Forwarding State Scalability for Multicast Provisioning in IP Networks

Baoxian Zhang and Hussein T. Mouftah, University of Ottawa

ABSTRACT

Forwarding state scalability is one of the critical issues that delay the multicast deployment in IP networks. With traditional multicast routing protocols, a forwarding tree is built for each multicast session, and each router is required to maintain a forwarding entry for each multicast session whose distribution tree passes through the router. This poses the multicast forwarding state scalability issue when the number of concurrent multicast sessions is very large. In this article we first present a survey of existing work addressing this scalability issue for providing scalable IP multicast. Then we extend an existing multicast routing protocol, *Multicast Extension to OSPF* (MOSPF), to scale well with respect to the number of concurrent multicast sessions by introducing tunnel support. This extension aims to reduce the protocol overhead associated with MOSPF. Simulation results show that the extension can significantly reduce multicast forwarding state and computational overhead at routers without affecting the per-destination shortest path characteristic of a resulting tree or introducing extra control overhead.

INTRODUCTION

IP multicast is an efficient point-to-multipoint delivery mechanism because packets disseminated from the source of a group to members of the group travel only once through common parts of the network. IP multicast has been a hot topic of research and development for more than one decade. However, there are still some open issues that make it difficult for IP multicast to be deployed in the global Internet.

The multicast forwarding state scalability is one of the critical issues that delay the deployment of IP multicast. With traditional Internet protocols, each router is required to maintain a forwarding entry for each multicast session whose distribution tree passes through the router. When there is a very large number of concurrent multicast sessions, the number of the corresponding multicast forwarding entries at routers is also very large. This could consume more router memory and might also result in slower packet forwarding as each packet forwarding involves a routing table lookup. This is the forwarding state scalability issue in providing scalable IP multicast.

To support unicast packet forwarding well, techniques such as hierarchical address allocation and packet forwarding based on the longest prefix match help to achieve scalability. However, this cannot easily be applied to IP multicast in the current Internet. The reason is as follows. For supporting IP multicast today, a class D IP address is allocated for each multicast session, and a multicast routing protocol is used to build a forwarding tree to cover members of the session. As a result, a class D IP address corresponds to a logical group and is not related to any topological information.

The forwarding state scalability issue in providing IP multicast has received much attention in recent years. A variety of work has been carried out to provide scalable multicasting in the Internet. In this article we first give a review of existing work addressing this scalability issue and discuss their advantages and disadvantages. Then we present a scalable multicasting mechanism to improve the scalability of an existing protocol, Multicast Extension to Open Shortest Path First (MOSPF) [1], by introducing tunnel support to reduce multicast forwarding state as well as computational overhead at routers.

EXISTING WORK

The forwarding state scalability issue in IP multicast has received much attention in recent years. A variety of scalable mechanisms have been proposed to address this critical issue.

In some mechanisms (e.g., [2]), multicast state at routers is completely eliminated by using *application layer multicast*, which pushes the complexity to endpoints. With such mechanisms, only unicast service is provided at the network layer, and endpoints implement all multicast-related functionalities, including membership management, packet replication, as well as error, flow, and congestion control. Compared with network layer multicast, duplicate transmission on some physical links may occur, and longer end-to-end delay can be introduced. These are not friendly to multimedia applications, which are usually delay-sensitive and require high transmission bandwidth. In the rest of this article we will not discuss this kind of mechanism in detail. Instead, we will focus exclusively on those mechanisms providing scalable multicasting at the network layer. For simplicity, the terms *node* and *router* will be used interchangeably unless otherwise specified.

