Overlay Multicast Tree
Minimizing Average Time Delay

Hwangjun Song and Dong Sup Lee

School of Electrical Engineering, Hongik University,
72-1 Sangsudong Mapogu Seoul, Korea 121-791.
hwangjun@wow.hongik.ac.kr

Abstract. In this work, we present an overlay multicast tree constructing algorithm to minimize the average time delay from the sender to end-systems. At the same time, the proposed algorithm considers the computing power and the network condition of each end-system as a control variable and thus we can avoid the disastrous case that loads are concentrated to only several end-systems. The multicast tree is constructed by clustering technique and modified Dijkstra's algorithm in two steps, i.e. tree among proxy-senders and tree in each cluster. By the experimental results, we show that the proposed algorithm can provide an effective solution.

1. Introduction

Recently, Internet plays an important role in multimedia communication area and becomes an important way to obtain the information. It can be understood that the number of end-systems (or hosts) connected to the Internet has been exponentially increasing. Furthermore, the demand of various multimedia services through the Internet has been increasing very fast. Internet architectures can be traditionally divided into two entities, i.e. end-systems called hosts and the network consisting of routers and switches. Generally speaking, QoS (quality of service) and multicast are the most important features that should be added to support various multimedia services and increase network utilization [1]. Since QoS functionality can not be serviced without the help of IP layer, many research efforts have been devoted to QoS of IP layer. So far, various Internet protocols such as DiffServ, IntServ and RSVP have been proposed. Multicast has gained a large amount of interests after IP multicast [2] was proposed. Many research efforts have been focused on IP multicast, that is, IP router supporting multicast function. However, IP multicast has not been widely employed so far since most of current routers do not identify the class D of IP addresses. To fully support IP multicast, all routers in the worldwide Internet must maintain the multicast function. It is unlikely that IP multicast will be widely supported soon or later.

As an alternative of IP multicast, overlay multicast has recently proposed to realize the multicast over the current IP network that does not support multicast functionality. While overlay multicast does not need any new additional modification in IP routers, some of end-systems in the multicast group have to replace the multicast function of IP routers.

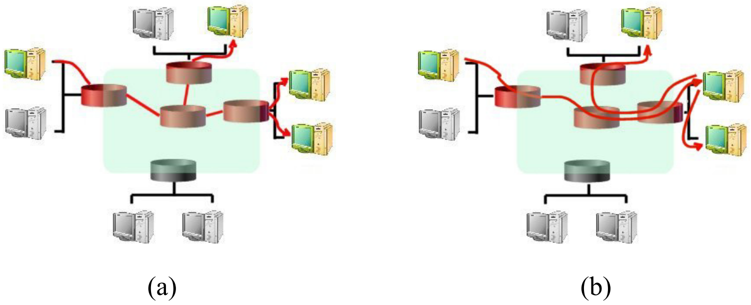


Fig. 1. Comparison between IP multicast and overlay multicast: (a) IP multicast and (b) overlay multicast.

Recently, overlay multicast becomes more feasible since the computing power of end-system becomes very strong due to the fast development of VLSI hardware technology and the network condition of end-system has been rapidly improved due to the fast progress of digital communication and network technology. If the end-system's resources such as the computing power, the storage devices, and network condition are efficiently used, the utilization of the Internet can be improved. The architectures of IP multicast and overlay multicast are shown in (a) and (b) of Figure 1, respectively.

