Optimizing State Allocation for Multicast Communications[1]

De-Nian Yang and Wanjiun Liao[2]
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
wjliao@cc.ee.ntu.edu.tw

***Abstract*—In this paper, we propose a new forwarding mechanism for IP multicast based on Explicit Multicast (Xcast). Compared with traditional IP multicast, our mechanism achieves a more flexible allocation of multicast forwarding states among routers. Previous works only focus on reducing the number of routers which store the forwarding states, ignoring the distribution of forwarding states among routers. The number of forwarding states stored in each router may be unbalanced. In this paper, we formulate the assignment of the routers with forwarding states as two optimization problems. The first one is to minimize the number of routers with the forwarding states in each multicast tree. The second one is to minimize the maximum number of forwarding states stored in a router. We design several algorithms for both problems. We prove that our algorithms can find the optimal solution to the first problem. By simulation, we also show that our algorithms perform well in the second problem. Moreover, we prove that assigning all branching routers as the only routers with the forwarding states is a special case in our mechanism. Due to higher flexibility, we show that our mechanism use fewer forwarding states, and the distribution of forwarding states is more balanced.**

Keywords-multicast; explicit multicast; Xcast; forwarding state

I. INTRODUCTION

Multicast is an efficient way for point-to-multipoint and multipoint–to-multipoint communications [1]. Traditional multicast schemes are based on the host group model [2] and the multicast routing protocols [3-6]. This approach associates a multicast group with a class-D IP address. Each group uses the address as the destination address of data packets. In order to guarantee the global uniqueness of each class-D address, multicast addresses have to be allocated dynamically [7]. Different from the traditional scheme, source-specific multicast [8] treats a one-to-many connection as a multicast channel. Each multicast channel is associated with a channel identifier which includes the sender's address and a class-D address. The class-D address is allocated by the sender. Therefore, it is not required to be globally unique.

Both the traditional multicast scheme and source-specific multicast use a shortest path tree to carry multicast data. The routing of the shortest path tree is the union of the shortest paths from the root to all receivers or from all receivers to the root. For point-to-multipoint communication, the root is the sender, and the tree is called a source-based tree. For multipoint-to-multipoint communication, the root is a router which is called the core in CBT [5] or the RP in PIM-SM [6], and the tree is called a shared tree. Each sender first sends the data to the root, and then the root relays the data to all receivers.

For traditional multicast schemes or source-specific multicast, each router in a shortest path tree has to maintain a forwarding state for the group or channel. The state specifies the adjacent routers which are in the multicast tree, and the ID of the forwarding state is a group address or a channel identifier. Multiple forwarding states can be aggregated into one state if their IDs are contiguous and their next-hop routers are the same. The ID of the aggregating state is the common prefix of the IDs of the aggregated forwarding states. Compared with unicast forwarding states, it is more difficult to aggregate multicast forwarding states [9-20]. The reason is that the ID of a unicast forwarding state is the destination IP address of data packets, and the destination IP address is allocated according to its geographical location. Receivers with the same prefix tend to reside in the same geographical area. For a router outside the area, the forwarding state corresponding to these addresses can be aggregated since the next-hop router in the forwarding states tends to be the same. However, for multicast forwarding states, the class-D addresses are allocated dynamically or by the sender. Besides, the parent router and the child routers of the groups with contiguous IDs may be totally different. Therefore, it is more difficult to aggregate multicast forwarding states. As there are a larger number of multicast groups or channels, routers may not have enough memory to maintain all forwarding states [9]. Moreover, a router may take a long time to find the forwarding state when it receives a data packet.

Several approaches [9-20] have been proposed to reduce the number of multicast forwarding states stored in a router. The first approach [9-12] uses a single multicast tree to deliver data of multiple multicast groups with similar receivers. A receiver may receive undesired data from a multicast group in which it does not join. Hence, the multicast tree has to be chosen carefully in order to reduce the amount of undesired data. In the second approach [13-18], only the branching routers of a multicast tree maintain the forwarding states. A branching router is a router which connects to at least three adjacent routers in the multicast tree. A multicast packet is not

[1] This work was supported in part by the MOE program for Promoting Academic Excellence of Universities under Grant number 89E-FA06-2-4-7, and in part by the National Science Council, Taiwan, under Grant Number NSC92-2213-E-002-064.
[2] This author is also with the Graduate Institute of Communication Engineering, National Taiwan University.

0-7803-8355-9/04/$20.00 ©2004 IEEE.

duplicated on the path from a branching router to its nearest downstream branching router. Therefore, it is sent via unicast between these two routers, and intermediate routers on the path maintain no forwarding state of the multicast tree.

The third approach, Explicit Multicast (Xcast) [19], uses the unicast forwarding table to forward multicast data. It does not require any router in a multicast tree to maintain a multicast forwarding state. It includes the addresses of all receivers of a multicast group in the header of each data packet. When a router in the tree receives a data packet, it first checks the next-hop router of each receiver, duplicates the packet if necessary, and then sends each packet to a next-hop router. The header of each duplicated packet contains only the receivers in the sub-tree rooted at the next-hop router. Although multicast routing protocols and multicast forwarding states are not required this approach is not suitable for groups with many receivers since the size of the header and the packet processing delay in a router grows as the number of receivers increases [19]. Therefore, Xcast is scalable in terms of the number of small groups but is not scalable in terms of the number of receivers in a group. On the contrary, traditional IP multicast is scalable in terms of the number of receivers in a group but is not scalable in terms of the number of small groups. Therefore, it is believed that both mechanisms are orthogonal and are used for different scenarios [19]. In reality, however, the number of receivers in a group may vary with time since a host can join or leave a group at any time. In other words, a mechanism chosen by a group initially may not be suitable later. Switching to another mechanism may cause service disruption, especially when the shortest path trees used by the two mechanisms are different[3]. Therefore, we need a multicast forwarding scheme scalable in terms of both the number of receivers and the number of groups.

Previous works only focus on reducing the number of forwarding states in all routers. However, so far, as we know, there is no related work on the distribution of forwarding states among different routers. Previous research [15] analyzed the distribution of multicast forwarding states and showed that the forwarding states were concentrated in backbone routers since they are used by more multicast trees. Moreover, they found that routers with larger degree, i.e., more adjacent routers or hosts, tend to have more forwarding states. It is reasonable since they are more likely to act as branching routers of a multicast tree. Therefore, the distribution of forwarding states among routers is not balanced. Some routers may not have enough memory to store the forwarding states, but others are under-utilized and can store more forwarding states.

In this paper, we propose a new multicast forwarding mechanism with resource optimization using Xcast. When a group has only a few receivers, no router in the tree maintains the forwarding state. As the number of receiver grows, some routers are chosen dynamically to store the forwarding states. As the number of receiver decreases, some of these routers abandon the forwarding states. For each downstream interface, the forwarding state records the addresses of a set of downstream routers that also store the forwarding states of the group. When a router with the forwarding state receives a data packet, it duplicates the packet to each downstream interface and substitutes the destination addresses in the header with the addresses of the downstream routers stored in the forwarding state. The router then delivers the packet via Xcast to the downstream routers. Fig. 1 is an example of our mechanism. Node 1 is the root of the multicast tree. Assume nodes 1, 4, 5, and 6 have the forwarding states of this tree. Fig. 1 also lists the downstream routers stored in the forwarding state of each router. Not all branching routers have to store the forwarding states. Our mechanism is orthogonal to the first approach which uses a single multicast tree to carry data of multiple groups and can be integrated together.

