Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2162–2171
computer communications
www.elsevier.com/locate/comcom

Proxy location for minimizing delivery delay in HRM networks

Sang-Seon Byun [a,*], Chuck Yoo [b]

[a] Graduate School of Embedded Software, Korea University, Seong-buk gu, An-am dong 5th Street, Seoul, Republic of Korea
[b] Department of Computer Science and Engineering, Korea University, Seong-buk gu, An-am dong 5th Street, Seoul, Republic of Korea

Received 14 December 2005; received in revised form 10 February 2007; accepted 23 February 2007
Available online 18 April 2007

Abstract

In hierarchical reliable multicast schemes, the number of repair proxies and their locations influence the delivery delay. Low delivery delay is essential for the transmission of real time media. In this paper, we propose a method to decide optimal locations of repair proxies that minimize the mean delivery delay of all receivers in heterogeneous networks using a dynamic programming approach. Additionally, we evaluate how well the optimal proxies endure the link fluctuations. The evaluation results of our proposal in a simulation topology show that the mean delivery delay of all receivers can be reduced by about 10ms in network size of 1000 nodes. Moreover, we show that if an optimal proxy set is configured once and if an application can tolerate a mean delivery delay increase of 2.5 ms, the proxy set can sustain about 200% fluctuation of link statistics. Our method can be used by network providers in order to reduce delivery delay in their HRM network.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Hierarchical reliable multicast; Repair proxy; Dynamic programming

1. Introduction

Many hierarchical reliable multicast (HRM) protocols deploy repair proxies that perform local recovery and feedback consolidation. A repair proxy can be set up as an exclusive server [1–3] or can be designated among adequate receivers [4,5].

The performance of HRM is evaluated by (1) delivery delay, (2) bandwidth overhead due to local recovery and feedback consolidation, and (3) inter-receiver fairness [6,7,23,29]. Delivery delay is the time that is required to successfully transmit a packet from the sender to a receiver. Mean delivery delay is delivery latency of a packet between the sender and a receiver with reflection of link heterogeneity and locations of repair proxies. Inter-receiver fairness is measured to estimate the diversity of each receiver's delivery delay. All these three metrics are affected by the locations of repair proxies. For example, if a proxy is adjacent with a receiver that is susceptible to packet losses and if the proxy is robust with packet losses, the recovery and feedback traffic are limited to the proxy's domain. Additionally, packet recovery time can be reduced, and therefore better mean delivery delay and inter-receiver fairness can be obtained. [31] demonstrates that local recovery mechanisms are much more robust and efficient in recovering from single or multiple packet losses within a single round-trip time (RTT).

Related with placement of proxies, many researches have focused on minimizing bandwidth overhead caused by recovery and feedback traffic [8–10,13,30] and load balancing among proxies [9,11]. Some researches have not considered optimal placement of proxies [12,13]. Also, they assume that every link has uniform delivery delay and loss rate. Li et al. [14] suggest a method to localize proxies to minimize web distribution time using dynamic programming formulation. However, in order to reduce combinatorial complexity, the available location of a proxy is limited to some area of the web distribution tree.

Low delivery delay is an important issue of real time applications like multi-conference system [25], interactive

* Corresponding author. Tel.: +82 2 3290 3639; fax: +82 2 922 6341.
E-mail addresses: ssbyun@os.korea.ac.kr (S.-S. Byun), hxy@os.korea.ac.kr (C. Yoo).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.02.023

distributed simulations, distributed games or delivery of MPEG I-frames [22]. Besides time constraint, low delays are vital for providing high throughput with a window-based sending scheme [24], and the low delay is essential to increase responsiveness in web services. Additionally, heterogeneous loss rate and delivery delay have big impacts on the placement of proxies.

In this paper, we propose a scheme that determines optimal locations of repair proxies to minimize the mean delivery delay using a dynamic programming formulation. In addition, we verify the stability of optimal proxies under the fluctuations of link statistics.

The rest of paper is organized as follows. In Section 2, we explain the HRM model used in this paper and describe a mean delivery delay model. In Section 3, we present our dynamic programming formulation for optimal placement of proxies. In Section 4, comparison by numerical evaluation is given. Finally, concluding remarks are presented in Section 5.

2. HRM and delivery delay model

In this section, we describe the HRM and expected delivery delay model. In order to determine locations of proxies to minimize mean delivery delay, both HRM and mean delivery delay model deal with link heterogeneity and locations of proxies.

2.1. HRM model

It is assumed that the HRM model in this paper has the following characteristics (Fig. 1).

(1) The root of a multicast tree is the unique source, all leaves are receivers, and each intermediate node can be assigned to a proxy [1,2,3].

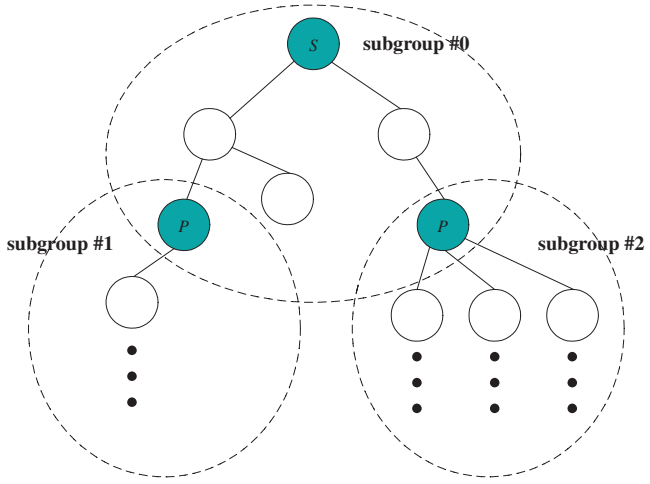


Fig. 1. HRM model. We consider a single-source multicast tree where the root is the unique source. The tree is divided into subgroups and feedback and transmissions/retransmissions are limited only between a proxy and the receivers of its subgroup.