As shown in (a) of Figure 1, router plays an important role to realize the multicast. Routers classify incoming multicast packets and transmit them to routers and end-systems participating in the multicast group only a time. Thus, we don't need to transmit the same packet several times. Many IP multicast protocols can be found in the literature [10]. On the other hand, overlay multicast does not need any modification in routers as shown in (b) of Figure 1. Instead, some end-systems participating in the multicast replace IP routers. So far, many algorithms have been proposed to construct and manage the overlay multicast tree. Y. Chu and et al [1] proposed Narada that constructs an overlay structure among participating end-systems in a self-organizing and fully distributed manner, Liebeherr and Nahas [3] proposed application-layer multicast with delaunay triangulations that each application locally derive next hop routing information without the need for a routing protocol in the overlay, Mathy and et al [4] studied a method to build a hierarchy of nodes, based on the notion of proximity, in a distributed and scalable way, and Park and et al proposed a realistic scheme that is based on the unicast transport from a remote sender to a local subnet and the multicast forwarding to receivers within the subnet in [6]. In addition, many effective algorithms have been proposed [7, 8, 9, 12].

While overlay multicast has many advantages as mentioned previously, there are still several problems to be considered. First of all, overlay multicast can not perform as well as IP multicast, that is, more redundant traffics over physical link are inevitable compared with IP multicast. But, it is still smaller than multiple unicasts. Thus, it may be acceptable. Secondly, more serious problem is the time delay. The time delay is increased compared with IP multicast since traffic must traverse several end-systems until it arrives at the final destination [1]. The increased time delay can be an obstacle for the real-time media delivery. In this paper, we consider an overlay multicast tree constructing algorithm to minimize the average time delay from the sender to the multicast members for the effective real-time media delivery. This paper is organized as follows. Problem formulation and effective multicast tree constructing algorithm are proposed in Section 2, experimental results are provided in Section 3 to show the superior performance of the proposed algorithm, and finally concluding remarks are presented in Section 4.

2. The Proposed Overlay Multicast Tree Constructing Algorithm

As mentioned earlier, we study the overlay multicast tree constructing algorithm to minimize the average time delay from sender to end-systems. RTT (round trip time) is employed as a measure of time delay, and we treat the computing capability and network condition of each end-system as a control variable, that is, the maximum number of streams that each end-system can support is determined considering its limited resources. It is one of the unique features of the proposed algorithm. First of all, we make the following assumptions.

① RTT values among multicast group members are known by using ping commands.
② The number of streams handled by each end-system is advertised to all members.

Under these assumptions, it is overlay multicast impossible to find the optimal multicast tree to minimize the average time delay from sender to end-systems since we have to investigate the full mesh cases since any two end-systems can be connected over IP layer and the number of possible multicast trees increases exponentially in terms of the number of end-systems.

2.1. Problem Formulation

In this scenario, end-systems participating in the multicast are divided into several clusters based on RTT values, and then one of end-systems in each cluster is selected as a proxy-sender that plays a role as a IP router and other end-systems in the cluster receive data from the proxy-sender. The reason is that it is more efficient for a proxy-sender to receive a stream from the remote original sender (or parent proxy-sender) and then distribute it to others when they are closely located than every end-system receives a stream from the remote sender with finite capability. After then, the effective tree among proxy-senders and tree in each cluster are constructed to minimize the time delay (An example constructed by the proposed algorithm is shown in Figure 2.). In this case, the RTT values experienced by a proxy-sender and an end-system can be described as follows.

$$RTT_{p_i}^{s} = \begin{cases} RTT_{p_i}^{s} & \text{if a proxy sender directly received a stream from sender,} \\ RTT_{p_1}^{s} + RTT_{p_2}^{p_1} + \cdots + RTT_{p_k}^{p_{k-1}} + RTT_{p_i}^{p_k} & \text{otherwise,} \end{cases}$$

Where $RTT_{p_i}^{s}$ is the RTT of the proxy-sender p_i and $RTT_{p_k}^{p_{k-1}}$ is the RTT between two adjacent proxy-senders on the path from the sender and the final proxy-sender. And the RTT experienced by an arbitrary end-system is described by

$$RTT_{j}^{s} = RTT_{p_i}^{s} + RTT_{j}^{p_i},$$

where $RTT_{j}^{p_i}$ is the RTT between the j_{th} end-system and the proxy-sender p_i in the i_{th} cluster. Then, the total sum of RTT values is

$$\sum_{i=1}^{n_p} RTT_{p_i}^{s} + \sum_{i=1}^{n_p}\sum_{j=1}^{m_i-1}\left(RTT_{p_i}^{s} + RTT_{j}^{p_i}\right),$$

where n_p is the number of clusters and m_i is the number of end-systems in the i_{th} cluster. Hence, the average RTT value can be expressed by the following Eq. 1. Now, we can formulate our problem to minimize the average RTT as follows.