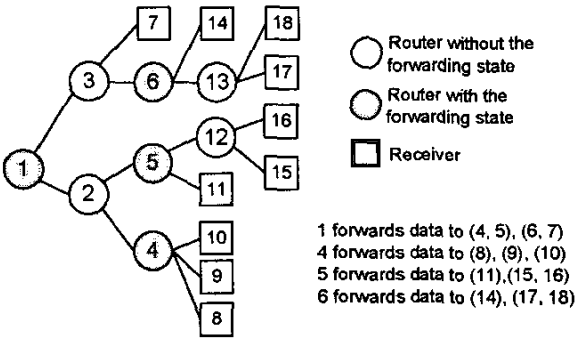


Figure 1. Example of our mechansim. The nodes within a bracket are the destinations of a Xcast datagram.

Compared with the second approach, our mechanism can assign the forwarding states more flexibly among routers. Since not all branching routers have to store the forwarding states. Moreover, non-branching routers can also maintain the forwarding states. In order to assign the forwarding states efficiently, we formulate two optimization problems. For each multicast tree, the first problem, denoted P1, is to minimize the number of routers storing the forwarding states. The second problem, denoted P2, is to minimize the maximum number of forwarding states stored in a router. We design several algorithms and conduct simulation for both problems. We also discuss related issues such as the implementation of our algorithms and the dynamic group membership.

The rest of this paper is summarized as follows. In section II, we address the problem description and definition. In section III, we present the algorithms to minimize the number of routers with forwarding states in each multicast tree. We prove that our algorithms can find the optimal solution. In section IV, we formulate the problem which minimizes the maximum number of forwarding states maintained in a router as an integer linear programming (ILP) problem and present our algorithms. In section V, our numerical results are shown. In section VI, several related issues are discussed. Finally, we conclude this paper in section VII.

II. Problem Description

In this paper, the network is modeled as a connected directed graph $G(V, A)$ where V and A are the set of vertices and arcs. Each vertex is either a host or a multicast router. Each

[3] Xcast uses a forward shortest path tree, and some traditional IP multicast routing protocols such as PIM-SM and CBT use a reverse shortest path tree. The routing of the two trees may be different if the network is asymmetric.

arc corresponds to a point-to-point link. A multi-access link can be represented as multiple arcs. A multicast tree is a forward shortest path tree. Data are delivered unidirectionally from the root of the tree to each receiver. For point-to-multipoint communication, the root is the sender. For multipoint-to-multipoint communication, the root is a relay node, like an RP in PIM-SM or a session relaying server in EXPRESS [21]. We assume each receiver of a group is a host connected to a Designated Router (DR). Therefore, each receiver must be a leaf node of the multicast tree, and all leaf nodes of a multicast tree are the receivers. For each multicast tree, a vertex m is upstream to another vertex n if m is on the shortest path from the root to n. In this case, n is also regarded downstream to m. In this paper, vertex and node are used interchangeably.

For each multicast group, a set of multicast routers is selected to maintain the forwarding states of the group. Multicast data are sent between these routers via Xcast. For each group, a multicast router d storing the forwarding state of a group is a *state node* of the multicast tree. The nearest state node u which is upstream to d is the *upstream state node* of d. In this case, d is a *downstream state node* of u. All intermediate nodes on the path from u to d are *stateless nodes*. Each receiver is a *downstream receiver* of its upstream state node. A state node may have more than one downstream state node and downstream receiver from each interface. In Fig. 1, for example, nodes 1, 4, 5, and 6 are state nodes of the multicast tree. Nodes 4 and 5 are the downstream state nodes of node 1 from the downstream interface to node 2. Node 15 and 16 are the downstream receivers of node 5 from the interface of to node 12. Node 1 is the upstream state node of node 6.

When the number of downstream state nodes and downstream receivers from an interface increases, the header of each data packet contains more destination addresses. Each stateless node has to look up more addresses in the unicast forwarding table. For each state node, therefore, we regard the maximum number of destinations from each downstream interface as a constraint. A *destination* is either a downstream state node or a downstream receiver. In other words, from each downstream interface, a state node can have at most δ destinations, $\delta \geq 1$. The root is a state node. Each leaf node, i.e. receiver, of a multicast tree is not a state node since it does not deliver data to any other node. The input parameters of a state assignment problem are listed as follows.

T the set of multicast trees;

t a multicast tree, $t \in T$

p_m^t the parent node of m in multicast tree t;

C_m^t the set of child nodes of m in multicast tree t;

r_t the root of multicast tree t;

R_t the set of receivers in multicast tree t.

The decision variables are listed as follows.

σ_m^t a binary variable; node m is a state node in t if $\sigma_m^t = 1$; otherwise, $\sigma_m^t = 0$.

u_m^t the upstream state node of m in multicast tree t;

D_m^t the set of destinations of m from all downstream interfaces in multicast tree t;

$D_{m,n}^t$ the set of destinations of m from the downstream interface to n in multicast tree t;

$\tau_m^t(j)$ the total number of state nodes in the sub-tree rooted at node m if p_m^t has j destinations from the downstream interface to m, $1 \leq j \leq \delta$;

$v_{\min}^t$ the minimum number of state nodes in multicast tree t; the objective value of P1;

$v_{\max}$ the maximum number of forwarding states maintained in a router, $0 \leq v_{\max} \leq |T|$; the objective value of P2.

III. Minimizing the Number of State Nodes in Each Multicast Tree

In this section, we propose two algorithms which can find the optimal solution, i.e., the minimum number of state nodes in a multicast tree. The first one is a dynamic programming algorithm. It first finds the minimum number of state nodes in a multicast tree from the leaves to the root and then assigns the state nodes from the root to the leaves. The second one is a distributed greedy algorithm. Each state node independently determines if it can remove its forwarding state or move the forwarding state to its parent node. The advantage of the first algorithm is that it can find the optimal assignment rapidly. The advantage of the second algorithm is that it can be implemented as a distributed asynchronous protocol.

A. Dynamic programming algorithm

Fig. 2 is the dynamic programming algorithm, which is denoted P1_dp. In step 2, it first calculates $\tau_m^t(j)$ from the corresponding values of its child nodes $\tau_n^t(j)$, $n \in C_m^t$. Variable $\tau_m^t(j)$ is chosen in the way such that the number of state nodes in the sub-tree rooted at m is minimized. For $j = 1$ and $|C_m^t| > 1$, $\tau_m^t(j)$ represents the case that m is a state node. For other scenarios, $\tau_m^t(j)$ represent the case the m is a stateless node. The root is not considered since we assume it must be a state node. After the minimum number of state nodes in the multicast tree is obtained, the algorithm then assigns the state nodes of the multicast tree. In step 3, j_m^* is the number of destinations of p_m^t from the downstream interface to m in the assignment obtained from our algorithm. Table I is an example of the algorithm with the multicast tree in Fig. 1. The resultant assignment of state nodes is the same as Fig. 1. The following lemma and theorems prove that the algorithm can find the optimal solution. Please note that the optimal assignment may not be unique.

Lemma 3.1 *There exists an optimal solution such that* $\tau_m^t(j_{m,opt}) = \tau_{m,opt}^t(j_{m,opt})$ *holds for node m,* $\forall m \in V_t - \{r_t\}$, *where*

Given: a tree t with node $V_t \subseteq V$ and arc $A_t \subseteq A$, δ.

Find: $v^t_{\min}$ and σ^t_m, $m \in V_t$.

Algorithm:

1. Initialization.

   $$\tau^t_m(j) \leftarrow \begin{cases} 0, & j = 1 \\ \infty, & 1 < j \le \delta \end{cases}, \; m \in R_t.$$

   Number all nodes from node 1 to $|V_t|$ by the breadth first search algorithm.