(2) The topology of the control tree is identical to that of its underlying multicast tree (IP multicast tree), and loss probabilities and delivery delays at the links of the control tree are given. Levien et al. [16–18,27] describe a way of establishing a control tree that is identical to its underlying multicast tree and how to collect link loss statistics. Especially, Caceres and Presti [26,28] propose methods to infer per-link loss rates and delays. Due to the dynamic nature of memberships and network environments, it is hard to acquire the exact link statistics quickly in the real world. However, using our proposed method, the locations of proxies can be obtained in a few tens of seconds with the network size of 1000 nodes. Also, we verify the stability of our method under a dynamic network environment. Finally, even if topology and statistical information is not known in real life, our method may still be useful for comparison and assessment purposes.
(3) The control tree is partitioned into subtrees that form a hierarchy rooted at the source. All nodes in a subtree are combined into a subgroup, and each subgroup has a proxy located at the root of its subtree. The source always has a proxy by default. These features are deployed in [1,2,3,5,8,19].
(4) A proxy multicasts the original data to its own subgroup. Each receiver sends feedback (NACK or Immediate ACK) to its proxy when a packet loss is detected, and the proxy retransmits the lost packet to the whole subgroup. Neither flow control nor congestion control is considered. All feedback packets are delivered via an out-of-band channel, so all feedback packets are delivered safely to proxies.
(5) Feedback, transmission and retransmission are limited only between a proxy and the receivers of its subgroup and they do not reach receivers/proxies of any other subgroup. For this purpose, a new multicast address per subgroup is assigned [2], or TTL (Time to live) may be used to scope the subgroup [4]. Additionally subcasting and TTL scoping can be used simultaneously [5].

2.2. Mean delivery delay model

In this paper, we use a mean delivery delay model in Fig. 2, which reflects heterogeneity and locations of proxies. Neither flow control nor congestion control is considered. It is assumed that packet losses at each link are independent [12]. A summary of the used notations is given in Table 1.

According to [7], the number of subsequent lost packets follows the expectation of the geometric distribution. Thus if we denote the number of subsequent lost packets that are lost on the path between proxy a and receiver b as n_{ab}, the following equation holds

$$n_{ab} = \frac{1 - \prod_{i \in \psi(a,b)}(1 - pi)}{\prod_{i \in \psi(a,b)}(1 - pi)}. \quad (1)$$

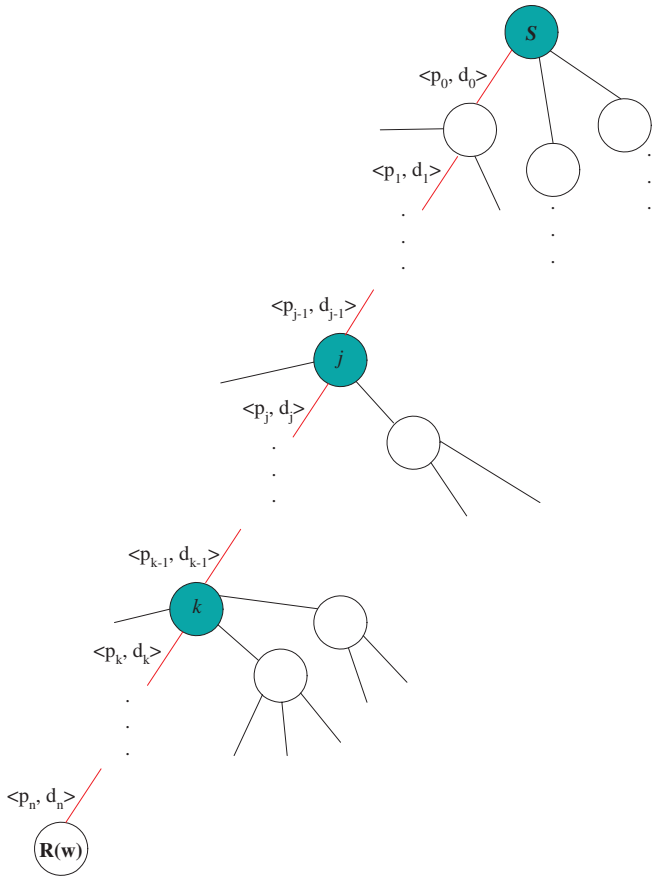


Fig. 2. Mean delivery delay model reflecting heterogeneity and locations of proxies in which <p, d> on each link means <loss rate, delivery delay>.

Table 1
Notations for expected delivery model

Value	Description
$L_{a,b}$	Summation of delivery delays on all links between node a and node b
$E(N_{a,b})$	When node a is a proxy of node b, $E(N_{a,b})$ is the expected inter-arrival time of two consecutive packets in node b. (their sequence numbers are consecutive)
$\pi(a,b)$	Set of all proxies located on the path between node a and node b.
$\theta(a,b)$	Number of elements in the set $\pi(a,b)$
$E(D_{S,R(w)})$	Mean delivery delay from sender node s and receiver node $R(w)$
$\psi(a,b)$	Set of all links between node a and node b.

In our HRM model, a receiver recognizes a packet loss by detecting packet sequence missing. Thus the required time to detect a packet loss at receiver b is written as follows, where t is inter-packet delay (gap)

$$(n_{ab}+1)\times t. \tag{2}$$

Thus $E(N_{a,b})$ can be written as follows, where $L_{a,b}$ indicates the delay that is needed by receiver b to deliver NACK or immediate ACK to proxy a through out-of-band channel

$$E(N_{a,b}) = (n_{ab}+1)\times t + L_{a,b}. \tag{3}$$

$E(D_{S,R(w)})$ can be written in two cases as follows:

2.3. In case $\pi(S,R(w)) = \{S\}$ (S is a unique proxy between S and $R(w)$)

Since there is no proxy between the source S and receiver $R(w)$, a packet dropped on the path between S and $R(w)$ can be recovered from only S. Thus $E(D_{S,R(w)})$ is expressed as Eq. (4), where $P_{S,R(w)}$ and $Rcv_{S,R(w)}$ are the probability that there is no packet loss and the delay required to recover a packet loss on the path between S and $R(w)$, respectively

$$E(D_{S,R(w)}) = P_{S,R(w)}\times L_{S,R(w)} + (1-P_{S,R(w)})\times Rcv_{S,R(w)}. \tag{4}$$

And $Rcv_{S,R(w)}$ is composed of the packet loss detection delay at $R(w)$, the delay to deliver feedback (NACK or immediate ACK) to S and retransmission delay from S. Thus

$$Rcv_{S,R(w)} = E(N_{S,R(w)}) + E(D_{S,R(w)}) \tag{5a}$$

and

$$P_{S,R(w)} = \prod_{i\in\psi(S,R(w))}(1-pi). \tag{5b}$$

Consequently $E(D_{S,R(w)})$ is

$$E(D_{S,R(w)}) = \left[\prod_{i\in\psi(S,R(w))}(1-pi)\right]L_{S,R(w)} + \left[1-\prod_{i\in\psi(S,R(w))}(1-pi)\right] \times \left(E\left(N_{S,R(w)}\right)+E\left(D_{S,R(w)}\right)\right). \tag{6}$$

By eliminating $E(D_{S,R(w)})$ at the right side of Eq. (6), we obtain

$$E(D_{S,R(w)}) = L_{S,R(w)} + \frac{1-\prod_{i\in\psi(S,R(w))}(1-pi)}{\prod_{i\in\psi(S,R(w))}(1-pi)}E\left(N_{S,R(w)}\right). \tag{7}$$

2.4. In case $\theta(S,R(w)) \geqslant 2$ and $node(j)\in\pi(S,R(w))$

In this case, $node(j)$ located on the path between S and $R(w)$ plays a role of a proxy for $R(w)$. Thus a packet dropped on the path between $node(j)$ and $R(w)$ can be recovered by $node(j)$. A packet dropped on the path between S and $node(j)$ should be recovered from S. Now the expected delivery delay between S and $R(w)$ is expressed as

time to detect a packet loss at $node(j)$
+ feedback transmission delay from $node(j)$ to S
+ $E(D_{s,R(w)})$.