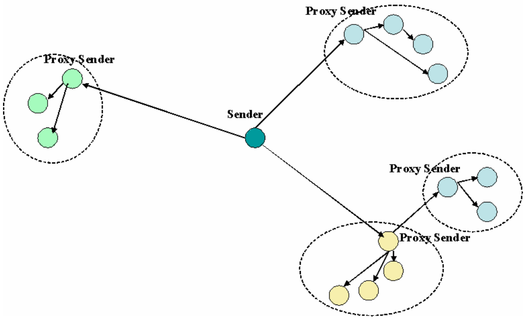


Fig. 2. An example of overlay multicast tree under consideration.

Problem Formulation: Determine the number of clusters and trees to minimize

$$\frac{1}{N}\sum_{i=1}^{n_p} m_i \cdot RTT_{p_i}^{s} + \frac{1}{N}\sum_{i=1}^{n_p}\sum_{j=1}^{m_i-1} RTT_{j}^{p_i}, \qquad (1)$$

$$\text{subject to } s_s \le ST_s \text{ and } s_i \le ST_i,$$

where N is the number of end-systems participating the multicast ($N = \sum_{i=1}^{n_p} m_i$), s_s and s_i are the numbers of streams handled by the sender and the i_{th} end-system respectively, and ST_s and ST_i are the maximum numbers of streams that can be supported by the sender and the i_{th} end-system, respectively. We have to investigate the full meshes among all end-systems to find the optimal solution. The number of possible trees increases exponentially in terms of the number of end-systems, thus it is overlay multicast impossible to find the optimal solution. To reduce the required computational complexity, we neglect the dependency between two terms in Eq. 1. Actually, the dependency could be negligible if $RTT_{p_i}^{s} \gg RTT_{j}^{p_i}$. Under this assumption, we can simplify the above problem as follows.

Simplified Problem Formulation: Determine the number of clusters and a tree among clusters to minimize

$$\frac{1}{N}\sum_{i=1}^{n_p} m_i \cdot RTT_{p_i}^{s} \text{ subject to } s_s \le ST_s, \qquad \textbf{(2)}$$

and then determine tree in each cluster to minimize

$$\sum_{j=1}^{m_i-1} RTT_{j}^{p_i} \text{ subject to } s_i \le ST_i, \text{ for } 1 \le i \le n_p. \qquad \textbf{(3)}$$

It means that the multicast tree is constructed by two steps, i.e. a tree among proxy-senders minimizing the average RTT value weighted by the number of end-systems in each cluster and trees minimizing average RTT value in all clusters. To obtain the optimal solution of simplified problem, the followings must be taken into account: how to cluster the end-systems, how to choose the proxy-sender in each cluster, how to construct among proxy-senders, and how to construct the trees in the clusters. They are described in detail in the followings.

2.2. Clustering and Tree Constructing Algorithm

In this section, we describe clustering algorithm, selection of proxy-sender in each cluster, and the modified Dijkstra's algorithm to construct effective trees among proxy-senders and in each cluster in detail.

2.2.1. Clustering Algorithm and Proxy-sender Selection

How to cluster the multicast members and select a proxy-sender in each cluster is greatly related to the performance of the proposed algorithm. The *k-mean clustering* algorithm [9], one of the most popular techniques in digital image processing, is employed in this paper. At the same time, we have to consider the number of streams that can be supported by sender when the number of clusters is determined. In this paper, the number of clusters is determined by iterative approach as shown in Figure 5.