0163-6804/03/$17.00 © 2003 IEEE

Methods	Category	Strategy	Multicast topology	QoS tree support	Small groups required
Router-centric aggregation [3]	State aggregation	Multistate aggregation	Any tree	Yes	No
Interface-centric aggregation [4]	State aggregation	Multistate aggregation	Any tree	Yes	No
Tree-centric aggregation [5]	State aggregation	Multitree aggregation	Any tree	Limited	No
Dynamic Steiner tree [7]	Nonbranching state elimination	Tunnel-based multicasting	Tunnel tree	Limited	Yes
DTM [8]	Nonbranching state elimination	Tunnel-based multicasting	Shortest path tree or core-based tree	No	Yes
REUNITE [9]	Nonbranching state elimination	Recursive unicast	Shortest path tree	No	Yes

Table 1. *Comparison of mechanisms addressing the multicast forwarding state scalability issue at the network layer.*

The performance of this leaky aggregation approach in achieving high-level aggregation of adjacent groups can largely benefit if group addresses can be allocated in such a way that groups with the same prefix are rooted at the same domain.

Before discussing mechanisms addressing the scalability issue at the network layer, we first give the information maintained in a multicast forwarding entry according to traditional protocols. In its most general form, the following information is usually maintained in such an entry: a globally unique class D IP address assigned for the group, source address, incoming interface, outgoing interface set, and possibly a timer associated with each outgoing interface depending on the selected routing protocol.

Next, we review recent work proposed to reduce multicast forwarding state at routers through state aggregation and nonbranching state elimination, respectively.

State Aggregation

Mechanisms implementing the concept of state aggregation reduce multicast forwarding state by aggregating entries for multiple groups as one if these groups have certain characteristics in common. Existing work performing multicast state aggregation works on a per-router [3], per-interface [4], or per-tree [5] basis as follows.

In [3], the authors present three approaches performing state aggregation on a per-router basis. In detail, at a router the state for multiple groups with both the same address prefix[1] and the same interface set (i.e., same incoming interface and outgoing interface set) can be aggregated into one entry locally maintained for forwarding packets belonging to each of these groups. Since a perfect match of the interface set is required, this is called *strict aggregation*. *Pseudo aggregation* is developed as a variant of strict aggregation, with which a non-locally-active group[2] can be among those groups to be aggregated if it has the same address prefix, because no data packets belonging to such a group will be received at the current router. In reality, it is unlikely that there will be many group prefixes that satisfy the above conditions. To further reduce state, leaky aggregation is developed as an alternative by relaxing the requirement of perfect match of the outgoing interface set of the groups (to be aggregated) to leaky match. The performance of this leaky aggregation approach in achieving high-level aggregation of adjacent groups can largely benefit if group addresses can be allocated in such a way that groups with the same prefix are rooted at the same domain, as suggested in [6]. A disadvantage of leaky match is that some bandwidth is wasted to deliver data to nodes not leading to any group member(s). An overall comparison of mechanisms addressing the multicast forwarding state scalability issue can be found in Table 1.

Instead of aggregating state for adjacent groups with an identical interface set as in [3], an alternative strategy [4] describes an interface-centric model to implement the concept of state aggregation on a per-interface basis. Entries for a number of groups with adjacent addresses at an interface can be aggregated as a single range if these groups make the same decision at this interface. Moreover, a non-locally-active group can be one of these groups to be aggregated if its address is located in the range. Accordingly, the forwarding table structure is reorganized such that for each packet, a per-interface decision is made on whether to forward a packet out or not if at an outgoing interface, or on whether to accept a received packet or not if at an incoming interface, based on the range to which the address of the packet belongs. This alternative organization promises greater non-leaky aggregation because there is more likelihood of adjacent groups making the same decision at a single interface than of adjacent groups making the same decision on the identical set of incoming and outgoing interfaces. Results in [4] show that state aggregation is possible by a factor of four using this approach. The cost of implementing this approach is that a processor is required at each interface of a router, which can lead to a much higher router cost.

Another strategy [5] achieves state reduction through intergroup tree sharing; it is targeted as an intradomain multicast provisioning mechanism. The key idea behind this strategy is that instead of constructing a tree for each individual group in the core network (backbone), multiple groups can be forced to share a single aggregated tree if their trees are of the same " shape" on the backbone. In contrast to the mechanisms in [3, 4],

[1] *In this case, these groups must be with adjacent addresses. In this work, groups with adjacent addresses are called* adjacent groups *without causing confusion.*

[2] *A non-locally-active group means its address is not in use at the current router.*

One major issue related to the design of a scheme supporting tunnel-based multicasting is how to keep a multicast tree converging to the correct one when there is group membership dynamics.