Therefore if $node(j)$ is a proxy on the path between source S and receiver $R(w)$ (refer Fig. 2), $E(D_{S,R(w)})$ is expressed as follows:

$$E\left(D_{S,R(w)}\right) = P_{S,node(j)}\times\left(L_{S,node(j)} + E\left(D_{node(j),R(w)}\right)\right) + \left(1-P_{S,R(w)}\right)\times Rcv_{S,R(w)}. \tag{8}$$

And $Rcv_{S,R(w)}$ is composed of the packet loss detection delay at $node(j)$, delay to deliver feedback (NACK or imme-

diate ACK) to S and expected delivery delay required to retransmit the lost packet to $R(w)$ from S. Thus

$$Rcv_{S,R(w)} = E(N_{S,node(j)}) + E(D_{S,R(w)}). \quad (9)$$

Consequently $E(D_{S,R(w)})$ is

$$E(D_{S,R(w)}) = \left[\prod_{i\in\psi(S,node(j))}(1-pi)\right](L_{S,node(j)} + E(D_{node(j),R(w)})) + \left[1-\prod_{i\in\psi(S,node(j))}(1-pi)\right](E(N_{S,node(j)}) + E(D_{S,R(w)})). \quad (10)$$

$$E(D_{S,R(w)}) = L_{S,node(j)} + \frac{1-\prod_{i\in\psi(S,node(j))}(1-pi)}{\prod_{i\in\psi(S,node(j))}(1-pi)}E(N_{S,node(j)}) + E(D_{node(j),R(w)}). \quad (11)$$

By definition of $E(D_{S,R(w)})$, we obtain

$$L_{S,node(j)} + \frac{1-\prod_{i\in\psi(S,node(j))}(1-pi)}{\prod_{i\in\psi(S,node(j))}(1-pi)}E(N_{S,node(j)}) \equiv E(D_{S,node(j)}). \quad (12)$$

so $E(D_{S,R(w)})$ can be written as

$$E(D_{S,R(w)}) = E(D_{S,node(j)}) + E(D_{node(j),R(w)}), \quad (13)$$

and if $node(k) \in \pi(node(j),R(w))$, we obtain

$$E(D_{node(j),R(w)}) = E(D_{node(j),node(k)}) + E(D_{node(k),R(w)}). \quad (14)$$

Thus by setting $\pi(S,R) = \{S, proxy_0, proxy_1, \ldots, proxy_z\}$, we obtain a recursive form as follows:

$$E(D_{S,R}) = E(D_{S,proxy0}) + E(D_{proxy0,proxy1}) + \cdots + E(D_{proxyz-1,proxyz}) + E(D_{proxyz,R}). \quad (15)$$

So using this form, we can compute mean delivery delays of all receivers if a set of proxies is known. This mean delivery delay model is also used as an analytic model to evaluate our proxy placement method.

3. Optimal placement of proxies

As the network size grows, the ways of selecting proxies increase dramatically. So, finding optimal locations of proxies among these numerous ways becomes a combinatorial problem with large computational cost. In this paper, we deploy bottom-up dynamic programming formulation [15] to alleviate the computational cost. When selecting k proxies in the network size of m nodes, the computation of minimizing the mean delivery delay can be performed using space of $O(m^2k)$.

3.1. Dynamic programming formulation

At First a HRM tree should be converted into a binary form by import of dummy nodes in order to adapt the dynamic programming formulation. Fig. 3 shows an example of binary tree conversion.

For each node u having children u_L and u_R, for each k, $0 \leqslant k \leqslant k_{max}$, where k_{max} is the maximum number of nodes on that proxy can be placed, we can formulate the quantity $D(u, k, v)$ in four different cases with the additional notations in Table 2:

(1) If u is a leaf node:
We need not locate proxy at u. Thus k is always 0

$$D(u,k,v) = E(D_{v,u}). \quad (16)$$

If u is not a leaf node, we can compute D at u [denoted as $D(u,k,v)$] by summation of D at u_L [denoted as $D(u_L,k,v)$] and D at u_R [denoted as $D(u_R,k,v)$] since the tree is binary form (see Fig. 4).

(2) If u is the sender,
Node u is a proxy itself by default, and it is not assigned to any proxy

$$D(u,k,-) = \min_{0\leqslant k'\leqslant k-1}[D(u_L,k',u) + D(u_R,k-k'-1,u)]. \quad (17)$$

(3) If u is an intermediate node, and we put a proxy at node u

$$D(u,k,v) = \min_{0\leqslant k'\leqslant k-1}[D(u_L,k',u) + D(u_R,k-k'-1,u) + E(D_{v,u}) \times n(\mathbf{R}_u)]. \quad (18)$$

(4) If u is an intermediate node, and no proxy is located at node u

$$D(u,k,v) = \min_{0\leqslant k'\leqslant k}[D(u_L,k',v) + D(u_R,k-k',v)]. \quad (19)$$

The proof of the above dynamic programming is shown in Appendix A.

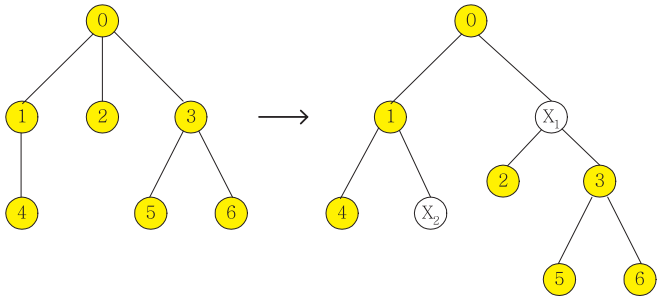


Fig. 3. Conversion of general tree into its binary form with import of dummy nodes X_1 and X_2.