To find the optimal selection of proxy-sender, we have to consider the two terms in Eq. 1 simultaneously. Thus, it is difficult to select a proxy-sender in each cluster to minimize the average of RTT values since the number of possible cases is too large. During the experiment, it is observed that it is optimal that an end-system with the minimum RTT from sender and maximum number of streams is chosen as a proxy-sender, and it is very close to the optimal solution that an end-system that is the closest to the sender and can support as many streams as other end-systems is selected as a proxy-sender. In this paper, we choose the end-system closest to sender/parent proxy-sender as a proxy-sender under the assumption that all members can support overlay multicast same number of streams.

2.2.2. Tree Construction by using the Modified *Dijkstra's* Algorithm

To construct the tree to minimize the average RTT, *Dijkstra's* algorithm [8] can be employed, which finds the shortest path from a source node to all other nodes in a network. However, it can not be directly applicable to our problem since the number of streams may be greater than the number of streams that can be handled by an end-system. Furthermore, the link cost between sender and proxy-sender must be adjusted by the cluster size since the sum of RTT values in a cluster with m_i end-systems increases by $m_i \cdot \Delta RTT$ if the RTT value of proxy-sender is increased by ΔRTT. Therefore, the RTT values between sender and proxy-sender must be weighted by ${}^{1}\!/\!{}_{m_i}$ before the *Dijkstra's* algorithm is applied. Basically, the proxy-sender with the minimum weighted RTT is connected to sender or parent proxy-sender to minimize the sum of RTT values. However, it is not always the best choice. More consideration is sometimes required when two clusters are closely located as shown in Figure 3. Under the assumption that $\frac{RTT^{s}_{p_i}}{m_i} < \frac{RTT^{s}_{p_j}}{m_j}$ and the RTT between two proxy-senders is $RTT^{p_i}_{p_j}$ (It is assumed that $RTT^{p_j}_{p_i}$ is equal to $RTT^{p_i}_{p_j}$.), we consider which proxy-sender must be chosen to minimize the average RTT value. The sum of RTT values of the solid line case in Figure 3 is $m_i \cdot RTT^{s}_{p_i} + m_j \cdot \left(RTT^{s}_{p_i} + RTT^{p_i}_{p_j}\right)$ while that of the dotted line case in Figure 3 $m_j \cdot RTT^{s}_{p_j} + m_i \cdot \left(RTT^{s}_{p_j} + RTT^{p_j}_{p_i}\right)$. Now we can get the following rules based on the above two equations.

- If $m_i \cdot RTT^{s}_{p_i} + m_j \cdot \left(RTT^{s}_{p_i} + RTT^{p_i}_{p_j}\right) < m_j \cdot RTT^{s}_{p_j} + m_i \cdot \left(RTT^{s}_{p_j} + RTT^{p_j}_{p_i}\right)$, then proxy-sender p_i must be connected to the sender/parent proxy-sender.
- Otherwise proxy-sender p_j must be connected to the sender/parent proxy-sender even though $\frac{RTT^{s}_{p_i}}{m_i} < \frac{RTT^{s}_{p_j}}{m_j}$.

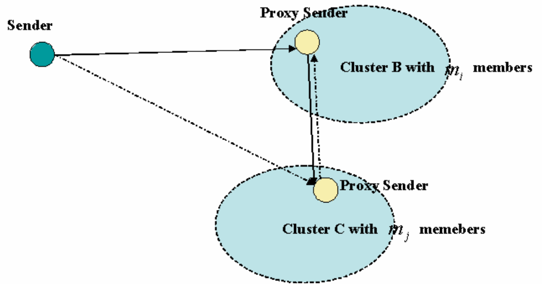


Fig. 3. The optimal selection of proxy-sender when two clusters are located closely.