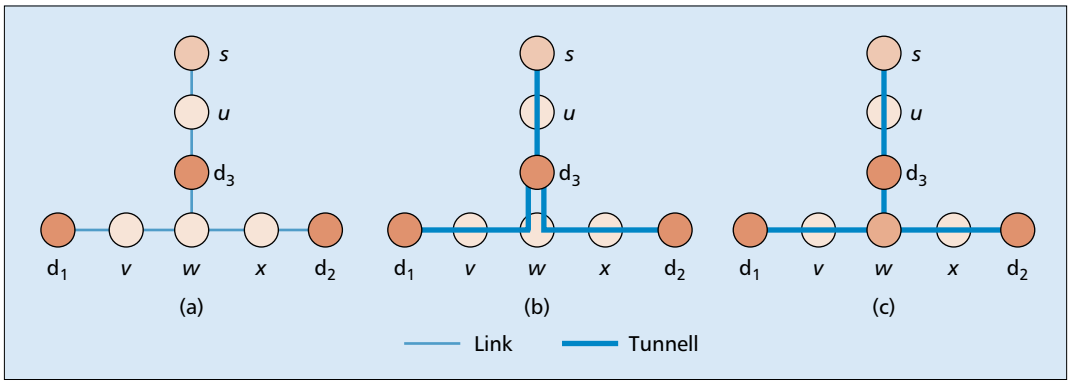

Figure 1. *An example of tunnel-based multicasting with a source node* s *and a set of destinations* S = {d_1, d_2, d_3}.

these aggregated groups do not necessarily have adjacent group addresses. Edge routers maintain state for each "pass-by" group, while core routers only need to maintain state per aggregated tree instead of per group. The more groups share an aggregated tree, the more state reduction can be achieved. Leaky aggregation can help further improve intergroup tree sharing at the cost of additional bandwidth to deliver packets to nodes not leading to any members of a group.

The mechanism in [5] has some limitations. First, it relies on a central manager for tree management and matching between groups and aggregated trees, which could become a bottleneck of the network. Second, compared to traditional protocols, a larger latency can be observed for tree migration to another aggregated tree when group dynamics occur. Other penalties paid include the CPU overhead of packet encapsulation/decapsulation at edge routers and the extra communication overhead.

Nonbranching State Elimination

Different from the concept of state aggregation, the strategy discussed next works by reducing the number of routers that must locally maintain a forwarding entry for a group. More specifically, state reduction is achieved by exempting those nonbranching nodes on a tree from maintaining group-specific forwarding state.

Mechanisms following this strategy are motivated by an observation that, in today's Internet, the typical group size is small for dominant multicast applications. That is, the number of *group member* routers (i.e., routers with host(s) subscribed to a particular group in its subnet) is significantly smaller than the number of routers in the network. In this case, the majority of routers simply forward packets from one incoming interface to one outgoing interface, and their maintenance of multicast forwarding state is unnecessary. In other words, the minority of routers are branching nodes. This is also referred to as *sparse mode multicast*. Results in [10] show that methods using nonbranching state elimination can achieve up to an order of magnitude in forwarding state reduction, which makes it a promising approach to address the multicast forwarding state scalability issue.

To exempt nonbranching on-tree nodes from maintaining forwarding state for a pass-by group, schemes in [7, 8] use the strategy of *tunnel-based multicasting*. The key idea behind these schemes is to view each multicast tree as a collection of unicast paths (tunnels). Each tunnel is a simple unicast route, on which all routers use the same routing metric for packet forwarding. A multicast datagram is encapsulated [11] inside a standard unicast datagram to go through a tunnel. That is, the entire multicast datagram is carried as the payload of an IP unicast datagram. Intermediate nodes of a tunnel do not have to maintain any forwarding state for the multicast session because they are involved only in regular unicast routing. The penalty paid is the CPU overhead of performing packet encapsulation and decapsulation, extra communication overhead, and additional state for more logical interfaces at tunnel endpoints.