Table 2
Notations for the dynamic programming formulation

Value	Description
$\mathbf{T}_u$	Set of nodes placed in subtree rooted at node u
$\mathbf{R}_u$	Set of receivers placed in subtree rooted at node u
$n(\mathbf{R}_u)$	Size of $\mathbf{R}_u$
$D(u,k,v)$	Minimum total mean delivery delay of $\mathbf{R}_u$ from node v, when k proxies are placed in $\mathbf{T}_u$. Node v is an immediate proxy of node u

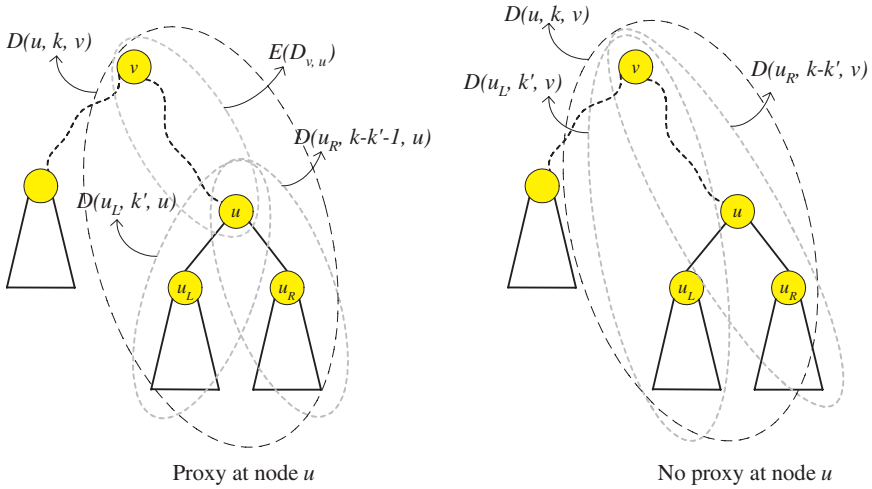


Fig. 4. Illustration of dynamic programming for $D(u,k,v)$ in a tree.

At each node, for each $0 \leqslant k' \leqslant k$ and its feasible next up-level proxy v, we have to check all possible partitions of k' to the left and right subtrees. Therefore, if the size of a tree is n, the overall time complexity is bounded by $O(nhk)$ where h is the height of a tree.

3.2. Computation procedure on example topology

Now we describe the computation procedure of $D(0,2,-)$ on a simple example topology in Fig. 3.

(1) Due to our bottom-up approach, the computations must be performed on the leaf nodes first. Thus we compute $D(4,0,0)$, $D(4,0,1)$, $D(2,0,0)$, $D(5,0,0)$, $D(5,0,X_1)$, $D(5,0,3)$, $D(6,0,0)$, $D(6,0,X_1)$, $D(6,0,3)$, $D(X_2,0,1)$ and $D(X_2,0,0)$ as follows:

$$\begin{aligned}
D(4,0,0) &= E(D_{0,4}),\\
D(4,0,1) &= E(D_{1,4}),\\
D(2,0,0) &= E(D_{0,2}),\\
D(5,0,0) &= E(D_{0,5}),\\
D(5,0,X_1) &= E(D_{X1,5}) = E(D_{0,5}),\\
D(5,0,3) &= E(D_{3,4}),\\
D(6,0,0) &= E(D_{0,6}),\\
D(6,0,X_1) &= E(D_{X1,5}) = E(D_{0,6}),\\
D(6,0,3) &= E(D_{3,6}),\\
D(X_2,0,1) &= D(X_2,0,0) = 0.
\end{aligned}$$

(2) Next we can compute D on the next up-level nodes 1 and 3 using the results obtained at step (1) as follows:

(a) $D(1,1,0)$: when a proxy is put at node 1

$$\begin{aligned}
D(1,1,0) &= D(4,0,1) + D(X_2,0,1) + E(D_{0,1}) \times 1\\
&= D(4,0,1) + E(D_{0,1}) \times 1.
\end{aligned}$$

(b) $D(1,0,0)$: when we do not locate a proxy at node 1

$$\begin{aligned}
D(1,0,0) &= D(4,0,0) + D(X_2,0,0)\\
&= D(4,0,0).
\end{aligned}$$

(c) $D(3,1,0)$ and $D(3,1,X_1)$: when we locate a proxy at node 3

$$\begin{aligned}
D(3,1,0) &= D(5,0,3) + D(6,0,3) + E(D_{0,3}) \times 2\\
D(3,1,X1) &= D(5,0,3) + D(6,0,3) + E(D_{X1,3})\\
&= D(5,0,3) + D(6,0,3) + E(D_{0,3}).
\end{aligned}$$

(d) $D(3,0,0)$ and $D(3,0,X_1)$: when we do not locate a proxy at node 3

$$\begin{aligned}
D(3,0,0) &= D(5,0,0) + D(6,0,0)\\
D(3,0,X_1) &= D(5,0,X_1) + D(6,0,X_1).
\end{aligned}$$

(3) Next we can compute D on node X_1. We cannot put any proxy on node X_1. Thus only one proxy can be located in $\mathbf{T}_{x1}$ (on node 3)

$$\begin{aligned}
D(X_1,0,0) &= D(2,0,0) + D(3,0,0)\\
D(X_1,1,0) &= D(2,0,0) + D(3,1,0).
\end{aligned}$$

(4) Finally, we can compute $D(0,2,-)$. In accordance with our assumption in HRM model (Section 2.1), there is a proxy on node *0* by default

$$D(0,2,-) = \min[D(1,1,0) + D(X_1,0,0), D(1,0,0) + D(X_1,1,0)].$$

$D(0,2-)$ can be expanded as follows:

$$D(0,2,-) = \min\begin{bmatrix}\{E(D_{0,1}) + E(D_{1,4})\} + \{E(D_{0,2})\} + \{E(D_{0,5})\} + \{E(D_{0,6})\},\\ \{E(D_{0,4})\} + \{E(D_{0,2})\} + \{E(D_{0,3}) \times 2 + E(D_{3,5}) + E(D_{3,6})\}.\end{bmatrix}$$

The first equation is the total mean delivery delay of all receivers (node 4, node 2, node 5, node 6) when proxies are located at node 0 and 1. The second one reflects the placement of proxies at node 0 and 3. Observing the expanded forms, we can verify easily that the results are same as our mean delivery delay model (Section 2.2).