2. Find the minimum number of state nodes.

   for m from $|V_t|$ to 2, $m \notin R_t$,

   $$\tau^t_m(1) \leftarrow \begin{cases} \tau^t_n(1), \text{ if } |C^t_m| = 1, n \in C^t_m \\ 1 + \sum_{n \in C^t_m} \min_{1 \le j \le \delta} \{\tau^t_n(j)\}, \text{ if } |C^t_m| > 1 \end{cases};$$

   for j_m from 2 to δ,

   if $\left\{ j_n : n \in C^t_m, 1 \le j_n \le \delta, \sum_{n \in C^t_m} j_n = j_m \right\} = \phi$, then

   $$\tau^t_m(j_m) \leftarrow \infty;$$

   else

   $$\tau^t_m(j_m) \leftarrow \min_{\{j_n : n \in C^t_m, 1 \le j_n \le \delta\}} \left\{ \sum_{n \in C^t_m} \tau^t_n(j_n) \middle| \sum_{n \in C^t_m} j_n = j_m \right\};$$

   end if;

   end for;

end for;

$$v^t_{\min} \leftarrow 1 + \sum_{n \in C^t_r} \min_{1 \le j_n \le \delta} \{\tau^t_n(j_n)\}.$$

3. Find the optimal assignment of state nodes.

   $$j^*_n \leftarrow \arg \min_{1 \le j_n \le \delta} \{\tau^t_n(j_n)\}, n \in C^t_r;$$

   for m from 2 to $|V_t|$, $m \notin R_t$,

   if $|C^t_m| = 1$ and $j^*_m = 1$, then

   $$\sigma^t_m \leftarrow 0, \; j^*_n \leftarrow 1, n \in C^t_m;$$

   else if $|C^t_m| > 1$ and $j^*_m = 1$, then

   $$\sigma^t_m \leftarrow 1, \; j^*_n \leftarrow \arg \min_{1 \le j_n \le \delta} \{\tau^t_n(j_n)\}, n \in C^t_m;$$

   else

   $$\sigma^t_m \leftarrow 0,$$

   $$\{j^*_n\} \leftarrow \arg \min_{\{j_n : n \in C^t_m, 1 \le j_n \le \delta\}} \left\{ \sum_{n \in C^t_m} \tau^t_n(j_n) \middle| \sum_{n \in C^t_m} j_n = j^*_m \right\};$$

   end if;

end for.

Figure 2. The dynamic programming algorithm (P1_dp).

$j_{m,opt}$ is the number of destinations of p^t_m from the downstream interface to m in the optimal solution, and $\tau^t_{m,opt}(j_{m,opt})$ is the total number of state nodes in the sub-tree rooted at node m if p^t_m has $j_{m,opt}$ destinations from the downstream interface to m in the optimal solution

TABLE I. AN EXAMPLE OF THE DYNAMIC PROGRAMMING ALGORITHM WITH $\delta = 2$

	m						
	2	3	4	5	6	12	13
$\tau^t_m(1)$	3	2	1	1	1	1	1
$\tau^t_m(2)$	2	1	∞	1	1	0	0

Proof: The lemma is proved by induction.

1. For each node n whose child nodes are all receivers, $j_{n,opt}$ must be equal to 1 or $|C^t_n|$. If $|C^t_n| = 1$, $j_{n,opt} = 1$ and $\tau^t_{n,opt}(j_{n,opt}) = \tau^t_n(j_{n,opt}) = 0$. If $|C^t_n| > \delta$, $j_{n,opt} = 1$ and $\tau^t_{n,opt}(j_{n,opt}) = \tau^t_n(j_{n,opt}) = 1$. If $1 < |C^t_n| \le \delta$ and $j_{n,opt} = 1$, $\tau^t_{n,opt}(j_{n,opt}) = \tau^t_n(j_{n,opt}) = 1$. If $1 < |C^t_n| \le \delta$ and $j_{n,opt} = |C^t_n|$, $\tau^t_{n,opt}(j_{n,opt}) = \tau^t_n(j_{n,opt}) = 0$.

2. For each node m whose child nodes are receivers or the nodes in step 1, if $\tau^t_m(j_{m,opt}) > \tau^t_{m,opt}(j_{m,opt})$, node m has a child node n with $\tau^t_n(j_{n,opt}) > \tau^t_{n,opt}(j_{n,opt})$; it contradicts step 1. Inequality $\tau^t_m(j_{m,opt}) < \tau^t_{m,opt}(j_{m,opt})$ does not hold since it implies $\tau^t_{m,opt}(j_{m,opt})$ is not an optimal solution. Therefore, $\tau^t_m(j_{m,opt}) = \tau^t_{m,opt}(j_{m,opt})$.

3. For all the other node m which is not the root or leaves, $\tau^t_m(j_{m,opt}) = \tau^t_{m,opt}(j_{m,opt})$ can be proved in a way similar to step 2. □

Theorem 3.2 *The above dynamic programming algorithm can find the minimum number of state nodes in the multicast tree and an optimal assignment of state nodes.*

Proof: The minimum number of state nodes in the multicast tree t is as follows.

$$v^t_{\min} = 1 + \sum_{n \in C^t_r} \tau^t_{n,opt}(j_{n,opt}).$$

With Lemma 1,

$$v^t_{\min} = 1 + \sum_{n \in C^t_r} \tau^t_n(j_{n,opt}).$$

Moreover, the following equation must hold, or $v^t_{\min}$ is not the optimal solution.

$$j_{n,opt} = \arg \min_{1 \le j_n \le \delta} \tau^t_n(j_n), n \in C^t_r.$$

The assignment of state nodes generated by our dynamic programming algorithm is feasible, and the total number of state nodes is equal to the optimal value. Therefore, the assignment is an optimal assignment. □

Corollary 3.3 *If $\delta = 1$, the root and the branching nodes, i.e., the nodes with more than one child node, are the only state nodes in the optimal solution.*

Proof: In the optimal solution, if any branching node does not have a forwarding state, its upstream state node must have

more than one destination from the downstream interface. It implies the solution is infeasible. For any non-branching node, if it has a forwarding state, the state can be removed. It implies the solution is not optimal.
□

From this corollary, it shows that the assignment that uses only the branching nodes to store forwarding states is a special case of our mechanism.

Corollary 3.4 *Given two problems P_1 and P_2 with the same tree but different δ, say, δ_1 and δ_2, if $\delta_1 \geq \delta_2$, the optimal solutions corresponding to δ_1 and δ_2, say, ν_1 and ν_2, must satisfy $\nu_1 \leq \nu_2$.*

Proof: Since ν_2 is a feasible solution to P1, and both problems are the minimization problems, ν_2 provides an upper bound to ν_1. Therefore, $\nu_1 \leq \nu_2$ must hold. □

Theorem 3.5 *The time complexity is* $O(|V_t| \times 4^{\lceil(\delta-1)/2\rceil})$.

Proof: In step 2, for each node, the number of elements in set $\left\{ j_n : n \in C'_m, 1 \leq j_n \leq \delta, \sum_{n \in C'_m} j_n = j \right\}$ is $\binom{\delta-1}{|C'_m|-1} = O\left(\binom{\delta-1}{\lceil(\delta-1)/2\rceil}\right)$ $= O\left(4^{\lceil(\delta-1)/2\rceil}\right)$. Therefore, the time complexity is $O(|V_t| \times 4^{\lceil(\delta-1)/2\rceil})$. □

In section V, we show that a small δ is sufficient to obtain good solutions. Therefore, the time complexity is dominated by $|V_t|$.