One major issue related to the design of a scheme supporting tunnel-based multicasting is how to keep a multicast tree converging to the correct one in the presence of group membership dynamics. To address this issue, the scheme in [7] imposes a restriction: only the source and multicast destinations can be endpoints of tunnels. This restriction, however, may lead to the creation of inefficient trees. Consider the example shown in Fig. 1a. The resulting structure is shown in Fig. 1b, in which duplicate transmissions of datagrams can be observed on link (d_3, w) because node w is not eligible to be a tunnel endpoint according to the scheme in [7]. Tian *et al.* [8] proposed the idea of dynamically establishing tunnels to replace unbranched path-segments on a tree. Periodic tunnel request messages, sent from each downstream tunnel endpoint to its upstream tunnel endpoint, are required for dynamic tunnel management. A major disadvantage of this scheme is that the periodic transmission of tunnel request messages introduces a large amount of extra communication overhead, which may overshadow its gain in multicast forwarding state reduction.

Different from the strategy of tunnel-based multicasting discussed above, a recursive unicast approach, called REUNITE, is designed in [9] to remove multicast forwarding state at nonbranching routers on a delivery tree. REUNITE does not use class D IP addresses. Instead, both group identification and data forwarding are based on unicast IP addresses. More specifically, a multicast group is identified by a two tuple (unicast_IP_address, port_number). The basic idea of this recursive unicast approach is that a packet has the

address of a group member as its unicast destination address, based on which nonbranching routers simply forward such a packet further downstream. A router that acts as a branching node is responsible for creating packet copies with modified destination addresses in such a way that all group members in the downstream subtree of the router can receive the packet. REUNITE supports incremental deployment, load balancing, and graceful degradation when there are hot spots in the network.

Disadvantages of REUNITE are listed as follows. First, its implementation can partly affect the efficiency of unicast packet forwarding due to its addressing mechanism. Upon receiving a packet, a REUNITE router first performs a lookup in its multicast forwarding table. If no entry is found, it then performs a second lookup in its unicast forwarding table. Two lookups are therefore required to forward a regular unicast packet. Second, REUNITE is sensitive to dynamics of network and group membership. Network or membership dynamics may result in duplicate transmission of data packets on certain links. A change in a unicast route that acts as a branch of a delivery tree caused by network dynamics may result in a failure of a branching node sitting on that route to receive data packets from the source. In this case, downstream members of the branching node cannot receive data packets either. As a result, a rejoin process has to be enforced by each affected member to stay in the group.

Summary

A variety of work has been carried out to address the forwarding state scalability issue in IP multicast. Some mechanisms perform state aggregation to reduce forwarding state by aggregating state for multiple groups into one entry when these groups have certain common characteristics. Some other mechanisms perform nonbranching state elimination to reduce forwarding state by minimizing the number of routers that must locally maintain a forwarding entry for a group. As discussed earlier, each of these mechanisms has its advantages and also introduces a certain extra cost for its implementation. Some more results are listed here to provide better understanding of these mechanisms:

- A mechanism falling into the category of state aggregation can work with any multicast routing protocol since it performs state aggregation independent of the characteristics of forwarding trees.
- A mechanism falling into the category of nonbranching state elimination can only work on a tree that can be decomposed into a collection of unicast paths on which all routers use the same routing metric for packet forwarding. This, however, would not be a big issue in its application because typical tree structures such as shortest path trees (SPTs), core-rooted trees, and rendezvous-point-rooted trees meet this requirement.