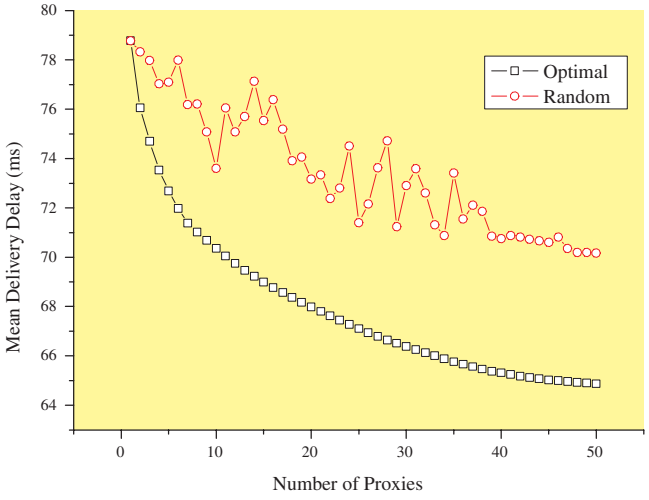
```
Output: Proxyset P_u
Proxyset Proxy_Set(node u, size k) {

        if (k == 1) {
                P_u = P_u(when k=1);
                return P_u;
        }
        if (k_uR + k_uL < k) // u has a proxy
                P_u = Proxy_Set(u_R, k_uR) ∪
                        Proxy_Set(u_L, k_uL) ∪ {u};
        else // u has no proxy
                P_u = Proxy_Set(u_R, k_uR) ∪ Proxy_Set(u_L, k_uL);

        return P_u;
}
```

Fig. 5. Pseudo code for configuring proxy set $\mathbf{P}_u$. ***Proxy_Set()*** is called recursively.

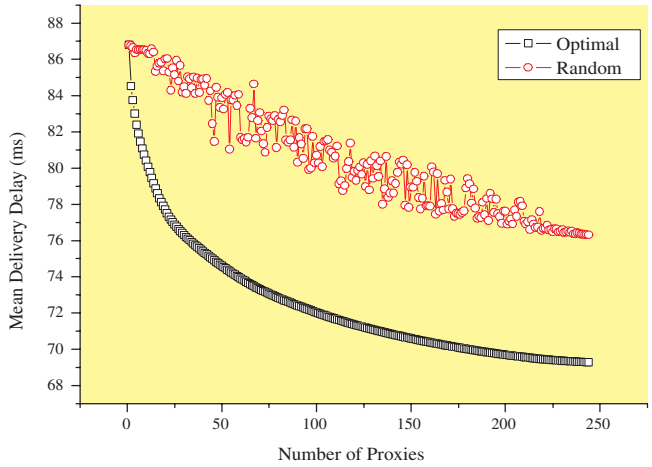
3.3. Configuration of proxy set

Our main purpose of this dynamic programming is to configure a set of proxies that minimizes the total mean delivery delay of all receivers. A proxy set can be configured during the computation process of the matrix D. In this manner, if $n(\mathbf{T}_u) = n$, we need an additional space of k^2n. However, if configuring a proxy set is preceded by the completion of computing D, it can be done with an additional space of $3kn$.

The configuring process can be implemented easily using the recurrence as in Fig. 5. Each node u, as $D(u,k,-)$ are computed, stores $k_{uR} = n(\mathbf{P}_{uR})$, $k_{uL} = n(\mathbf{P}_{uL})$, and $\mathbf{P}_u$ (when $k = 1$).



(a) n(**T**)=200, n(**R**)=72, max[n(**P**)]=50



(b) n(**T**)=1000, n(**R**)=331, max[n(**P**)]=244

Fig. 6. Mean delivery delay of all receivers with respect to the number of proxies.

4. Numerical evaluations

4.1. Mean delivery delay of all receivers

The performance of our proposal and the random placement method are compared. Simulation topologies are generated by ToGenD[20]. Delivery delays ($10 \sim 40$ ms) and loss rates of all links ($0.0001 \sim 0.1$) are assigned heterogeneously. The multicast delivery tree is constituted using the Dijikstra Algorithm [21] in order to minimize the total delivery delay of all source-receiver pairs. We assume that inter-packet gap is 25 ms.

Fig. 6 shows the arithmetic average of mean delivery delays of all receivers with respect to the number of proxies. The legend 'Optimal' indicates the min. total delivery delay obtained when we locate proxies using our proposal, the dynamic programming formulation. The 'Random' means the total delivery delay obtained when the proxies are located randomly. As expected, our proposal for the placement of proxies yields a lower mean delivery delay than the random placement method regardless of the number of proxies. The computation is completed in a few tens of seconds on our Pentium IV 3.0 GHz machine.

4.2. Stability of optimal proxies

If the fluctuation of link statistics may force frequent proxy relocations to alleviate the degradation of performance, it may difficult to deploy our method in real dynamic network environments. Thus we need to verify the stability of our method under such fluctuations. In this paper, our focus is not to verify the stability under the degradations of link statistics. If the link statistics get worse, the total mean delivery delay will increase with no regard to the proxy relocation.

For the simulation, artificial fluctuations have been adapted to every link on the simulation topology, and the difference is measured between the optimal proxy sets before and after fluctuation. A maximum fluctuation rate is given, and measurements are repeated eight times. It is assumed that the loss rate and delay of every link fluctuate on every measurement.

First, we measure the number of different proxies between the two proxy sets. The proxy size in each figure means the number of deployed proxies. Each proxy set is configured optimally reflecting the first and eighth measurement, respectively.[1] Fig. 7 plots the number of different proxy nodes between the two proxy sets when only the per-link loss rate fluctuates. Fig. 8 plots the number of different proxies when only the per-link delay fluctuates. Fig. 9 plots the number of different proxies when the loss rates and delays fluctuate simultaneously. As expected, the number of different proxies increases as the maximum fluctuation rate increases. Since the number of available node that can be selected as proxies decrease as the proxy size grows, the differences in the case of proxy size 110 are larger than that in the case of proxy size 210. Next, we observe the influence of the number of different proxies between each measured sets on the mean delivery delay.

First, we configure an optimal proxy set reflecting the first measured link statistics. Then, we adapt this proxy set to the eighth measured link statistics and compute the arithmetic average mean delivery delay of all receivers. Finally, we compute the gap between this average mean delivery delay and the measured one when the proxy set is configured optimally reflecting the eighth link statistics. By doing so, we check the stability of the proxy set under the fluctuating state.

Fig. 10 plots the gaps with respect to the maximum fluctuation rate. As shown in the graph, the gap increases as the maximum fluctuation rate increases. In these simulations, the gap is measured to be below 2.5 ms under about 200%'s fluctuation rate. When fluctuation rate is above 250%, the gap is measured to be more than 5ms. If an application does not allow a mean delay increase of above 5ms, the proxy set must be reconfigured at the fluctuation rate of 250%. Also this result says that, fewer than 200% fluctuation of the link statistics, if once a proxy set is configured optimally, and if an application can tolerate a mean delivery delay increase of 2.5 ms, the original proxy set can be maintained.

We can describe the above result more formally using the expected delivery delay model.