B. Distributed greedy algorithm

Although the above dynamic programming algorithm can find the optimal assignment rapidly, it is not suitable to be implemented as a protocol since it induces large overhead when a receiver joins or leaves a multicast tree. Each node on the path from the root to the receiver has to update $\tau'_m(j)$, and some stateless nodes have to cache the information. In this section, we propose a distributed greedy algorithm, which is denoted P1_greedy. The algorithm is more suitable to be implemented as a protocol. At any instant, each state node independently checks if it can remove the forwarding state or move the state to its parent node. The algorithm is a greedy algorithm because the former operation reduces the number of state nodes, and the latter operation makes the allocation of state nodes more compact such that more state nodes can become stateless later. The algorithm stops when all state nodes can no longer perform the above two operations. In this algorithm, each node does not need to know the topology of the whole multicast tree. It has to know only the identities of its upstream state node, parent node, child nodes, and destinations from all downstream interfaces. The operations of different state nodes are asynchronous. We assume that at most one state node removes or moves its forwarding state at any instant.

Given: a tree t with node $V_t \subseteq V$ and arc $A_t \subseteq A$, δ.

Find: $\nu'_{\min}$, and σ'_m, $m \in V_t$.

Algorithm:

1. Initialization.
   $\sigma'_m \leftarrow 1, x'_m \leftarrow 1, m \in V_t - R_t - \{r_t\}$;
   $\sigma'_r \leftarrow 1, x'_r \leftarrow 0$;
   $\sigma'_m \leftarrow 0, x'_m \leftarrow 0, m \in R_t$.
2. While $\exists m \in V_t - R_t - \{r_t\}$ such that $x'_m = 1$,
   if $|D'_{u_m,m}| + |D'_m| - 1 \leq \delta$, then
   $\sigma'_m \leftarrow 0$, $x'_m \leftarrow 0$;
   $x'_n \leftarrow 1, n \in D'_{u_m,m} \cup \{u'_m\}$;
   else if $\sigma'_{p'_m} = 0$ and $|d'_m| \leq \delta$, then
   $\sigma'_m \leftarrow 0$, $x'_m \leftarrow 0$; $\sigma'_{p'_m} \leftarrow 1$, $x'_{p'_m} \leftarrow 1$;
   $x'_n \leftarrow 1, n \in D'_{p'_m} \cup D'_{u'_{p'_m}, p'_m} \cup \{u'_{p'_m}\}$;
   else $x'_m \leftarrow 0$;
   end if;
   end while.
3. $\nu'_{\min} \leftarrow \sum_{m \in V_t} \sigma'_m$.

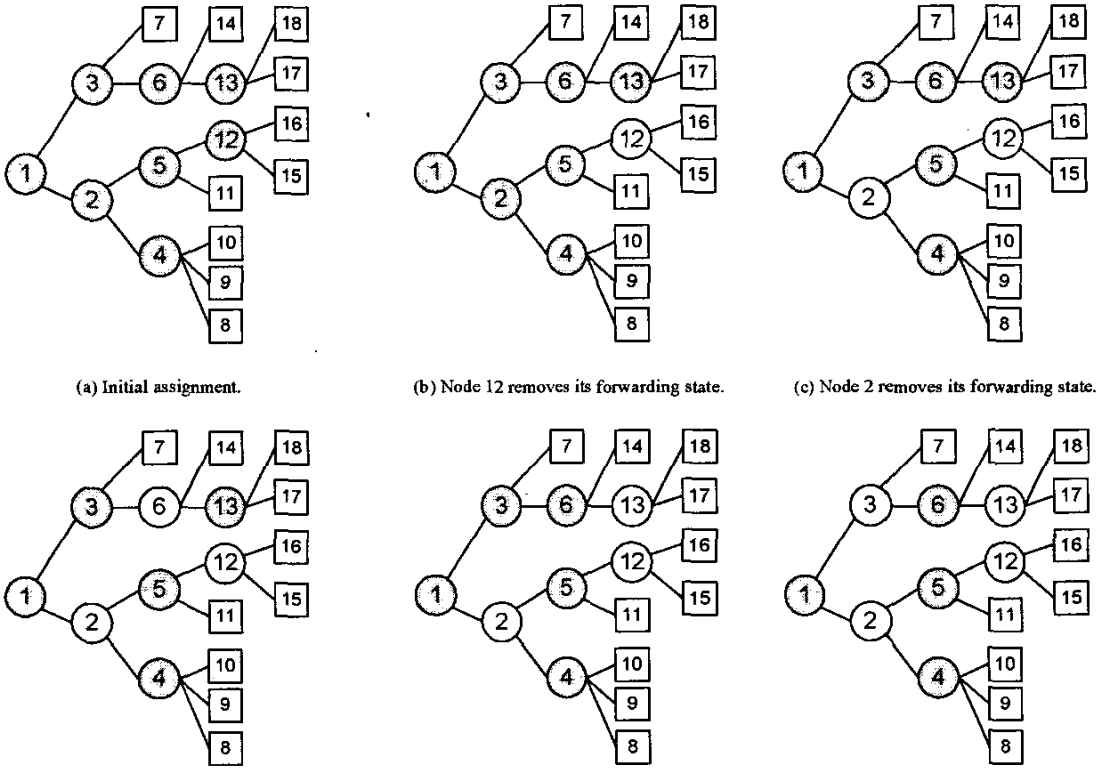
Figure 3. The distributed greedy algorithm to minimize the number of state nodes in a multicast tree (P1_greedy).

Fig. 3 is the distributed greedy algorithm. The auxiliary variable x'_m is a binary variable to represent whether node m has to check if it can remove or move its forwarding state. The algorithm stops when $x'_m = 0$, $\forall m \in V_t$. Fig. 4 is an example using the multicast tree in Fig. 1. It shows that only removing the forwarding states can not find the optimal solution. In Fig. 4 (d), nodes 3 and 13 cannot remove its forwarding states. However, after node 13 moves its forwarding state to node 6, the forwarding state of node 3 can be removed. Although the algorithm does not specify the sequence of the nodes which remove or move the forwarding states, the following lemmas and theorem prove that the algorithm can find the optimal solution for arbitrary sequence of nodes.

Lemma 3.6 *Given an optimal assignment S^*, for any feasible assignment S which is not optimal, there is a sequence of operations on S^* and S such that another optimal assignment can be obtained. Each operation corresponds to a node which removes its forwarding state or moves the state to its parent node.*

Proof: We prove this lemma by introducing an algorithm with a sequence of operations on S^* and S. After the algorithm stops, S^* and S are identical, and both are optimal. Let σ^*_m and σ_m denote the assignment of S^* and S at node m. The following is the details of the algorithm.

1. Number all nodes from 1 to $|V_t|$ by the breadth first search algorithm.
2. For m from $|V_t|$ to 2, m is not a receiver



(a) Initial assignment. (b) Node 12 removes its forwarding state. (c) Node 2 removes its forwarding state.

(d) Node 6 removes its forwarding state. (e) Node 13 moves its forwarding state to node 6. (f) Node 3 removes its forwarding state.

Figure 4. An example of the distributed greedy algorithm P1_greedy with $\delta = 2$.

if $\sigma_m^* = 0$ and $\sigma_m = 1$, then $\sigma_m \leftarrow 0$, $\sigma_{p_m'} \leftarrow 1$;
else if $\sigma_m^* = 1$ and $\sigma_m = 0$, then $\sigma_m^* \leftarrow 0$, $\sigma_m^* \leftarrow 1$;
end if;
end for.

