1. In Tapestry overlay network, Node ID is a fixed-length bit-sequences as $a_n\ a_{n-1}...a_i a_{i-1}...a_2 a_1$. Where, a_i is an integer between 0 and 2^b-1. n_T denotes length of the sequence; b_T denotes the base of Node ID and b_T is equal to 2^b; N_T denotes the namespace size of Node ID and N_T is equal to $(b_T)^{n_T}$.

Tapestry uses local routing maps at each node, called neighbor maps, to incrementally route overlay messages to the destination ID digit by digit (e.g., ***8—>**98—>*598—>4598 where *'s represent wildcards). This approach is similar to longest prefix routing in the CIDR (Classless InterDomain Routing) IP address allocation architecture. A node has a neighbor map with multiple levels, where each level represents a matching suffix up to a digit position in the ID. A given level of the neighbor map contains a number of entries equal to the base of the ID, where the ith entry in the jth level is the ID and location of the closest node which ends in i+ suffix (N, j-1). For example, the 9th entry of the 4th level for node 325AE is the node closest to 325AE in network distance which ends in 95AE. This routing method guarantees that any existing unique node in the system will be found within at most $Log_{b^T} N_T$ logical hops.

Bayeux basic concepts and improvement: There are two existing application-layer multicast approaches on Tapestry: Bayeux (Shelley *et al*., 2001) and DOMulti (Lin *et al*., 2007). In Bayeux, group session is described as <Session Name, UID> and Session Name is a title of group content, UID is a global, unique identity descriptor, that distinct in different groups. The group session is published to the group source node (as delivery tree's root) through Tapestry route path. Bayeux constructs and maintains a delivery tree using four types of control message: JOIN, LEAVE, TREE and PRUNE. A new member X sends a JOIN request message to the root along Tapestry route path. After receiving JOIN request message, the root responses TREE message to X on Tapestry route path. This path on TREE message's way to X sets up a branch of Bayeux delivery tree. Figure 1 shows an example where node 7876 is the root of a multicast session and node 1250 tries to join. Bayeux multicast is scalable and fault-tolerant. But many non-member interior nodes in the delivery tree result in two drawbacks. First, it increases the distance between source node and receiver nodes. Second, as each of these non-member nodes leaves or is in failure, the delivery tree need to be rebuilt. So, the stability of multicast system is influenced.

THE PROTOCOL OF TAPMULTI

In TapMulti protocol, to achieve scalability, low-delay, stability and self-organization, we use the following four principles to build the multicast delivery tree:

- To balance the end-to-end delay and network traffic load of a parent node, the width and depth of the multicast tree are restricted effectively in terms of the delivery tree architecture with Node ID prefix-layered matching
- To maintain the multicast delivery tree in distributed way. Every member node X in delivery tree must record Parent(X) and Children(X) information (such as ip, port, name). To reduce the control messages of system, the multicast delivery tree is used for not only video stream delivery but some kinds of control messages communication
- To route and exchange control messages efficiently, all control messages are delivered by unicasting over Tapestry overlay network or the multicast delivery tree. The JOIN Message is delivered over Tapestry; the LEAVE, TREE, PRUNE and UPDATE Messages are delivered by unicasting over the multicast delivery tree
- To reduce end-to-end delay, the multicast delivery tree consists of member nodes only. To compare with Bayeux, this principle can avoid the bad influence of the system stability, when the non-member nodes leave or are in failure. So it can improve stability of the multicast delivery tree as well

The main of TapMulti protocol is the multicast delivery tree maintenance. It includes: node joining group, node leaving group and handling failure of node.

Architecture of the multicast delivery tree: In TapMulti protocol, a multicast delivery tree must have four features: scalability, low-delay, stability and self-organization. To implement these features, the architecture of multicast delivery tree is designed in terms of the rule of Node ID prefix-layered matching. This rule can be expressed as Eq. 1.