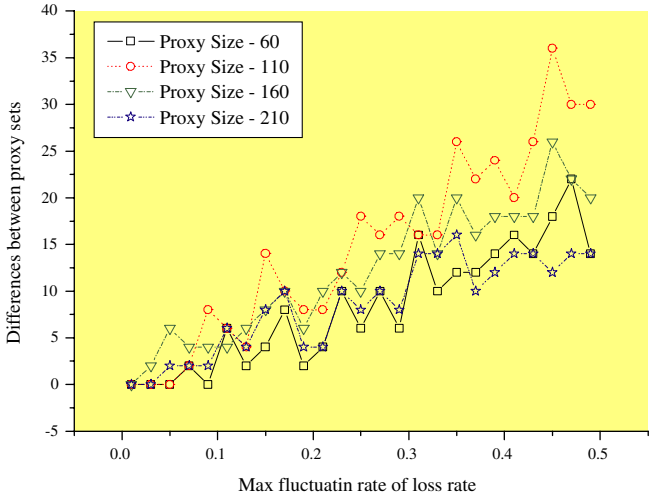


Fig. 7. The number of different proxy nodes between the proxy sets with respect to the max fluctuation rate of per-link loss rate. Each proxy set is configured according to the first and the eighth measured loss rates.

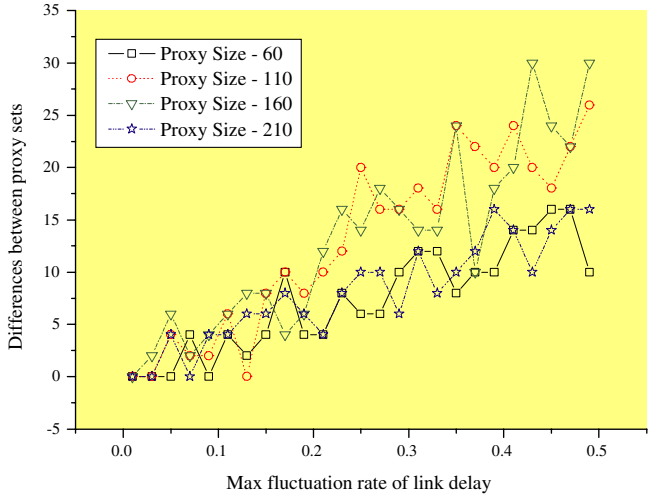


Fig. 8. The number of different proxy nodes between the proxy sets with respect to the max fluctuation rate of per-link delay. Each proxy set is configured according to the first and the eighth measured delays.

In a HRM Tree T, let S be the root of T and $\{R_1,\ldots,R_k\}$ be the set of receivers in T, that is $\mathbf{R}_s=\{R_1,\ldots,R_k\}$, and we denote the minimum total delivery delay of all receivers in T on mth measurement as $D^m(S,k,-)$ and define the value *Gap* as follows:

$$Gap = D^8(S,k,-) - D^1(S,k,-). \tag{20}$$

If we denote the expected delivery delay between node a and b and that is obtained on m-th measurement as $E^m(D_{a,b})$,

$$Gap = \sum_{Ri\in\mathbf{R}} E^8(D_{S,Ri}) - \sum_{Ri\in\mathbf{R}} E^1(D_{S,Ri}). \tag{21}$$

[1] We assume that the statistics of each link fluctuates on every inter-measurement. So if we attempt to measure 8 times, we generate fluctuation seven times. In this situation, we want to show how our proposal endures on long term fluctuations. Therefore we obtain and compare the mean delivery delay before fluctuation – means from the first measurement and after the seventh fluctuation – means from the eighth measurement.

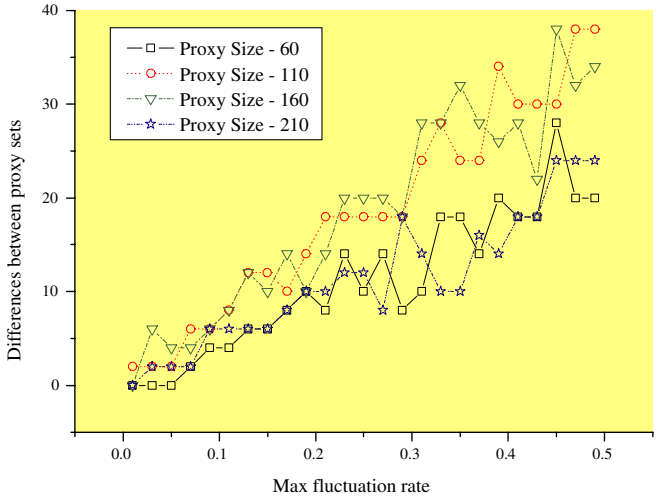
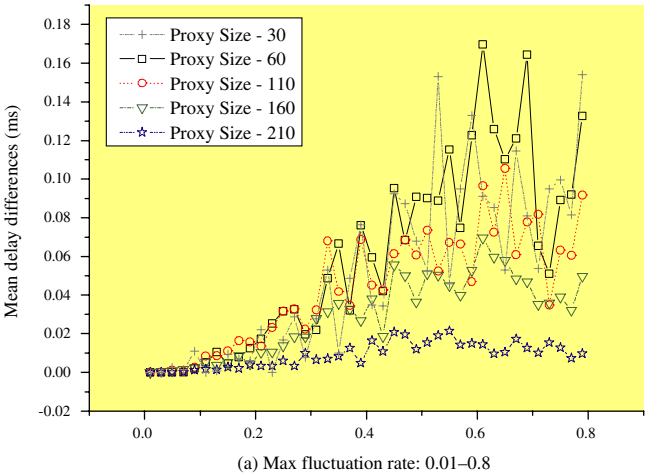
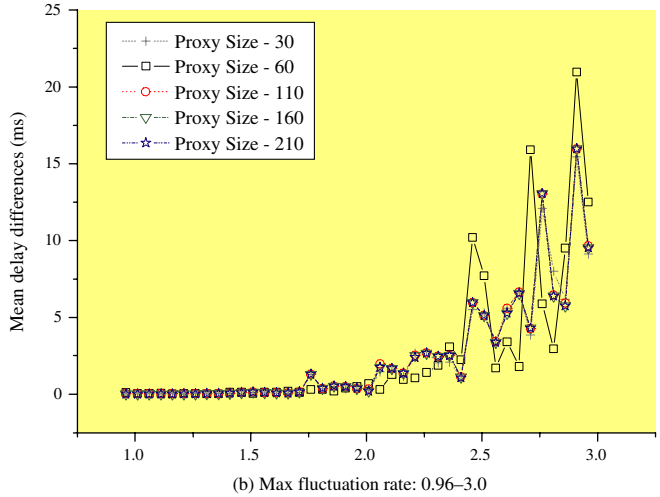


Fig. 9. The number of different proxy nodes between the proxy sets when the per-link loss rates and delays are fluctuate simultaneously.