In step 2, the algorithm maintains feasibility of both solutions in each iteration. The algorithm compares node m in S and S^*. If m is a state node in only S or S^*, say, S', m removes its forwarding state when the parent node is a state node, or moves the forwarding state to its parent node when the parent node is stateless. The state nodes in the sub-tree rooted at m in both assignments must be identical since all nodes in the sub-tree have been compared. At the end of each iteration, the upstream state node of m in another assignment, say, S'', is identical or upstream to the upstream state node of m in S'. Therefore, S' is feasible since S'' is also feasible. After the algorithm stops, the state nodes in both assignments are the same. Since the number of state nodes in S^* is not increased, S^* is still an optimal assignment. Therefore, S is also an optimal assignment.

Lemma 3.7 *If no state node can remove its forwarding state or move the state to its parent node, the assignment is optimal.*

Proof: We assume that the assignment is not optimal. Since it is a feasible assignment, an optimal assignment can be obtained after a sequence of operations on the assignment according to lemma 3.6. It contradicts that no state node can remove or move its forwarding state.

Theorem 3.8 *The distributed greedy algorithm can find an optimal assignment with* $O(|V_t|^2)$ *operations.*

Proof: According to lemma 3.6, since the assignment at the end of each iteration is feasible, there exists a sequence of operations which leads to an optimal solution. Since there are at most $|V_t|$ state nodes, and a forwarding state can be removed once and moved upstream at most $|V_t|$ times, there are at most $O(|V_t|^2)$ operations before the algorithm stops. According to lemma 3.7, an optimal solution is obtained when the algorithm stops. Therefore, the algorithm can find an optimal assignment with $O(|V_t|^2)$ operations.

IV. Minimizing the Maximum Number of Forwarding States in a Router

In this section, we consider the problem of optimizing the assignment of state nodes among multiple multicast groups. Although the algorithms proposed in last section can minimize the total number of forwarding states maintained in all routers, the distribution of forwarding states among routers is not balanced. Since it is more difficult to aggregate multicast

forwarding states, some routers may not have enough memory to store all forwarding states, but others are under-utilized and capable of storing more forwarding states. In this section, therefore, we regard minimizing the maximum number of forwarding states in a router as the objective of the optimization problem. With the objective, a router with too many states will move some states to other routers in order to reduce the objective value. Therefore, the distribution of forwarding states among routers can be more balanced. Note that we can also use minimizing the maximum memory usage of a router as the objective function, where the memory usage of a router is the number of forwarding states stored in the router over the maximum number of forwarding states that the router can have.

We model the optimization problem as an Integer Linear Programming (ILP) problem. We design two algorithms to solve this problem. The first one is based on Lagrangean relaxation on the proposed ILP formulation. It decomposes the problem into multiple sub-problems. Each sub-problem is to assign the state nodes in a single multicast tree, similar to P1. The second algorithm is based on the distributed greedy algorithm described in last section.

A. *Integer linear programming*

The objective function is as follows.

$$\text{minimize } v_{\max}.$$

The problem is subject to the following constraints.

$$d^t_{p^t_n,n} \le (1-\sigma^t_n)\times\delta, \forall t, \forall n \in V_t, \quad (1)$$

$$\sum_{n\in C^t_m} \sigma^t_n + d^t_{p^t_n,n} \le d^t_{p^t_m,m} + \sigma^t_m \times |C^t_m| \times \delta, \forall t, \forall m \in V_t - \{r_t\} - R_t, \quad (2)$$

$$\sum_{t:m\in V_t} \sigma^t_m \le v_{\max}, \forall m \in V, \quad (3)$$

$$\sigma^t_m = 0, \forall t, \forall m \in R_t, \quad (4)$$

$$\sigma^t_r = 1, \forall t, \quad (5)$$

$$d^t_{p^t_m,m} = 1, \forall t, \forall m \in R_t, \quad (6)$$

where $d^t_{p^t_n,n}$ is an auxiliary integer variable. In the above constraints, $\sigma^t_n + d^t_{p^t_n,n}$ represents the number of destinations of p^t_n from the downstream interface to n. If $\sigma^t_n = 0$, constraint (1) enforces that $d^t_{p^t_n,n}$ must be no more than δ. If $\sigma^t_n = 1$, constraint (1) enforces that $d^t_{p^t_n,n} = 0$. The summation in the left hand side of constraint (2) is the total number of destinations of m from all downstream interfaces. Constraint (2) states that if m is not a state node in t, $d^t_{p^t_m,m}$ must be equal to or larger than the number of destinations of m from all downstream interfaces. If m is a state node in t, constraint (2) always holds. Constraint (3) guarantees that $v_{\max}$ must be no less than the number of forwarding states stored in each router. Constraint (4) enforces that all receivers of t are stateless nodes. Constraint (5) states that the root of t is a state node. Constraint (6) guarantees that each receiver in t must have an upstream state node.

B. *Lagrangean relaxation*

Although standard algorithms for ILP, such as branch-and-bound and cutting-plane algorithms, can find the optimal solution, the computational time grows exponentially for large problems [22]. Therefore, we design an algorithm using Lagrangean relaxation on the ILP formulation. Although the algorithm cannot obtain the optimal solution, it can find a good solution in reasonable time. The algorithm first finds a feasible solution, and then improves the solution iteratively. It decomposes the original problem into multiple sub-problems. In each sub-problem, each node is associated with a cost, i.e., the Lagrange multiplier. Each sub-problem is to assign the state nodes of a single multicast tree such that the total cost of all state nodes in the tree is minimized.

We decompose the original problem in the following way. We relax constraint (3), and the new objective function is as follows,

$$\text{minimize } v_{\max} + \sum_{m\in V} u_m \times \left(\sum_{t:m\in V_t} \sigma^t_m - v_{\max} \right),$$

where u_m is the Lagrange multiplier of node m, $u_m \ge 0, m \in V$. The above objective function is the same as follows.

$$\text{minimize } \left(1 - \sum_{m\in V} u_m\right) \times v_{\max} + \sum_t \sum_{m\in V_t} u_m \times \delta^t_m.$$

With the objective function, we decompose the original problem into multiple sub-problems. Each sub-problem is associated with a multicast tree with the following objective function.

$$\text{minimize } \sum_{m\in V_t} u_m \times \sigma^t_m.$$

Therefore, each sub-problem is to minimize the total cost of all state nodes in multicast tree t, where u_m corresponds to the cost of node m. In the beginning of each iteration, we use the sub-gradient algorithm [22] with the results in last iteration to find the Lagrange multipliers of all nodes. The Lagrange multipliers first determine the optimal $v_{\max}$ in the iteration[4]. Then the multipliers are given as input parameters for all sub-problems. Each sub-problem is solved by the proposed dynamic programming algorithm P1_dp with some modifications[5]. Intuitively, a node with more forwarding states

---

[4] If $1 - \sum_{m\in V} u_m \ge 0$, $v_{\max} \leftarrow 0$; otherwise $v_{\max} \leftarrow |T|$. In the Lagrangean relaxation problem, $v_{\max}$ does not represent the maximum number of forwarding states in a router since constraint (3) is relaxed.

[5] In the Lagrangean relaxation problem, $\tau^t_m(j)$ represents the total cost of state nodes in the sub-tree rooted at m if p^t_m has j destinations from the downstream interface to m. Variable $v^t_{\min}$ represents the minimum cost of multicast tree t, namely, the optimal value of the sub-problem. Some assignments in step 2 of P1_dp are changed as follows.

in last iteration induces a larger Lagrange multiplier in this section. Therefore, less multicast tree will use the node as a state node. Since the solution of each sub-problem is also a feasible solution to P2, the solution of our algorithm is the assignment whose maximum number of forwarding state is minimal among all iterations

C. *Distributed greedy algorithm*

The methods based on ILP and Lagrangean relaxation require centralized computation and can not be implemented as a distributed protocol. Here we propose a distributed algorithm, P2_greedy, which is modified from the distributed greedy algorithm in section III. The algorithm is based on the following observations.