Multicast Extension to OSPF Protocol

MOSPF is one of the protocols currently employed to provide IP multicast in the Internet, and it is the only one that builds SPTs. An SPT is a widely used type of tree for multicast provisioning due to its simplicity and low per-destination cost. As shown in the following, MOSPF also has the scalability issue over multicast forwarding state as well as computational overhead.

MOSPF is designed to work on top of OSPF and works as follows. Basically, an MOSPF router, if it is a *group member* router, floods a group membership link state advertisement (LSA) throughout the network, which allows all routers to have the same view of group membership for a multicast session. Accordingly, each MOSPF router can construct an identical SPT for each [source network, multicast destinations] combination by using Dijkstra's SPT algorithm. To ease the computational demand at routers, these trees are built on demand when the first datagram with a particular combination of source network and multicast destinations is received. The MOSPF router then determines its on-tree position and creates a forwarding entry for the session.

Problems in the MOSPF Protocol

One major disadvantage of MOSPF is the processing cost of the Dijkstra's SPT calculations required to compute the delivery tree for each pass-by group, which could be a big burden at routers when the number of concurrent multicast sessions is large. Furthermore, the maintenance of a group-specific forwarding entry at each on-tree router also creates the multicast forwarding scalability issue.

An Extension to MOSPF

To improve the scalability of MOSPF, we introduce the concept of tunnel-based multicasting into its implementation. The reason for selecting tunnel-based multicasting is that it requires nearly no change to the current IP multicast model, introduces little extra protocol overhead, and can achieve a high state reduction ratio, as reported in related work [7, 8]. The scheme designed in this work is therefore referred to as *Tunnel-MOSPF* (T-MOSPF). T-MOSPF works differently from those in [7, 8] for tunnel management and has the following characteristics:

- It removes both multicast forwarding state and Dijkstra's SPT computational overhead associated with a multicast session at intermediate nodes of tunnels.
- It supports dynamic group membership well.
- No extra control overhead for tunnel management is introduced.

Procedures for T-MOSPF

T-MOSPF maintains the following properties of MOSPF in its implementation. First, each router maintains global network state information and the same view of group membership via network-wide flooding of group membership LSAs. Second, a multicast tree is built on demand when the first multicast datagram with a particular [source network, multicast destinations] combination is received. Accordingly, we can see that protocol processing associated with a multicast session at a router is triggered upon arrival of the first multicast datagram of the session.

T-MOSPF extends MOSPF to achieve scalable multicasting by introducing tunnel support.

A variety of work has been carried out to address the forwarding state scalability issue in IP multicast. Some mechanisms perform state aggregation to reduce forwarding state by aggregating state for multiple groups into one entry when these groups have certain common characteristics.