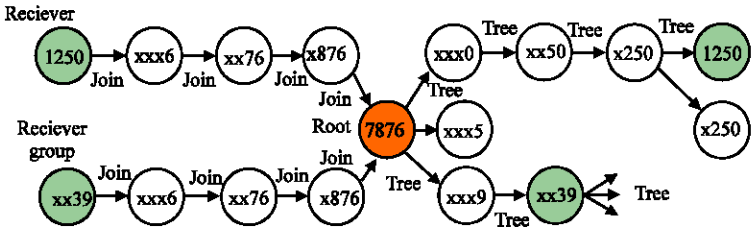

Fig. 1: Bayeux multicast delivery tree maintenance

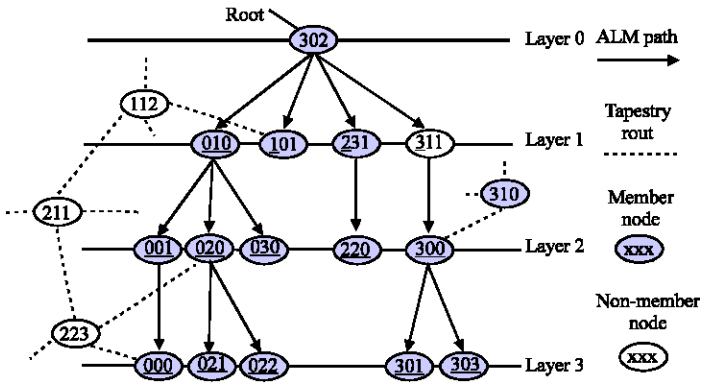

Fig. 2: An instance of TapMulti multicast delivery tree

$$(\forall j)\, j \in \text{Children}(i) \Leftrightarrow \begin{cases} \text{Prefix}(j,\text{layer}(i)) = \text{Prefix}(i,\text{layer}(i)) \\ (\neg\exists k)\left(j \neq k \wedge k \in \text{Children}(i) \wedge \text{Prefix}(j,\text{layer}(i)+1) = \text{Prefix}(k,\text{layer}(i)+1)\right) \end{cases} \quad (1)$$

where, i, j, k represent three nodes in the multicast delivery tree. Children(i) is the i's children set. Layer(i) is the i's layer in multicast delivery tree and Perfix(i, m) denotes the m digital sequence of prefix i's ID and m is usually a nonnegative integer

The rule defines the parent-child relationship between all nodes in the multicast delivery tree. According to Eq. 1, if j is i's a child and the i's layer is m, the m digital prefix of i's ID must be equal to the m digital prefix of j's ID and the (m+1)th digital prefix of j's ID must be different from the (m+1)th digital prefix of the each other i's children IDs. In Fig. 2, the nodes 001, 020 and 030 are in Layer 2 and they have an identical parent node 010. As they are in Layer 2, the one digital prefix of each their IDs is same as the one digital prefix of their parent ID and is '0'. Moreover the 2nd digital prefix of each their IDs must be different.

According to the rule of Node ID prefix-layered matching, it's very clear that the maximal depth of the delivery tree is n_T and the maximal out-degree of every node is b_T (n_T denotes length of Node ID; b_T denotes the base of Node ID). It's well-known that the end-to-end delay of multicast delivery tree depends on depth of the tree and the network-traffic load determined by the out-degree of a node. Therefore, the multicast delivery tree based on this architecture can guarantees a tradeoff between end-to-end delay and network-traffic load of multicast system by the two parameters of n_T and b_T.

Join group: In Bayeux and DOMulti, there are two basic steps in node joining phase. First, a JOIN request is sent from new receiver node to root node on Tapestry. Second, after receiving the JOIN request, root node sends the TREE message to the new receiver node on Tapestry. Therefore, all JOIN requests need to be handled by the root node of delivery tree. This process of node joining group leads to a large number of control messages and a single-bottleneck of the root in the system.

Actually, not all of the JOIN requests need to be handled by the root node. While some lucky JOIN requests route to root node over Tapestry, they maybe pass some member nodes already existing in TapMulti delivery tree. Each of these member nodes can handle the JOIN request and it determines the relationship between it and the new joining node in terms of Node ID prefix-layered matching. By means of the relationship, the new joining node sends a TREE message along the multicast delivery path to search it's parent node. Because the new node position in the multicast delivery tree is found more efficient along the multicast delivery path than along Tapestry route path, the process mentioned above speeds up joining the multicast tree. Moreover, the JOIN request needn't route to the root node. So, the process of joining group has been optimized to reduce the control message of maintaining system and the number of messages handled by root node. In TapMulti protocol, a whole joining phase has three steps as follows (S Stands for the root of a TapMulti tree):

Step 1: A new node i sends a JOIN request to S on Tapestry.