- The number of state nodes in a multicast tree must be reduced in order to minimize the number of forwarding states maintained in a node.
- In order to balance the distribution of forwarding states, a state node should move its forwarding state to the node with the least number of forwarding states.

Based on the above two observations, each state node first checks if it can remove its forwarding state. If it can not remove the state, it tries to move the forwarding state to the upstream stateless node with the least number of forwarding states. The upstream stateless node may not be the parent node. The algorithm is summarized in Fig. 5. In the next section, we show that the algorithm can obtain close-to-optimal solutions in our simulation results.

V. SIMULATION

In this section, we show the simulation results of the above two optimization problems with the proposed algorithms. We first use small flat graphs with the Waxman distribution [23] as the network topologies to test the performance of our algorithms in graphs with different characteristics. In order to test our algorithm in more realistic networks, we also use large graphs with the power-law distribution generated by Inet [24]. The simulation parameters are listed as follows.

- Graph characteristic. We use $(\alpha=0.2,\beta=0.2)$, $(\alpha=0.25,\beta=0.25)$, and $(\alpha=0.3,\beta=0.3)$ as the parameters of the Waxman distribution in the flat graphs. The graphs with larger α and β have larger node degree and smaller graph diameter [25]. Therefore, the height of a multicast tree in the graph is smaller, and a node has more child nodes in a multicast tree.
- Group size. The group size is the number of receivers in a multicast group.

$$\tau_m^t(1) \leftarrow \begin{cases} \tau_n^t(1) - u_k + \min\{u_k, u_m\}, & \text{if } \left|C_m^t\right| = 1, n \in C_m^t \\ u_m + \sum_{n \in C_m^t} \min_{1 \le j_n \le \delta} \left\{\tau_n^t(j)\right\}, & \text{if } \left|C_m^t\right| > 1 \end{cases},$$

where k is the downstream state node of m from the interface to n; moreover,

$$\nu_{\min}^t \leftarrow u_{r_t} + \sum_{n \in C_{r_t}^t} \min_{1 \le j_n \le \delta} \left\{\tau_n^t(j_n)\right\}.$$

Given: a set of multicast trees T, each tree $t \in T$ with node $V_t \subseteq V$ and arc $A_t \subseteq A$, δ.

Find: $\nu_{\max}$, and σ_m^t, $t \in T, m \in V_t$.

Algorithm:

1. Initialization.
   $\sigma_m^t \leftarrow 1, x_m^t \leftarrow 1, t \in T, m \in V_t - R_t - \{r_t\}$;
   $\sigma_{r_t}^t \leftarrow 1, x_{r_t}^t \leftarrow 0, t \in T$;
   $\sigma_m^t \leftarrow 0, x_m^t \leftarrow 0, t \in T, m \in R_t$.
2. While $\exists m \in V_t - R_t - \{r_t\}, t \in T$ such that $x_m^t = 1$
   if $\left|D_{u_m,m}^t\right| + \left|D_m^t\right| - 1 \le \delta$, then
   $\sigma_m^t \leftarrow 0, x_m^t \leftarrow 0$;
   $x_n^t \leftarrow 1, \forall n \in D_{u_m,m}^t \cup \{u_m^t\}$;
   else let P denote the set of nodes on the path from u_m to m;
   if $\exists k \in P$ such that $\sigma_k^t = 0$ and $\left|D_{k,m}^t\right| + \left|D_m^t\right| - 1 \le \delta$, then
   $j \leftarrow \arg\min_k \sum_{t' \in T} \sigma_k^{t'}$;
   $\sigma_m^t \leftarrow 0$, $x_m^t \leftarrow 0$; $\sigma_j^t \leftarrow 1$, $x_j^t \leftarrow 1$;
   $x_n^t \leftarrow 1, n \in D_j^t \cup D_{u_j,j}^t \cup \{u_j^t\}$;
   else $x_m^t \leftarrow 0$;
   end if;
   end if;
   end while.
3. $\nu_{\max} \leftarrow \max_{m \in V} \sum_{t: m \in V_t} \sigma_m^t$.

Figure 5. The distributed greedy algorithm to minimize the maximum number of forwarding states in a router (P2_greedy).

- The maximum number of destinations from a downstream interface, δ. When $\delta = 1$, all branching nodes are the only state nodes.
- Distribution of receivers. We use the affinity and disaffinity model in [26] to describe the distribution of receivers in a multicast group. The distribution of receivers is correlated with the topology of a multicast tree. With positive affinity index, receivers tend to cluster together, and the distribution is suitable for applications such as local news and traffic reports. With negative affinity index, the receivers tend to spread out, and the distribution is proper for applications such as video conferencing. When affinity index is zero, all receivers are chosen uniformly over all nodes.

Figs. 6, 7, and 8 are the optimal solutions to the problem which minimizes the number of state nodes in each multicast tree, i.e., P1. The network has 100 nodes and 100 multicast trees. The simulation results are averaged over 100 samples. Fig. 6 is the total number of state nodes in all multicast trees versus different δ with different group sizes. The number of state nodes in a multicast tree decreases as δ increases. A multicast group with larger group size has more state nodes. Fig. 6 also shows that a small δ is sufficient to eliminate over

50% forwarding states. Fig. 7 is the percentage of nodes with the forwarding states in a multicast tree versus different group sizes with different affinity indices. With larger group size, more nodes in a multicast tree have to maintain forwarding states. With larger affinity index, receivers tend to cluster together, so fewer nodes in a multicast tree have to maintain forwarding states. As the group size approaches 100, the results of different affinity indices converge since most nodes in the network have receivers of a multicast tree. Fig. 8 is the percentage of nodes with forwarding states in a multicast tree versus different group sizes with different parameters in Waxman distribution. For graphs with larger α and β, each state node has more child nodes and can serve more receivers, so a smaller percentage of nodes has to maintain forwarding states

Figs. 9 and 10 focus on the problem of minimizing the maximum number of forwarding states maintained in a router, i.e., P2. In addition to this performance metrics, we also measure the standard deviation of the number of forwarding states maintained in all nodes. We use P2_greedy algorithm to solve this problem. Fig. 9 shows that the distribution of forwarding states is more balanced as δ increases. When $\delta = 1$, nodes with higher degree tend to be used by more multicast trees as branching nodes, and they have to maintain a large number of forwarding states. With $\delta > 1$, the standard deviation decreases since our algorithm can remove the forwarding states or move them to nearby non-branching nodes with less forwarding states. Fig. 10 shows the distribution of forwarding states is more balanced for graphs with larger α and β due to higher connectivity.

Fig. 11 compares the performance of different algorithms. In Fig. 11 (a), we use P2_greedy to solve P1. The group size is 70, and the affinity index is zero. It shows that results are very close to the optimal solutions of P1. In Fig. 11 (b), we compare the performance of our proposed algorithms with the optimal solutions in P2. The optimal solutions are obtained by solving ILP using CPLEX [27]. Due to the computational time of solving ILP, we use a smaller network with 30 nodes and 20 groups. It shows that the results of P2_greedy are close to the optimal solutions. The results of P1_greedy and Lagrangean relaxation are also shown. From Fig. 11, it shows that P2_greedy performs well in both problems.

Fig. 12 is the results of two problems using the graphs with power-law distribution. Each graph has 3500 nodes, and the simulation contains 1000 multicast trees. The two problems are solved by P1_greedy and P2_greedy. With a small δ, it shows that the both the number of forwarding states in a multicast tree and the standard deviation can be reduced over 50%.