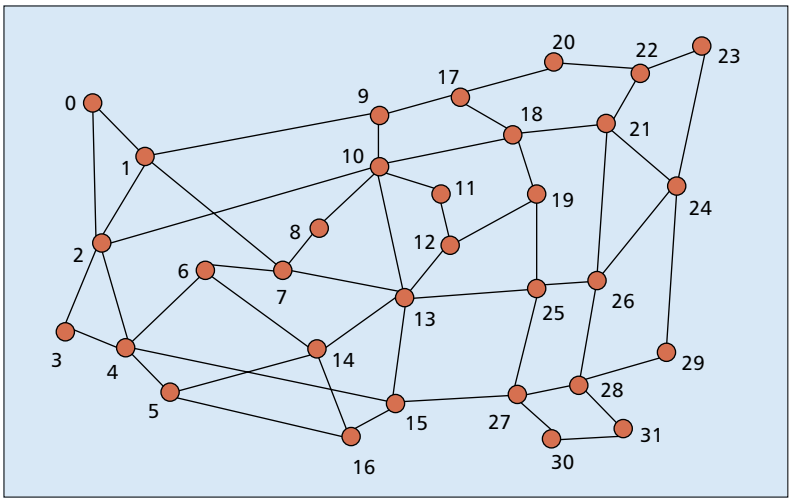

Figure 2. *An example network.*

Unlike the scheme in [7], with T-MOSPF all branching nodes, in addition to the source node and each multicast destination, are eligible to be endpoints of tunnels. To support tunnel-based multicasting, two types of downstream neighbors are defined and maintained in the routing table of an on-tree router for a particular group. A downstream router is said to be a *direct neighbor* if it is directly connected via a link, and if it is either a branching router or a group member router. Multicast datagrams are directly forwarded to such a neighbor without encapsulation. The other type of neighbor is a *logical neighbor*, which is a router connected via a tunnel with at least two hops. Multicast datagrams are encapsulated and unicast to such logical neighbors via tunnels. In this way, intermediate nodes of tunnels are not aware of the existence of the forwarding tree and then can be exempted from tree calculation and maintenance associated with the group.

Consider again the example shown in Fig. 1a. The resulting tree by T-MOSPF is shown in Fig. 1c, which can be viewed as a collection of three tunnels (i.e., $s \rightarrow d_3$, $w \rightarrow d_1$, $w \rightarrow d_2$) and one link $d_3 \rightarrow w$ because node w is a *direct downstream neighbor* of d_3 on the tree. Endpoints of these tunnels are nodes in $\{s\} \cup S \cup B$, where s is the source, S is the set of group members, and B is the set of nodes not in $S \cup \{s\}$, which have at least three on-tree neighboring links (e.g., node w in Fig. 1c is such a node and is called a *branching node*).

One issue related to the above extension to MOSPF is how to deal with group membership dynamics. This proves to be very easy with the assistance of the same view of group membership stored at each router. With MOSPF, when a change in group membership occurs (reflected by a change in group membership LSAs), all cache entries associated with the particular multicast group at a router must be cleared. A new forwarding tree will be built for the multicast application upon arrival of the next multicast datagram. Since all MOSPF routers have the same view of the updated group membership, it is easy for each router associated with the multicast application to calculate the correct SPT after receiving the first multicast datagram with a combination of the source network and the (updated) multicast destinations, and then to determine its position on the new forwarding tree. Likewise, when the network topology changes, an MOSPF router must clear its entire forwarding cache because all the SPTs must be rebuilt.

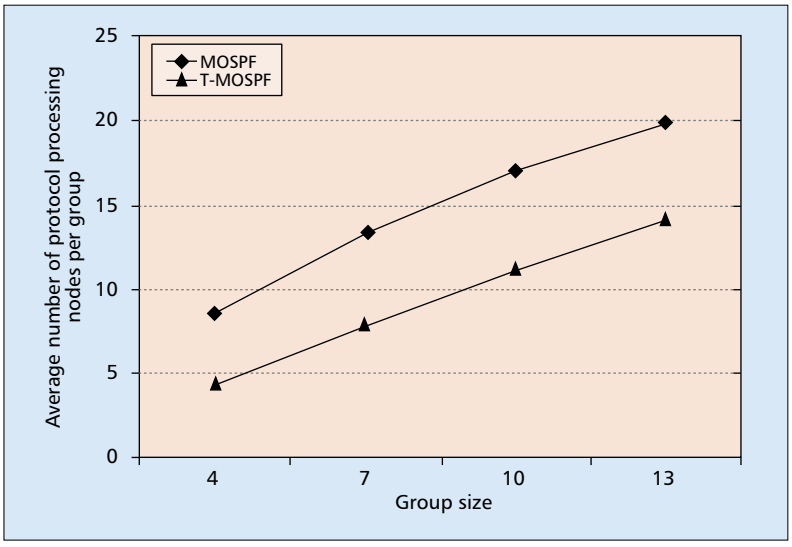

Figure 3. *Comparison of the average number of protocol processing nodes per group vs. group size for the example network in Fig. 2.*

Simulation Results

The cost performance of a resulting tree by T-MOSPF is the same as that by MOSPF since both of them return SPTs by using Dijkstra's SPT algorithm. Here, we focus on the protocol overhead associated with T-MOSPF for tree constructions. More specifically, we compare the average number of protocol processing nodes associated with a group by using T-MOSPF and MOSPF, respectively. A protocol processing node is a node that is required to perform the multicast-related functionalities (including forwarding state maintenance and tree calculation) for a group. The average number of protocol processing nodes per group is defined as the sum of protocol processing nodes for each multicast request over the total number of multicast requests.