Thus, the condition that proxy-sender p_j must be connected to the sender/parent proxy-sender instead of the proxy-sender p_i with the minimum weighted RTT is

$$RTT_{p_i}^{p_j} < \frac{m_i + m_j}{m_i - m_j}\left(RTT_{p_j}^{s} - RTT_{p_i}^{s}\right).$$

It can be easily shown that $\frac{m_i + m_j}{m_i - m_j}\left(RTT_{p_j}^{s} - RTT_{p_i}^{p_j}\right) > 0$ by using $\frac{RTT_{p_i}^{s}}{m_i} < \frac{RTT_{p_j}^{s}}{m_j}$ and the fact that the proxy-sender p_i has the minimum weighted RTT. In the following, the modified *Dijkstra's* algorithm is summarized in Figure 4.

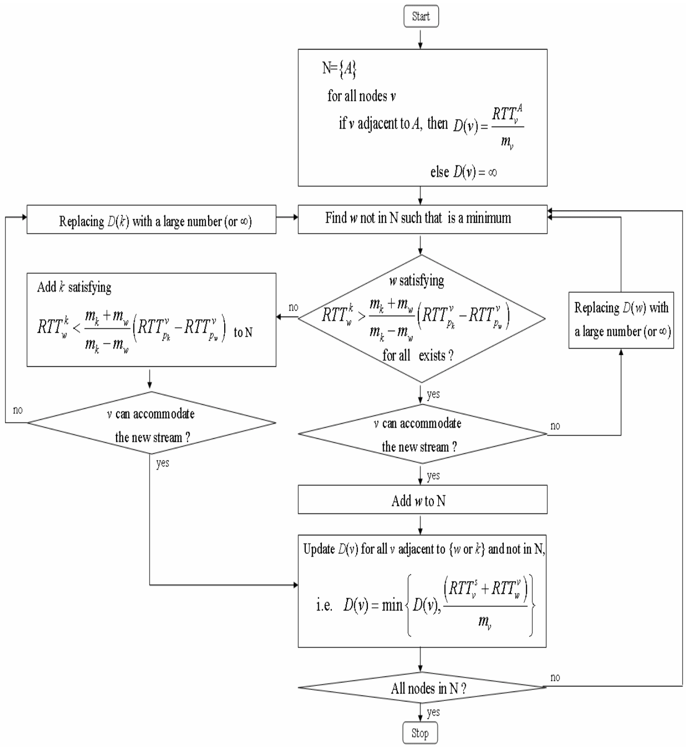

Start
N={A}
for all nodes v
if v adjacent to A, then $D(v) = \frac{RTT_v^A}{m_v}$
else $D(v) = \infty$
Find w not in N such that is a minimum
Replacing D(k) with a large number (or ∞)
w satisfying
$RTT_w^k > \frac{m_k + m_w}{m_k - m_w}\left(RTT_{P_k}^v - RTT_{P_w}^v\right)$
for all exists ?
no
Add k satisfying
$RTT_w^k < \frac{m_k + m_w}{m_k - m_w}\left(RTT_{P_k}^v - RTT_{P_w}^v\right)$ to N
Replacing D(w) with a large number (or ∞)
yes
no
v can accommodate the new stream ?
yes
v can accommodate the new stream ?
no
yes
Add w to N
Update D(v) for all v adjacent to {w or k} and not in N,
i.e. $D(v) = \min\left\{D(v), \frac{\left(RTT_v^s + RTT_w^v\right)}{m_v}\right\}$
All nodes in N ?
no
yes
Stop

Fig. 4. Flow chart of the proposed Dijkstra's algorithm: where N is the set of nodes whose least RTT path from the sender is definitely known, RTT_v^A is the RTT from A to v, m_v is the number of end-systems in the v cluster, and $D(v)$ is the weighted RTT of the path from sender to destination v that has currently the least weighted RTT.