Step 2: every member node which the JOIN request past by deals with it according to the algorithm of node ID layered prefix match. There are three cases for the match as follows (j is a member node that i's JOIN request past by on Tapestry):

Case 1: Prefix(i, 1) ≠ Prefix(j, 1). That is i shouldn't be placed at the same branch as member j in TapMulti tree. Continue to route the JOIN request to the root on Tapestry.

Case 2: Prefix(i, layer(j)) = Prefix(j, layer(j)). That is i must be a descendant of j and furthermore we need judge whether member j is i's parent according to Eq. 1. If Eq. 1 is satisfied, node i is a member j's child and the process of i join group is completed successfully. Otherwise we need search j's children and try to find i's parent node. We use TREE message to locate i's parent. This message is send to a node k which is j's child node and satisfies Prefix(i, layer(k)) = Prefix (k, layer(k)). The node k judges the equation of Eq. 1. This procedure is executed recursively until i find it's parent and finish the join process.

Case 3: Prefix(i,layer (j)) ≠ Prefix(j,layer(j)) and Prefix(j, m), 1 ≤m<layer(j). That is i should position at the same branch as j in delivery tree, but i can't be a descendant of j. Then j sends a JOINTREE request for i to j's parent in order to lookup i's ancestry. The JOINTREE request is transferred upward through member node's parent continuously, until a member node k whose ID satisfies Prefix(i,layer(k)) = Prefix(k,layer(k)) is found. Then like Case 2-2, i uses k's information to join the tree and finally completes the join process.

Step 3: S has received the JOIN request and the JOIN request is handled according to Case 2-2 by S. Then, the process of join group is completed successfully.

In Fig. 3a, the path of JOIN request message of node 101 over Tapestry overlay network is: 101→112→002→302. Both member 002 and member 302 handle the JOIN request. In the member 002, Case 1 is accorded with. So, member 002 continues to route JOIN request to next node over Tapestry. In the member 302, the Step 3 is executed and the procedure is jump to Case 2. After implementing the procedure of Case 2, the new node 101 will be as a child of member 102. In Fig. 3b, the path of JOIN request message of node 211 over Tapestry is: 211→222→231→302. in the member 222, the procedure of Case 3 is executed and finally the new node 211 selects member 231 as parent node.

Leave group: To leave the multicast group, the member A needs to execute two operations: One operation is that A notifies B who is Parent (A) it's leaving and waits B update children information. The other is that A notifies one of Children (A) that you will take my place. The member A who will leave group may be in two situations. If A is a leaf node, A just notifies B to remove A from Children (B) for leaving group successfully. In another situation, as A is not a leaf node, the procedure of A leaving group has three steps as follow:

Step 1: A selects a substitution A1 in Children (A) to take place of A. Normally the first one in Children (A) is chosen and A sends a LEAVE message that includes A1 information to B. After receiving LEAVE message from A, B substitutes A from Children (B) with A1 and B

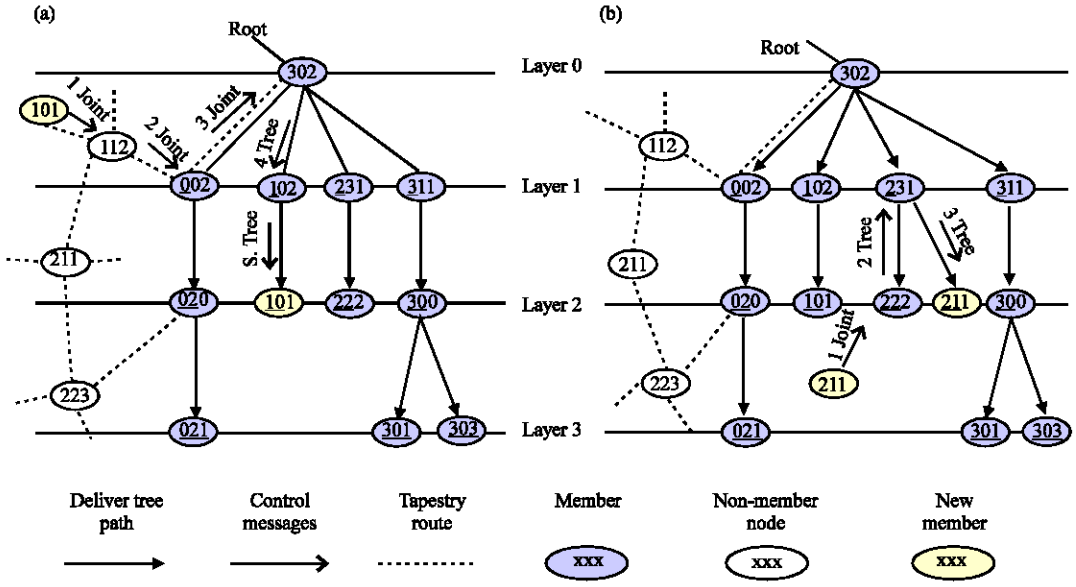

Fig. 3: Node joining phase (a) the situation of step 3 and (b) the situation of care 2-3

responses a LEAVE_ACK message to A for acknowledgment. Generally, the result of ACK represents SUCCESS.