VI. Discussion

The choice of δ is important in our algorithm. Although the number and the distribution of forwarding states are better for a larger δ, it takes more time to forward a packet in each stateless node since the state node has to look up more addresses when a data packet arrives. However, the end-to-end forwarding time may not increase compared with traditional IP multicast which uses only the branching routers to store the

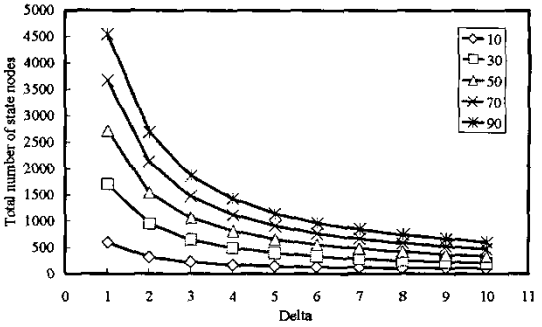


Figure 6. Comparison of the total number of state nodes in all multicast trees with different group sizes, affinity index = 0.

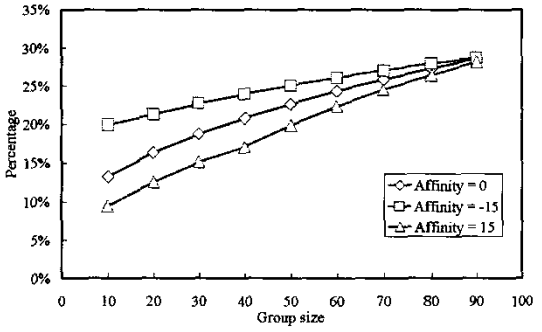


Figure 7. Comparison of the percentage of nodes with the forwarding states in a multicast tree with different affinity indices, $\delta = 2$.

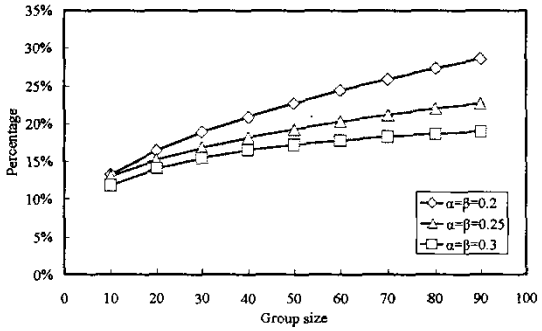
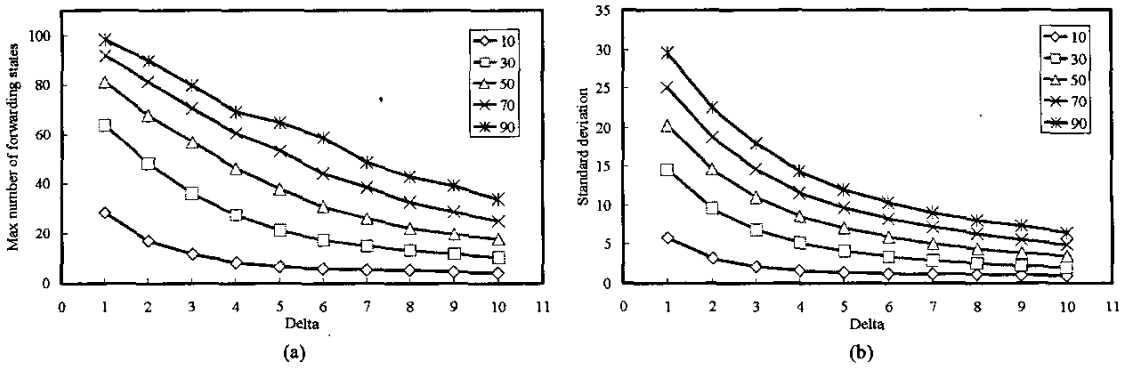


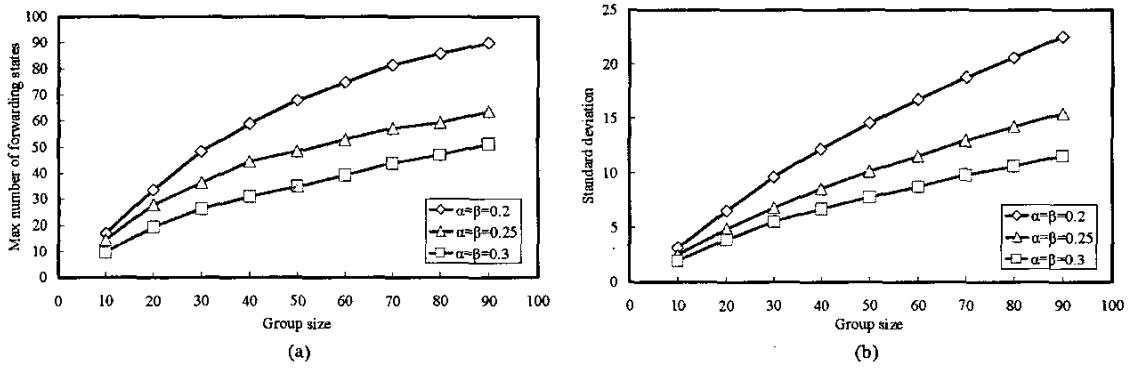
Figure 8. Comparison of the percentage of nodes with the forwarding states in a multicast tree with different (α, β) in Waxman distribution, $\delta = 2$, affinity index = 0.

forwarding states due to the following reasons. First, in an end-to-end path, much fewer nodes have to look up multicast forwarding states. To look up a multicast forwarding state using a multicast address or a channel identifier takes much more time than a unicast address since multicast forwarding states are hard to be aggregated and need an exact-match algorithm for address look-up. Second, the forwarding speed of Xcast can be improved by writing bitmaps [19], Xcast to unicast [19], and hierarchical encoding of destination addresses [20]. Third, in this paper, we assume δ is the same for all



(a) (b)

Figure 9. Comparison of the distribution of forwarding states with different group sizes, affinity index = 0.



(a) (b)

Figure 10. Comparison of the distribution of forwarding states with different (α, β) in Waxman distribution, $\delta = 2$, affinity index = 0.

routers. It can be extended that each router can have different δ according to its forwarding speed.

The distributed greedy algorithm P2_greedy is suitable to be implemented as a protocol due to the following reasons. First, the algorithm is distributed and asynchronous. Each state node operates independently and only use information stored in the node and neighbor nodes. Second, the algorithm is much simpler than others. Third, as shown in the last section, the algorithm obtains very good solutions to both optimization problems. Fourth, it can support the dynamic group membership easily. When a receiver joins or leaves a multicast tree, its upstream state node adds or removes the receiver's address in the forwarding state. The assignment of forwarding state may need to be adjusted. The upstream state node may remove or move its forwarding state to adapt to the new multicast tree. On the contrary, the upstream state node may create a new state node on the path downstream to the receiver if the number of destinations from the downstream interface to the receiver exceeds δ. Compared with the distributed greedy algorithm, P1_dp is not suitable to be implemented as a protocol due to high overhead of handling the dynamic group membership. All nodes on the path from the root to the receiver have to calculate $\tau'_m(j)$, and the stateless nodes on the path have to cache the information temporarily. The protocol design based on P2_greedy is on-going.