(a) Max fluctuation rate: 0.01–0.8



(b) Max fluctuation rate: 0.96–3.0

Fig. 10. Differences of all receivers' average mean delay between the two proxy sets that are located using the first and eighth measured link statistics, respectively.

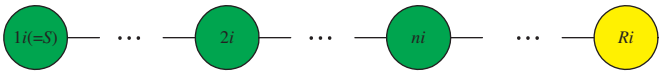


Fig. 11. Proxy set for receiver R_i.

If we let $\mathbf{P}_i(=\{1i(=S), 2i, \ldots\})$ be the set of proxies for receiver R_i as shown in Fig. 11, then by using Eq. (15), the following equation holds where $h(i) = |\mathbf{P}_i|$.

$$E^m(D_{S,Ri}) = E^m(D_{S,2i}) + E(D_{2i,3i}) + \cdots + E\left(D_{(h(i)-1)i,h(i)i}\right) + E\left(D_{h(i)i,Ri}\right). \quad (22)$$

If we let p_j^m be the packet loss rate of link j obtained at mth measurement, the expected number of retransmissions between proxy $(z-1)i$ and zi at mth measurement is

$$PR_{z-1,z}^{im} = \frac{1 - \prod_{j\in\psi((z-1)i,zi)}\left(1-p_j^m\right)}{\prod_{j\in\psi((z-1)i,zi)}\left(1-p_j^m\right)}. \quad (23)$$

Since there can be no other proxy between $(z-1)i$ and zi, using the Eq. (7), $E^m(D_{(z-1)i,zi})$ is computed as follows:

$$E^m\left(D_{(z-1)i,zi}\right) = L_{(z-1)i,zi}^m + PR_{z-1,z}^{im} E\left(N_{(z-1)i,zi}\right). \quad (24)$$

Eq. (24) can be rewritten using Eq. (3) as follows:

$$E^m\left(D_{(z-1)i,zi}\right) = L_{(z-1)i,zi}^m + PR_{z-1,z}^{im}\left[\left(PR_{z-1,z}^{im}+1\right)t + L_{(z-1)i,zi}^m\right]. \quad (25)$$

Consequently,

$$Gap = \sum_{Ri\in\mathbf{R}}\sum_{z=1}^{h(i)}\left[\left(L_{zi,(z+1)i}^{8} - L_{zi,(z+1)i}^{1}\right) + \left(PR_{z,(z+1)}^{i8} - PR_{z,(z+1)}^{i1}\right)\left(PR_{z,(z+1)}^{i8} + PR_{z,(z+1)}^{i1} + 1\right)t + PR_{z,(z+1)}^{i8} L_{zi,(z+1)i}^{i8} - PR_{z,(z+1)}^{i1} L_{zi,(z+1)i}^{i1}\right]. \quad (26)$$

Using Eq. (26), a network manager can predict the degradation amount of total delivery delay in accordance with the fluctuation of each link statistics, and he (or she) is able to determine when the proxies are to be relocated. When the above simulation topology suffers 200% of fluctuation, we can predict that average delivery delay of all receivers increase about 2.5 ms using Eq. (26).

5. Conclusions

In this paper, we propose a scheme to configure repair proxies that minimizes a mean delivery delay in heterogeneous network environments. We describe a mean delivery delay model to reflect heterogeneity and locations of proxies, and apply dynamic programming to configure an optimal proxy set in reasonable time.

Through numerical evaluations, the performance of our proposal is compared with that of a random method. Additionally, we observe that the optimally configured proxy set can be maintained without the significant degradation of the mean delay even under 250%'s fluctuation of the link statistics.

Acknowledgement

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

Appendix A

In this appendix, we describe the proof of equations in the dynamic programming

(1) If u is a leaf node:
Node u has no child. Thus $D(u,k,v)$ is the mean delivery delay from v to u. So simply (see Fig. 12)

$$D(u,k,v) = E(D_{v,u}).$$

(2) If u is a sender:
If k' proxies are placed in $\mathbf{T}_{uL}$, $\mathbf{T}_{uR}$ has k–k'–1 proxies. In addition, both u_L and u_R are assigned to their immediate proxy u. Thus (see Fig. 13)

$$D(u,k,-) = \min_{0\leqslant k'\leqslant k-1}[D(u_L,k',u) + D(u_R,k-k'-1,u)].$$

(3) In case u is an intermediate node, and we put a proxy at u:
If k' proxies are placed in $\mathbf{T}_{uL}$, $\mathbf{T}_{uR}$ has k–k'–1 proxies. Thus the minimum total delay of $\mathbf{R}_u$ is $D(u_L,k',u) + D(u_R, k\text{–}k'\text{–}1, u)$. In addition we obtain the minimum total mean delivery delay of $\mathbf{R}_u$ from v by addition of $E(D_{v,u}) \times \boldsymbol{n}(\mathbf{R}_u)$. Therefore (see Fig. 14)

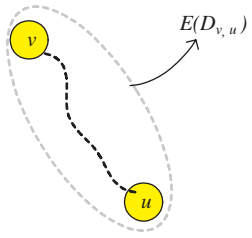


Fig. 12. Illustration of case (1).

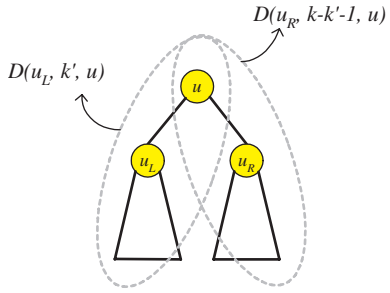


Fig. 13. Illustration of case (2).

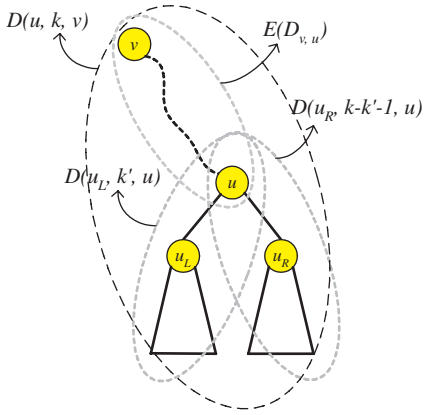


Fig. 14. Illustration of case (3).