Step 2: A needs to reconstruct the subtree. A sends a A's UPDATE message including the information of Parent (A) and Children (A) to A1. After having receiving the UPDATE message, if A1 isn't the leaf of the tree, A1 sends a A1's UPDATE message including Parent (A1) and Children (A1) to A11 that A1 chooses as A1's substitution node continuously. The recursive process is not stopped, until the last substitution node AXXX is a leaf node. At this time, AXXX updates Parent (AXXX) with Parent (pre-node) and inserts Children (pre-node) into Children (AXXX). Then AXXX responses a UPDATE_ACK message. Finally the UPDATE_ACK message is transferred to A.

Step 3: After having receiving UPDATE_ACK message, A can exit successfully. The multicast delivery tree has been rebuilt.

In Fig. 4a, the node 311 is leaving group. The node 300 is selected as substitution of 311. Furthermore, the node 301 is chosen as substitution of 300. Meanwhile, the root 302 removes 311 from Children (302) and inserts 300 into Children (302). Figure 4b shows that the delivery tree has been rebuilt after the node 311 leaved. Note that the structure of delivery tree complies with the rule of node ID prefix-layered matching still, after the delivery is rebuilt. Eq. 1 is always satisfied.

Handle node's failure: Tapestry overlay is based on DHT overlay network and it provides reliable mechanism of fault-tolerant to maintain overlay network's connectivity, as all nodes are highly dynamic in the P2P network.

For TapMulti, it's necessary to provide reliable mechanism of reconstructing delivery tree resulting from failure of node. This fault handling mechanism includes two parts.

Detecting node's failure: TapMulti sets up heart-throb linkages between parents and children to detect nodes' failure. A parent node in delivery tree sends a KEEPALIVE message periodically to all its children and every child must response a KEEPALIVE_ACK message to its parent immediately after having received KEEPALIVE. For parent A, if it has received KEEPALIVE_ACK message from a child A1 during the period of T_{max} (as the max active time and is 35 sec in common), A1 is regarded as activity. Otherwise, A1 is deactive and is in failure. For the child A1, if it has received KEEPALIVE from A during T_{max}, its parent A is active. Otherwise, A is in failure.

Reconstructing subtree: We can easily detect whether a node is active or not by the method above. If a node is not active, according to it's role, it can be handled as follows:

- Failure of a A's child: if A detects its child node A1 is failed, A just removes A1 from Children (A)
- Failure of A1's parent: if A1 detects its parent is failed, A1 should initiate a join request to the root of the TapMulti tree and A1 will update the layer information of Children (A1) after rejoining successfully

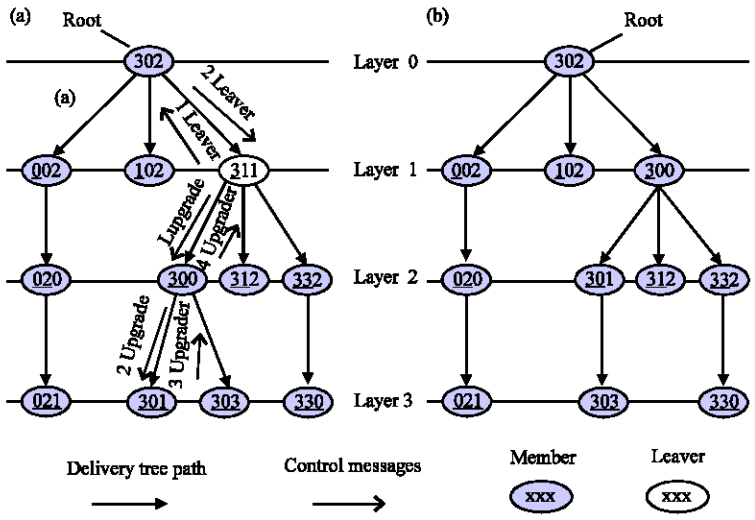

Fig. 4: The node 311 leave group (a) before 311 leave and (b) after 311 leave

EXPERIMENT AND RESULTS

Here, we compare Bayeux or DOMulti to TapMulti in two aspects: end-to-end delay and the control cost of multicast system.