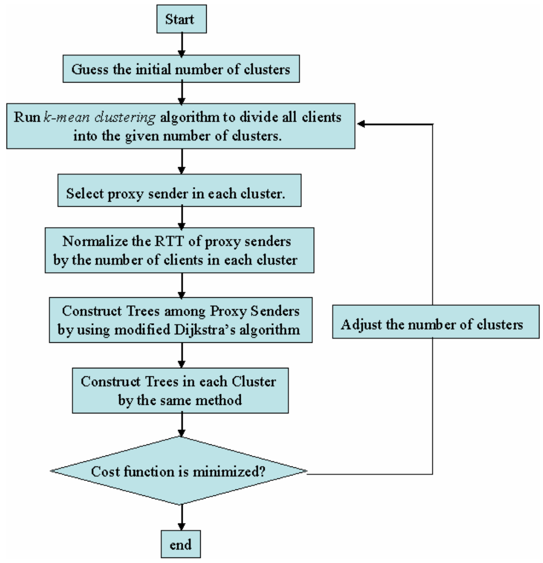
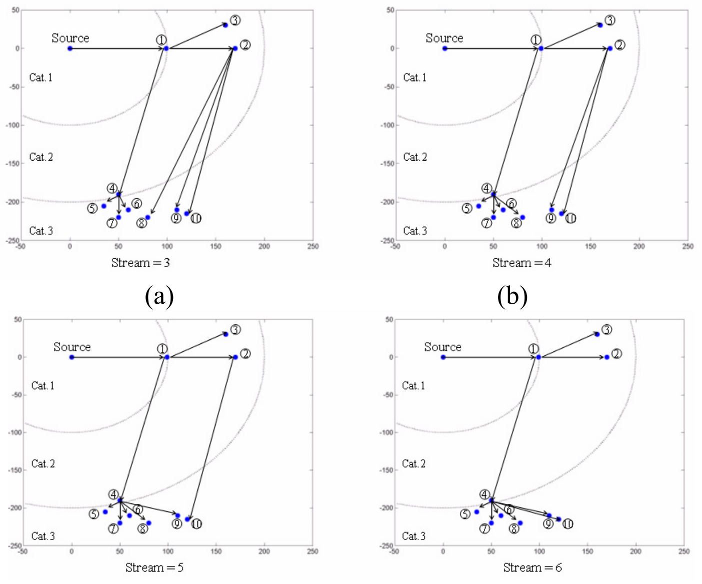

Start
Guess the initial number of clusters
Run *k-mean clustering* algorithm to divide all clients into the given number of clusters.
Select proxy sender in each cluster.
Normalize the RTT of proxy senders by the number of clients in each cluster
Construct Trees among Proxy Senders by using modified Dijkstra's algorithm
Adjust the number of clusters
Construct Trees in each Cluster by the same method
Cost function is minimized?
end

Fig. 5. Flow chart of the proposed algorithm.

By the same procedure, the effective multicast tree can be constructed in each cluster. The summarized flow chart is given in Figure 5.

3. Experimental Results

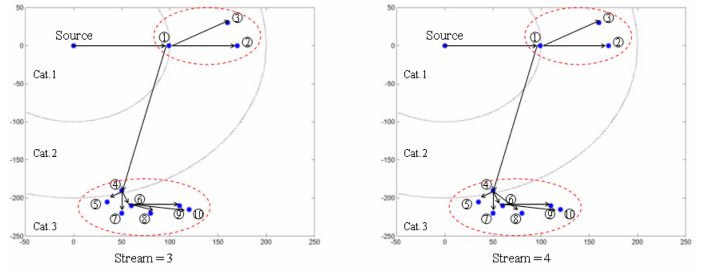
In this section, we compare the performance of the proposed algorithm in terms of tree structure, average time delay, and computational complexity. That is, average RTT value, CPU time and the number of streams assigned to each end-system are used as performance measures. For the simplicity, we assume that the members have the same computing power and network condition and thus they can support the same number of streams. However, it is still not easy to search other algorithms in order to directly compare the performance since they consider the different network conditions. In this paper, the proposed algorithm is compared with the expanded ring algorithm studied in [12] since they consider the relatively similar situations and also use average RTT as performance measure. The resulting trees of the proposed algorithm and the expanded ring algorithm are given in Figure 6, 7 and 8, and the performance comparison is provided in Figure 9 and 10. No matter how end-systems are located, the expanded ring algorithm give the same structure multicast tree shown in Figure 6. On the other hand, the proposed algorithm makes different trees according to the distribution of end-systems. When end-systems are relatively densely located and can support less than 5 streams, the proposed algorithm connect end-systems 9 and 10 to the closest other end-systems while the expanded ring algorithm connect them to the remote end-system 2 as shown in Figure 6 and 7, and thus the average RTT can be reduced. However, both algorithms give the same trees as shown in (d) of Figure 6 and 7 when end-systems can support more than 6 streams. Secondly, the difference is much more obvious when end-systems are sparsely located. That is, a stream from sender is given to end-system 4 instead of end-system 1 due to the numbers of end-systems in two clusters and the RTT between two proxy-senders. In this case, the proposed algorithm can significantly reduce the average RTT value compared to the expanded ring algorithm as shown in Figure 10. However, the required computational complexity and the number of ping commands may be greater than those of the expanded ring algorithm.