VII. Conclusion

In this paper, we propose a new multicast forwarding mechanism with resource optimization based on Xcast. With our algorithms, a set of routers in each multicast tree is adaptively chosen to maintain forwarding states. Multicast packets are sent between these routers via Xcast. The assignment of the forwarding states at routers in a multicast tree is formulated as two optimization problems. The first one is to minimize the number of routers which maintain the forwarding states in each multicast tree. The second one is to minimize the maximum number of forwarding states stored in a router. Several algorithms for both problems are proposed. We prove that our algorithms can find the optimal solution to the first problem. By simulation, we also show that the solution to the second problem is close to the optimal solution. We prove that the approach which assigns all branching routers as the only routers with forwarding states is a special case in our mechanism. We show that our mechanism uses less forwarding states, and the distribution of forwarding states is more balanced. More flexible and efficient allocation of multicast forwarding states among routers can be achieved by our algorithms compared with traditional IP multicast.

References

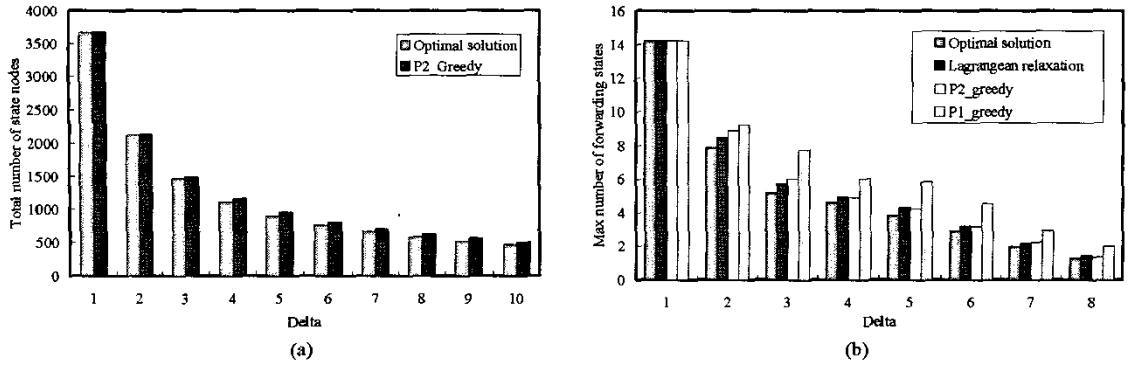


Figure 11. Comparison of the performance of different algorithms.

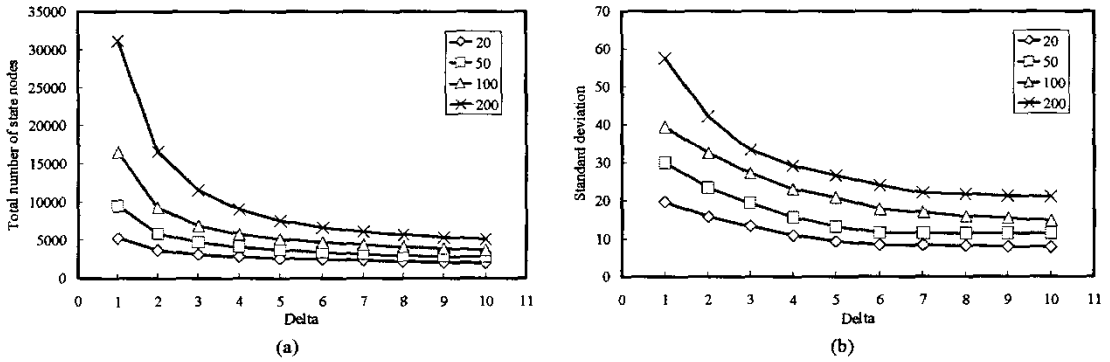


Figure 12. Results of the two problems using power-law topologies generated by Inet with different group sizes, affinity index = 0.

[1] P. V. Mieghem, G. Hooghiemstra, and R. Hofstad, "On the efficiency of multicast," IEEE/ACM Transactions on Networking, vol. 9, no. 6, pp. 719-732, Dec. 2001.

[2] W. Fenner, "Internet group management protocol, version 2," IETF RFC 2236, Nov. 1997.

[3] D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol," IETF RFC 1075, Nov. 1988.

[4] J. Moy, "Multicast routing extensions for OSPF," Communications of the ACM, vol. 37, no. 8, pp 61-66, Aug. 1994.

[5] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT)," ACM SIGCOMM, 1993, pp. 85-95.

[6] S. Deering et al., "The PIM architecture for wide-area multicast routing," IEEE/ACM Transactions on Networking, vol. 4, no. 2, pp. 153-162, Apr. 1996.

[7] D. Thaler, M. Handley, and D. Estrin, "The Internet multicast address allocation architecture," RFC 2908, Sep. 2000.

[8] H. Holbrook and B. Cain, "Source-specific multicast for IP," IETF Internet Draft, draft-ietf-ssm-arch-04.txt, Oct. 2003.

[9] P. I. Radoslavov, E. Estrin, and R. Govindan, "Exploiting the bandwidth-memory tradeoff in multicast state aggregation," Tech. Rep. 99-697, USC Computer Science Department, 1999.

[10] T. Wong, R. Katz, and S. McCanne, "An evaluation of preference clustering in large-scale multicast applications," IEEE INFOCOM, pp. 451-460, 2000.

[11] S. Song, Z. Zhang, B. Choi, and D. H. C. Du, "Protocol independent multicast group aggregation scheme for the global area multicast," IEEE GLOBECOM, pp. 370-375, 2000.

[12] A. Fei, J. Cui, M. Gerla, and M. Faloutsos, "Aggregated multicast: an approach to reduce multicast State," IEEE GLOBECOM, pp. 1595-1599, 2001.

[13] J. Tian and G. Neufeld, "Forwarding state reduction for sparse mode multicast communication," IEEE INFOCOM, pp. 711-719, 1998.

[14] I. Stoica, T. S. E. Ng, and H. Zhang, "REUNITE: a recursive unicast approach to multicast," IEEE INFOCOM, pp. 1644-1653, 2000.

[15] T. Wong and R. Katz, "An analysis of multicast forwarding state scalability," IEEE ICNP, pp. 105-115, 2000.

[16] V. Visoottiviseth, H. Kido, and Y. Takahashi, "Sender initiated multicast (SIM)," IETF Internet Draft, draft-vasaka-xcast-sim-01.txt, Mar. 2003.

[17] A. Boudani and B. Cousin, "Simple explicit multicast (SEM)," IETF Internet Draft, draft-boudani-simple-xcast-03.txt, Jun. 2003.

[18] A. Boudani, B. Cousin, and J. Bonnin, "An effective solution for multicast scalability: The MPLS Multicast Tree (MMT)," IETF Internet Draft, draft-boudani-mpls-multicast-tree-04.txt," Jun. 2003.

[19] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens, "Explicit multicast (Xcast) basic specification," IETF Internet Draft, draft-ooms-xcast-basic-spec-05.txt, Aug. 2003.

[20] D. Ooms and W. Livens, "Connectionless multicast," IETF Internet Draft, draft-ooms-çl-multicast-02.txt, April 2000.

[21] H. W. Holbrook and D. R. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications," ACM SIGCOMM, pp. 65-77, 1999.

[22] G. L. Nemhauser and L. A. Wosley, "Integer and combinatorial optimization," Wiley-Interscience series in discrete mathematics and optimization, 1999.

[23] B. M. Waxman, "Routing of multipoint connections," IEEE Journal on Selected Areas in Communications. vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[24] INet topology generator, http://topology.eecs.umich.edu/inet/.

[25] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for Internet topology," IEEE/ACM Transactions on Networking, vol. 5, no. 6, pp. 770-783, Dec. 1997.

[26] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of multicast trees: comments on the Chuang-Sirbu scaling law," ACM SIGCOMM, pp. 41-51, 1999.

[27] CPLEX optimization package, http://www.ilog.com/products/cplex/.