Performance metrics: We adopt the four metrics to evaluate the effectiveness of our application-level multicast technique:

- **AMD (Average Multicast Delay):** a measure of average value of sum of every path end-to-end delay between source node (root node) and receiver node. It can be expressed as follow:

$$AMD = \frac{\sum_{i=0, v_i \neq r}^{n-1} d(r, v_i)}{n-1}, v_i \in G \quad (2)$$

where, d(r,v) is the end-to-end delay between node r and node v; G denotes the set of all nodes in multicast group; n is the size of G:

- **JGL (Join Group Load):** The sum of messages on overlay network when the certain number of nodes joined group in once time
- **JLR (Join Load of Root):** The count of JOIN requests handled by the root node when the certain number of nodes joined group in once time
- **JLL (Join and Leave Load):** The sum of messages on overlay network when the certain number of nodes join group and leave group in once time successfully.

Simulation setup: To evaluate present protocol, we implemented Tapestry unicast routing. And the Bayeux tree protocol, the DOMulti tree protocol and the TapMulti tree protocol have been implemented as three packet-level simulators. There measurements focus on end-to-end delay on network only and do not model the effects of any cross traffic or router queuing delays.

We used a topology generated by GT-ITM with the transit-stub model consisting of 20000 nodes. And the Tapestry overlay uses node namespace size of 4096, ID base of 4 and a multicast group size of 4096 members.

Snapshot measurements: We select 4096 nodes as Tapestry nodes from 20000 nodes randomly and evaluate metrics with adjust the size of group. The minimum size of group is 50 and the size of group is increased by the step size 50 to the maximal size of 4096. For every certain size of group, we measure the metrics in 20 times randomly and calculate the average value in 20 times as the result. This multiple random processes can eliminate the uncertainty caused by measuring in single time.

In Fig. 5, the variation of average multicast delay is compared in TapMulti to that in Bayeux. In the same size of group, AMD of TapMulti is shorter than Bayeux. Especially, in small group, this advantage is obvious. The bigger the size of group is, the closer the performance of AML in Bayeux multicast is to that in TapMulti multicast. This result verifies that non-members in multicast tree increase the end-to-end delay as we mentioned earlier.

In Fig. 6, the variation of performance of JGL is compared in Bayeux, DOMulti and TapMulti. At the certain size of multicast group, the number of join group messages of TapMulti is a litter smaller than that of Bayeux and is a much smaller than that of DOMulti. This result show that the primitive, recursive join process in DOMulti suffer from heavy workload and our protocol has been improved much this performance with efficient utilization of member