The first experiment was designed to run on top of the network shown in Fig. 2. In the experiment, 100 trees were generated for each group size by T-MOSPF and MOSPF, respectively. For each group, the source and group members are randomly selected.

From Fig. 3 we can see that, compared to MOSPF, T-MOSPF significantly reduces the average number of protocol processing nodes per group, especially for small-size groups. Here, we define a measure called a *reduction ratio*, which is the difference between the average number of protocol processing nodes on a tree by MOSPF and the average number of such nodes on a tree by T-MOSPF, normalized to the average number of such nodes on a tree by T-MOSPF. In Fig. 3 the reduction ratio is up to 54 percent, which occurs when the group size is 4. This ratio decreases with the increase in group size. This is because all multicast destinations must be endpoints of tunnels, and hence with the group size increasing, the number of nodes

that can be bypassed as intermediate nodes (i.e., neither destinations nor branching nodes) of tunnels decreases.

The second experiment was run on an 8×8 grid mesh network. Figure 4 shows that the reduction ratio can be up to 68 percent.

In summary, from both experiments we observe that T-MOSPF scales well over MOSPF in terms of multicast forwarding state and computational overhead. Thus, the performance of T-MOSPF is quite satisfactory, particularly considering that no extra control overhead is introduced. Of course, the performance improvement comes at the cost of encapsulation/decapsulation overhead at endpoints of tunnels and extra communication overhead. However, this turns out to be trivial compared to the significant reduction in computational overhead and multicast forwarding state at routers.

Conclusions

IP multicast suffers from the multicast forwarding state scalability issue when the number of concurrent multicast sessions is very large. Techniques used to reduce unicast forwarding table size such as hierarchical address allocation and forwarding based on the longest prefix match cannot easily be applied to IP multicasting. Recently, a variety of work has been proposed to reduce multicast forwarding state at routers to provide scalable multicasting.

In this article we first present a review of existing work addressing this scalability issue to provide scalable multicasting at the network layer. In this work, some mechanisms reduce multicast state by aggregating state for multiple groups into one entry when these groups have certain characteristics in common, and perform state aggregation on a per-interface, per-router, or per-tree basis. Some other mechanisms reduce multicast forwarding state by removing forwarding state at nonbranching nodes on a tree, and are designed to support small groups.

Second, we extend an existing protocol, MOSPF, by introducing tunnel support to improve its scalability in terms of forwarding state and computational overhead. Simulation results show that the extension can significantly reduce forwarding state and computational overhead at routers. The extended protocol supports dynamic membership well and introduces no extra control overhead. Therefore, it is a simple and practical protocol for providing scalable multicast.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments and also Dr. D. Thaler for valuable discussion of his algorithm.

References

[1] J. Moy, "Multicast Extension to OSPF," RFC 1584, Mar. 1994.

[2] Y. H. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM SIGMETRICS 2000*, June 2000, pp. 1–12.

[3] P. I. Radoslavov, D. Estrin, and R. Govindan, "Exploiting the Bandwidth-memory Tradeoff in Multicast State Aggregation," Tech. rep. 99-697, Dept. of Comp. Sci., USC, Feb. 1999.