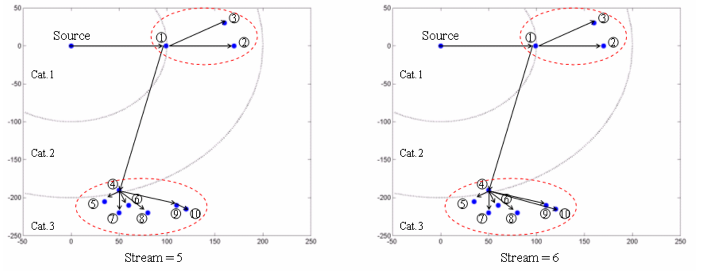
(a) (b)

(c) (d)

Fig. 6. Tree generated by expanded ring algorithm: (a) the number of streams that each end-system can support is 3, (b) the number of streams that each end-system can support is 4, (c) the number of streams that each end-system can support is 5, and (d) the number of streams that each end-system can support is 6.

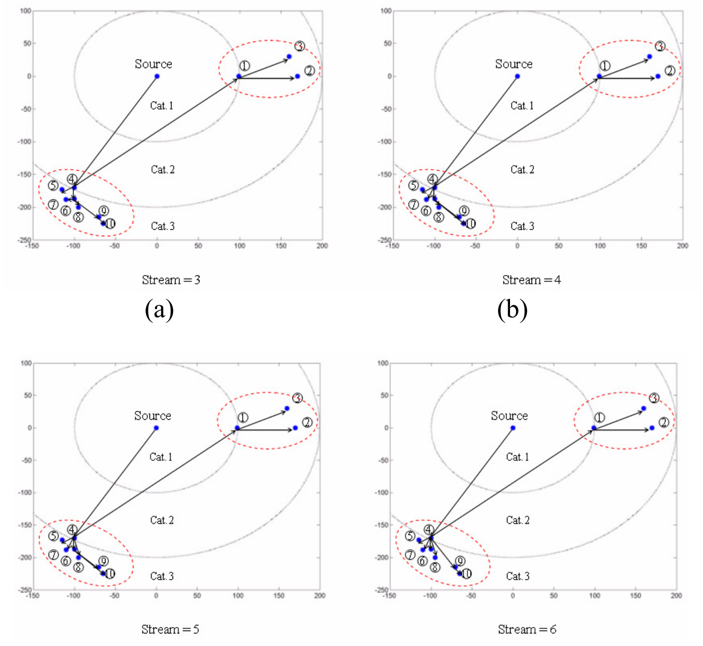


(a) (b)



(c) (d)

Fig. 7. Tree generated by the proposed algorithm when end-systems are densely located: (a) the number of streams that each end-system can support is 3, (b) the number of streams that each end-system can support is 4, (c) the number of streams that each end-system can support is 5, and (d) the number of streams that each end-system can support is 6.