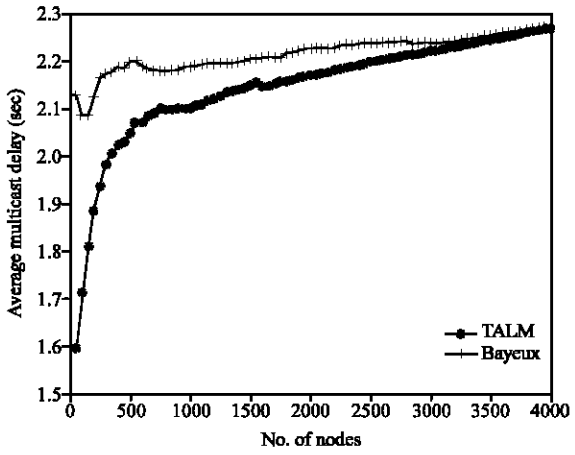

Fig. 5: Average multicast delay

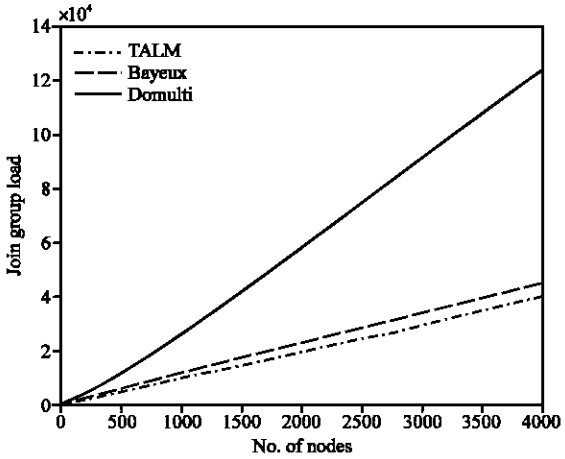

Fig. 6: The messages of join group phase

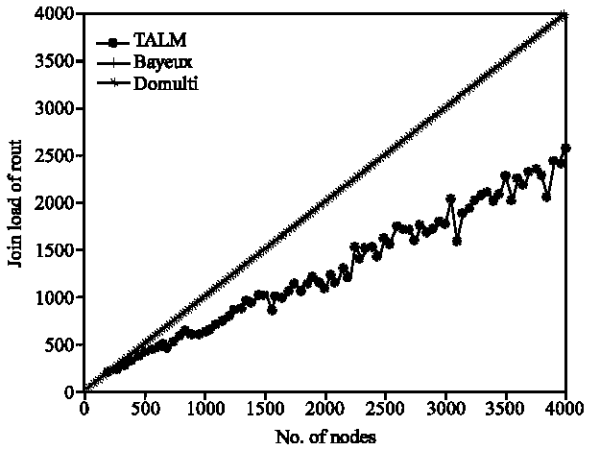

Fig. 7: The messages Handled by the Root in Joining phase

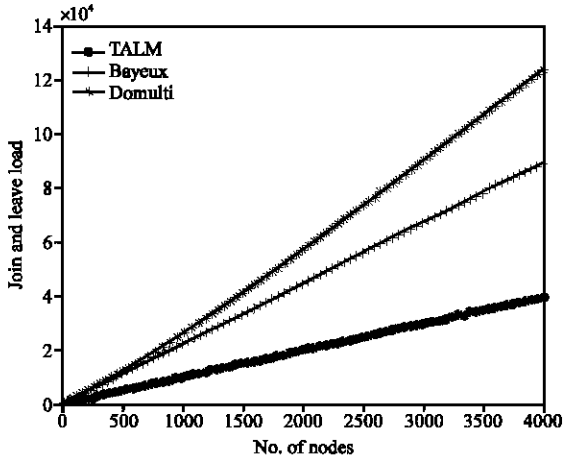

Fig. 8: Sum of messages in Joining and Leaving phase

information in JOIN message route phase. Figure 7 show that the number of our TapMulti's handling join request message is smaller than Bayeux and DOMulti. Both the number of root's handling join request messages in Bayeux and in DOMulti are equal to the size of group.

In Fig. 8, the variation of JLL is compared in Bayeux, DOMulti and TapMulti. TapMulti optimizes the join and leave phase and the sum of JOIN and LEAVE message is smaller than Bayeux and DOMulti.

CONCLUSION

In this study, we propose an efficient, low-delay, application-layer multicast protocol named TapMulti on Tapestry for video stream delivering. In TapMulti protocol, a low-delay multicast delivery tree is constructed over Tapestry overlay network and it guarantees an tradeoff between delay and network-traffic load of multicast system. The efficient and proportional route mechanism of Tapestry is exploited to reduce the cost of maintaining the delivery tree. Compared with Bayeux, to reduce of the delay of delivery tree and improve the stability of the tree, the delivery tree of TapMulti does prune non-member nodes of the delivery tree. Compared with DOMulti, in TapMulti, the route path over Tapestry overlay network is exploited efficiently to improve the process of search for parent node rapidly in node joining phase. So, the control message of multicast system is reduced and the scalability of system is improved. Simulated results indicate that TapMulti protocol not only keep scalability of application-layer multicast on P2P overlay network, but make video stream multicast low-delay. It is suitable for scalable video broadcasting application requiring low-delay in P2P.

ACKNOWLEDGMENTS

This research was supported by the National High Tech. Development Plan No. 2006BAH02A24-2.

REFERENCES

Antony, R. and P. Druschel, 2001. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. Proceeding of IFIF/ACM International Conference on Distributed System Platforms, 2001, Springer-Verlag, Springer House 8 Alexandria Road London SW19 7JZ UK., pp: 329-350.

Chu, Y.H., S.G. Rao and S. Seshan, 2002. A case for end system multicast. IEEE J. Selected Areas Commun., 20: 1456-1471.

Garces-Erice, L. and E.W. Biersack, 2005. MULTI+: A robust and topology-aware peer-to-peer multicast service. Comput. Commun., 29: 900-910.