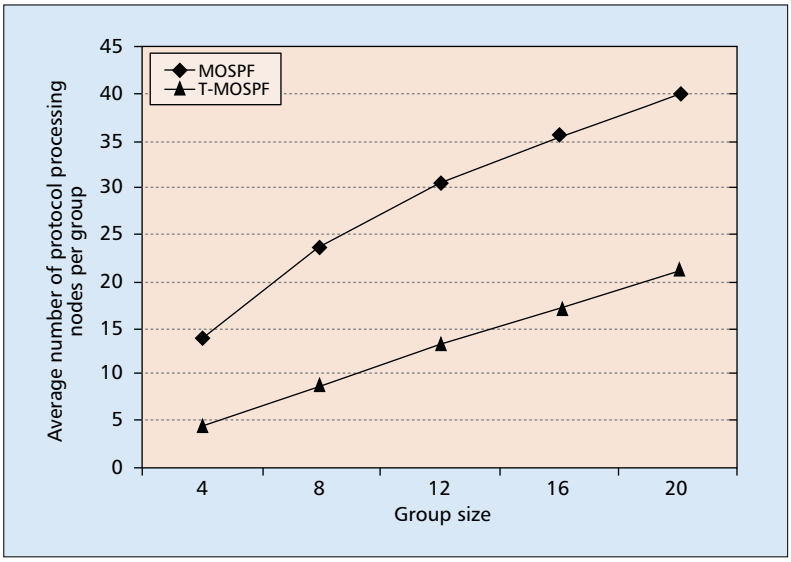

Figure 4. *Comparison of the average number of protocol processing nodes per group vs. group size for an 8×8 grid mesh network.*

[4] D. Thaler and M. Handley, "On the Aggregatability of Multicast Forwarding State," *Proc. INFOCOM 2000*, Mar. 2000, pp. 1654–63.

[5] A. Fei *et al.*, "Aggregated Multicast: An Approach to Reduce Multicast State," *Proc. IEEE GLOBECOM '01*, Nov. 2001, pp. 1595–99.

[6] K. Kumar *et al.*, "The MASC/BGMP Architecture for Inter-Domain Multicast Routing," *Proc. ACM Sigcomm '98*, Sept. 1998.

[7] E. Aharoni and R. Cohen, "Restricted Dynamic Steiner Trees for Scalable Multicast in Datagram Networks," *IEEE/ACM Trans. Net.*, vol. 6, no. 3, Jun. 1998, pp. 286–97.

[8] J. Tian and G. Neufeld, "Forwarding State Reduction for Sparse Mode Multicast Communication," *Proc. INFOCOM '98*, Apr. 1998, pp. 711–19.

[9] I. Stoica, T. S. Eugene Ng, and H. Zhang, "REUNITE: A Recursive Unicast Approach to Multicast," *Proc. INFOCOM 2000*, Mar. 2000, pp. 1644–53.

[10] T. Wong and R. Ratz, "An Analysis of Multicast Forwarding State Scalability," *Proc. ICNP 2000*, Nov. 2000, pp. 105–15.

[11] C. Perkins, "Minimal Encapsulation within IP," RFC 2004, Oct. 1996.

Biographies

Baoxian Zhang [M] (bxzhang@site.uottawa.ca) received his B.S., M.S., and Ph.D. degrees in electrical engineering from Northern Jiaotong University, Beijing, China, in 1994, 1997, and 2000, respectively. From January 2001 to August 2002, he worked with the Department of Electrical and Computer Engineering, Queen's University, as a postdoctoral fellow. He is now a researchassociate at the School of Information Technology and Engineering, University of Ottawa. His research interests include routing algorithm and protocol design, QoS management, and wireless ad hoc networks.

Hussein Mouftah [F] (mouftah@site.uottawa.ca) joined the School of Information Technology and Engineering (SITE) of the University of Ottawa in September 2002 as a Canada Research Chair Professor (Tier 1). He was with the Department of Electrical and Computer Engineering at Queen's University since 1979, where prior to his departure in August 2002 he was a full professor and department associate head. He has served as Editor-in-Chief of *IEEE Communications Magazine* (1995–1997) and IEEE Communications Society Director of Magazines (1998–1999). He is the author or co-author of three books and more than 700 technical papers and 8 patents in the area of broadband packet switching networks, mobile wireless networks, and quality of service over the optical Internet. He was the recipient of the 1989 Engineering Medal for Research and Development of the Association of Professional Engineers of Ontario (PEO).