(a) (b)

(c) (d)

Fig. 8. Tree generated by the proposed algorithm when end-systems are sparsely located: (a) the number of streams that each end-system can support is 3, (b) the number of streams that each end-system can support is 4, (c) the number of streams that each end-system can support is 5, and (d) the number of streams that each end-system can support is 6.

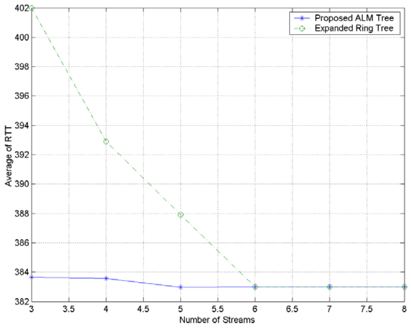


Fig. 9. Average RTT comparison between the proposed algorithm and the expanded ring algorithm when end-systems are densely located.

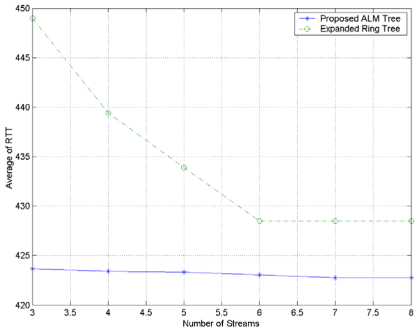


Fig. 10. Average RTT comparison between the proposed algorithm and the expanded ring algorithm when end-systems are sparsely located.

4. Conclusion

In this paper, we have presented an effective overlay multicast tree constructing algorithm minimizing the average RTT value of end-systems for real-time media delivery. Furthermore, the proposed algorithm has considered the computing power and network condition of each end-system as a control parameter to avoid the disastrous situation that loads are concentrated to several end-systems. The multicast tree has been constructed by clustering algorithm and modified Dijkstra's algorithm in two steps, i.e. a tree among proxy-senders and trees in every cluster. By the experimental results, we have showed that the proposed algorithm can provide an effective solution. For a complete solution, dynamic tree maintenance is needed when a new member joins or leaves the multicast group. Actually, it is under the current investigation.

References

1. Y. Chu, S. Rao, S. Seshan and H. Zhang, “A case for end-system multicast,” ACM SIGMETRICS, Santa Clara, June 2000.
2. S. Deering, “Host Extensions for IP Multicasting,” RFC1112, August 1989.
3. J. Liebeherr and M. Nahas, “Application-layer Multicast with Delaunay Triangulations,” IEEE GLOBECOM, 2001.
4. L. Mathy, R. Canonico, S. Simpson, and D. Hutchison, “Scalable adaptive hierarchical clustering,” IEEE Communication Letter, Vol. 6, No. 3, March 2002.
5. D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, “OVERLAY MULTICASTI: An Application level multicast infrastructure,” Proceedings of 3rd Usenix Symposium on Internet Technologies & Systefeeder (USITS 2001), San Francisco, Mar. 2001.
6. J. Park, S. J. Koh, S. G.. Kang, D. Y. Kim, “Multicast Delivery Based on Unicast and Subnet Multicast,” IEEE Communications Letters, Vol. 5 No. 4 April 2001.
7. M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, “Scribe: A large scale and decentralized application-level multicast infrastructure,” IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.
8. S. Y. Shi and J. S. Turner, “Multicast routing and bandwidth dimensioning in overlay network,” IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.
9. S. Banerjee and B. Bhattacharjee, “Scalable secure group multicast over IP multicast,” IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002.
10. J. F. Kurose and K. W. Ross, Computer Networking: A Top-down approach featuring the Internet, Addison Wesley, 2001.
11. W. K. Pratt, Digital Image Processing, Wiley Interscience, 1991.
12. C. K. Yeo, B. S. Lee and M. H. Er, “A framework for multicast video streaming over IP networks,” Journal of Network and Computer Applications, pp. 273-28, Vol. 26, 2003.