Guo, H. and K.T. Lo, 2008. Cooperative media data streaming with scalable video coding. Trans. Knowledge Data Eng., 20: 1273-1281.

Hai, Z. and L. Feng, 2008. Distributed suffix tree overlay for peer-to-peer search. IEEE Trans. Knowledge Data Eng., 20: 276-285.

Hsiao, H.C. and C.P. He, 2008. A tree-based peer-to-peer network with quality guarantees. IEEE Trans. Parallel Distributed Syst., 19: 1099-1110.

Jin, X.Y., W.P. Ken and C.S.H. Gary, 2008. Loss recovery in application-layer multicast. IEEE Multimedia, 15: 18-27.

Jörg, L., M. Nahas and W. Si, 2002. Application-layer multicasting with delaunay triangulation overlays. IEEE J. Selected Areas Commun., 20: 1472-1488.

Kubiatowicz, J., D. Bindel and Y. Chen, 2000. Ocean store: An architecture for global-scale persistent storage. ACM Sigplan Notices, 35: 190-201.

Leslie, S.L. and R. Zimmermann, 2006. Adaptive low-latency peer-to-peer streaming and its application. Multimedia Syst., 11: 497-512.

Lin, L.X., Z. Jie, Z. Ling and Y. Zhao, 2007. DOMulti: A delay-optimized P2P application layer multicast protocol. J. South China Univ. Technol., 35: 78-83.

Maymounkov, P. and D. Mazieres, 2002. Kademlia: A peer-to-peer information system based on the XOR metric. Lecture Notes Comput. Sci., 2429: 53-65.

Padmanabhan, V.N., H.J. Wang, P.A. Chou and Sripanidkulchaik, 2002. Distributing streaming media content using cooperative networking. Proceedings of the 12th International Workshop on Network and Operating Systems Support For Digital Audio and Video, May 12-14, Miami, Florida, USA., pp: 177-186.

Pendakaris, D. and S. Shi, 2001. ALMI: An application level multicast infrastructure. The 3rd USENIX Symposium on Internet Technologies and Systems, March 26-28, San Francisco, pp: 49-60.

Ratnasamy, S., Handleym, R. Karp and S.Shenker, 2001. Application-level multicast using content-addressable networks Lecture Notes Comput. Sci., 2233: 14-29.

Shelley, Q.Z., B.Y. Zhao, A.D. Joseph, R.H. Katz and J. D. Kubiatowicz, 2001. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. The Eleventh International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '01, ACM, New York, USA., pp: 11-20.

Sonia, F. and K. Minseok, 2007. Characterizing overlay multicast networks and their costs. IEEE-ACM Trans. Network., 15: 373-386.

Stoica, I., R. Morris, D. Karger, M.F. Kaashoek and H. Balakrishnan, 2001. Chord: A scalable peer-to-peer lookup service for Internet applications. Comput. Commun. Rev., 31: 149-160.

Tian, Y., D. Wu and K.W. Ng, 2008. A novel caching mechanism for peer-to-peer based media-on-demand streaming. J. Syst. Architect., 54: 55-69.

Vishnevsky, V., S. Alexander, Y. Mikhail, G. D. Alexander and S. Eunsoon, 2008. Scalable blind search and broadcasting over distributed hash tables. Comput. Commun., 31: 292-303.

Wu, D.P., Y. T. Hou and Y.A.Q. Zhang, 2000. Transporting real-time video over the Internet: Challenges and approaches. Proceedings of the IEEE, 88: 1855-1877.

Yuval, S. and T. Tomer, 2008. Hyperbolic embedding of internet graph for distance estimation and overlay construction. IEEE-ACM Trans. Network., 16: 25-36.

Zhao, B.Y., L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph and J.D. Kubiatowicz, 2004. Tapestry: A resilient global-scale overlay for service deployment. IEEE J. Selected Areas Commun., 22: 41-53.

Zhong, Y., S. Shirmohammadi and A. El- Saddik, 2005. Measurement of the effectiveness of application-layer multicasting. Instrumentation and Measurement Technology Conference, May 16-19, IMTC, Canada, pp: 2334-2339.

Zhu, J.B., T. Tsuchiya and K. Koyanagi, 2007. Peer-to-peer collaborative application-level multicast. E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, CEC-EEE '07. IEEE, Comput. Society, USA., pp: 228-238.