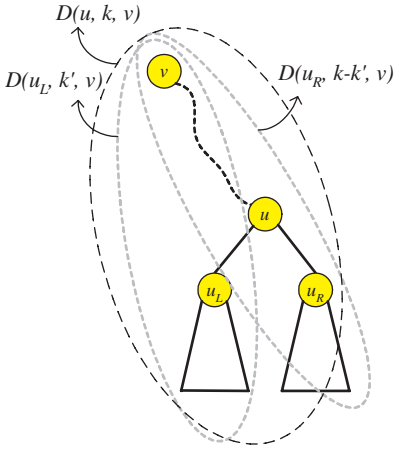


Fig. 15. Illustration of case (4).

$$D(u,k,v) = \min_{0\leqslant k'\leqslant k-1}[D(u_L,k',u) + D(u_R,k-k'-1,u) + E(D_{v,u}) \times n(\mathbf{R}_u)].$$

(4) In case u is an intermediate node, and no proxy is located at u:
If k' proxies are placed in $\mathbf{T}_{uL}$, $\mathbf{T}_{uR}$ has k–k' proxies since no proxy is placed at u. Thus the total mean delivery delay of $\mathbf{R}_u$ from v is (see Fig. 15)

$$D(u,k,v) = \min_{0\leqslant k'\leqslant k}[D(u_L,k',v) + D(u_R,k-k',v)].$$

References

[1] M. Hoffman, A generic concept of large-scale multicast, in: International Zurich Seminar on Digital Communications, Lecture Notes of Computer Science, Springer, vol. 1044, 1996, pp. 95–106.
[2] Y. Chawathe, S. McCanne, E. Brewer, RMX: Reliable multicast in heterogeneous networks, in: IEEE INFOCOM, 2000.
[3] H. Holbrook, S. Singhal, D. Cheriton, Log-based receiver-reliable multicast for distributed interactive simulation, in: ACM SIGCOMM, 1995.

[4] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, L. Zhang, A reliable multicast framework for light-weight sessions and application level framing, IEEE/ACM Trans. Network 5 (1997) 784–803.
[5] J. Lin, S.Paul, RMTP: a reliable multicast transport protocol, in: IEEE INFOCOM, 1998.
[6] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, G. Carle, How bad is reliable multicast without local recovery? in: INFOCOM, 1998.
[7] C. Maihofer, K. Rothermel, A delay analysis of tree-based reliable multicast protocols, Comput. Commun. Networks (2001).
[8] A. Markpoulou, F. Tobagi, Hierarchical reliable multicast: performance analysis and placement of proxies, in: ACM SIGCOMM, 2000.
[9] H. Lin, K. Yang, Placement of repair servers to support server-based reliable multicast, in: ICC, 2001.
[10] Z. Wan, M. Kadoch, A. Elhakeem, Performance evaluation of tree-based reliable multicast, in: ICCCN, 2003.
[11] O. Daescu, R. Jothi, B. Raghavachari, K. Sarac, Optimal placement of NAK-suppressing agents for reliable multicast: a partial deployment case, in: ACM SAC, 2004.
[12] S. Ratnasamy, S. McCanne, Scaling end-to-end multicast transports with a topologically-sensitive group formation protocol, in: ICNP, 1999.
[13] B. Li, F. Chen, L. Yin, Server replication and its placement for reliable multicast, in: ICCCN, 2000.
[14] B. Li, M. Golin, G. Italiano, X. Deng, K. Sohraby, On the optimal placement of web proxies in the internet, in: INFOCOM, 1999.
[15] A. Tamir, An O(pn^2) algorithm for the p-median and related problems on tree graphs, Oper. Res. Lett. 19 (1996) 59–64.
[16] B. Levine, S. Paul, J. Garcia-Luna-Aceves, Organizing multicast receivers deterministically by packet-loss correlation, in: ACM International Conference Multimedia, 1998.
[17] S. Casner, A. Thyagarajan, mtrace(8): Tool to Print Multicast Path Form a Source to a Receiver, UNIX manual command.
[18] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, L. Vicisano, PGM reliable transport protocol specification, in: IETF, RFC 3208, 2001.
[19] D. Li, D.R. Cheriton, OTERS (on-tree efficient recovery using subcasting): a reliable multicast protocol, in: ICNP, 1998.
[20] O. Mokryn, ToGenD – A Notre-Dame Based Topology Generator, 2000, http://www.eng.tau.ac.il/~osnaty/togend.html.
[21] S. Deryfus, An appraisal of some shortest-path algorithms, Oper. Res. 17 (1969).
[22] S. Pejhan, M. Schwartz, D. Anastassiou, Error control using retransmission schemes in multicast transport protocols for real time media, IEEE/ACM Trans. Network 4 (1996) 413–427.
[23] T. Nunome, S. Tasaka, An application-level QoS comparisons of inter-destination synchronization schemes for continuous media multicasting, in: IEEE GLOBECOM, 2003.
[24] M. Allman, D. Glover, L. Sanchez, Enhancing TCP over satellite channels using standard mechanisms, RFC 2488 (1999).
[25] T. Turletti, The INRIA videoconferencing system (IVS), ConneXions-The Interoperability Report Journal 8 (1994) 20–24.
[26] R. Caceres, N.G. Duffield, J. Horowitz, D.F. Towsley, Multicast-based inference of network-internal loss characteristics, IEEE Trans. Inform. Theory 45 (1999) 2462–2480.
[27] K. Sarac, K.C. Almeroth, Tracetree: a scalable mechanism to discover multicast tree topologies in the internet, IEEE/ACM Trans. Network 12 (2004) 795–808.
[28] F.L. Presti, N.G. Duffield, J. Horowitz, D. Towsley, Multicast-based inference of network-internal delay distributions, IEEE/ACM Trans. Network 10 (2002) 761–775.
[29] S. Byun, C. Yoo, Placement of repair proxies to improve inter-receiver delivery delay fairness in hierarchical reliable multicast networks, AICT (2005).
[30] C. Tau, T. Wang, Performance evaluation of the loss-collected retransmission scheme in reliable multicast protocol, IEE Proc. Commun. 153 (2006) 376–382.
[31] F. Xie, G. Feng, C. Siew, The impact of loss recovery on congestion control for reliable multicast, IEEE/ACM Trans. Network 14 (2006) 1323–1335.

Sang-Seon Byun received the M.S. and Ph.D. degree in computer science from Korea University, Seoul, Korea, in 2002 and 2007, respectively. He is currently a research professor at Graduate School of Embedded Software, Korea University, Seoul, Korea. His Current interests are in wireless multicast, reliable multicast and network modeling and optimizations.

Chuck Yoo received the B.S. and M.S. degree in electronic engineering from Seoul National University, Seoul, Korea and the M.S. and Ph.D. in computer science in University of Michigan. He worked as a researcher in Sun Microsystems Lab. from 1990 to 1995. He is now an professor in Department of Computer Science and Engineering, Korea University, Seoul, Korea. His research interests include high performance network, multimedia streaming, and operating systems. He served as a member of the organizing committee for NOSSDAV 2001 and JCCI 2006.