A Scalable Protocol For Reporting Periodically Using Multicast IP

Ljubica Blazević, Eric Gauthier
ICA, Swiss Federal Institute of Technology Lausanne

Abstract

We propose a protocol that controls the members of a multicast group that send periodically status reports to all members. The protocol, called Multicast Access Protocol (MAP), limits the number of concurrent multicast reports as the group size becomes large. MAP is a decentralised protocol that provides an access control mechanism to an IP multicast group. The protocol supports members joining and leaving the group dynamically as well as changes in the underlying network topology. MAP is a self-configuring mechanism and requires every member to keep only local information independent of the group size without using random timers. We describe the protocol both formally and informally.

Keywords

multicast IP, access control, distributed protocol, self-configuration

1 INTRODUCTION

We address the problem of limiting the number of concurrent members of a multicast group that send periodically status information to all other members. This problem arises in the Scalable Reliable Multicast protocol where each member must report the largest sequence number of the data packet that it has received from each sender to the whole group [1]. The periodical sending of reports does not scale to large groups since the number of reports received by a member is proportional to the number of group members. Multicast IP, as defined in [2], does not control the number of members in a given multicast group that can send data simultaneously. One solution to reduce the number of received reports is for every member to send its report to a server that combines the received reports and send a combined report to all members. However the server becomes a bottleneck as the group size increases. To overcome this limitation Mark Handley [3] proposed to use multiple servers: each server receives the reports from a different subgroup of members and sends a combined report to its members and to the other servers. This scheme does not support well new members joining and leaving the multicast group as

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42

H. R. van As (ed.), *High Performance Networking*

well as changes in the underlying topology since the servers must be statically configured. An adaptive alternative is to use a two-level hierarchy [4] in the following way: each member reports to a local representative member elected using random timers which then sends a combined report to all other local representatives. Each representative sends then a combined report to all its local members. However in the absence of hierarchy this approach requires that each member maintains a table of distance to all other members in order to adjust the random timers. Using a two level hierarchy a member must only keep a table of distance to all other members having the same representative. Also in this case each representative is also required to maintain a table of distance to all other representatives.

We propose in this paper a different approach called Multicast Access Protocol (MAP) that does not require random timers. A member that uses MAP does not keep a distance table to all members but only to a subset of the members. Let $d(i,j)$ be the number of hops that a multicast packet sent by member i goes through before arriving at member j. We assume that $d(i,j) = d(j,i)$. We say that member i is a *neighbor* of member j if $d(i,j) < d(k,j)$ for all members k such that $d(i,k) < d(i,j)$. Thus a member keeps a table of distances only to its neighbors. In the worst case a member keeps a table of distance to all other members. This worst case occurs only if $d(i,j) = d(i,k)$ for all members i,j,k. In general the number of members in the table of distance should be independent of the group size. MAP can also be used in a two-level hierarchy but in this paper we restrict the description of MAP for the case of a flat hierarchy.

MAP relies on the notion of a *grant* as defined in the SMART[5] protocol originally designed for ATM. SMART guarantees that, at a given instant, only one end-system is concurrently sending data on a given connection of an ATM multicast tree. In a similar way, MAP limits the number of concurrent report senders to a maximum number n which is fixed initially. The idea of applying SMART to IP was suggested to the authors by Jean-Yves Le Boudec [6].

To achieve this on multicast IP, MAP builds and maintains n spanning trees of the multicast group members, where n is the maximum number of reports that can be sent at the same time. All spanning trees of a given group are the same except that they can be directed differently. Each spanning tree is rooted and directed inward so that a member needs to keep track of one of its neighbor, called its *parent*. The member of a spanning tree without parent is called *root*. If a member of a spanning tree is a *root* and it is allowed to send a report then we say the member is a *leader*. A member and its neighbors exchange control informations by sending MAP control packets by unicast. Members retransmits state information if the protocol requires it and do not require a reliable transport protocol.

The purpose of a MAP control packet is to change the root of a particular spanning tree and to ensure that every member becomes root of this tree in its turn. When a member becomes root of a spanning tree the protocol

determines if it is the leader of this tree and is thus allowed to send one report. Assuming that no member join or leave the group from time $t = 0$, if $n(i,t)$ is the number of times member i sends a report during $[0,t]$, then MAP ensures that $|n(i,t) - n(j,t)| \leq 1$ for all members i, j, and time t.

MAP constructs the spanning trees using the reports sent by the members. Every host is always free to join or leave the multicast group and the spanning trees are reconfigured accordingly. MAP ensures that adjacent members on the spanning trees are neighbors. The spanning trees are completely distributed and do not rely on any centralised coordinator. The spanning tree is built dynamically as members join and leave the multicast group.

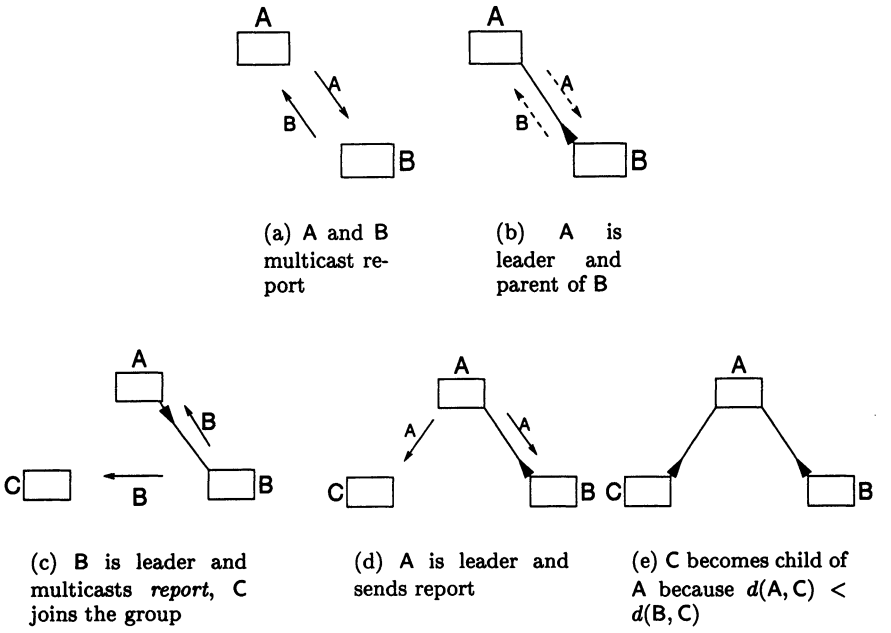
The protocol requires that every member keeps the following state information 4bits per spanning tree and 3 bits per spanning tree and per neighbor. MAP uses only one safety timer that guarantees the spanning trees are loop-frees. A MAP control packet carries $2 + \lceil \log n \rceil$ bits where n is the maximum number of reports that can be sent at the same time. In the two following sections we assume that $n = 1$. The paper is organised as follows: Section 2 gives an overview of the protocol for the case where it allows only one member to send a report concurrently. Then the protocol in the case it allows only one member to send a report concurrently is formally specified in Section 3. Section 4 lists some open issues.

2 OVERVIEW OF THE PROTOCOL

This section describes the general mechanisms of the MAP protocol for the case where only one report can be sent at the same time. A detailed specification of the protocol is given in Section 3.

The construction of a spanning tree is shown in Figure 1. Hosts A and B join simultaneously a multicast group identified by address mcast. A and B start their timer T_1. When their timer T_1 expires A and B are both leaders and multicast a report, see Figure 1(a). A solid arrow shows a report in transit whereas a dashed arrow indicates that the report was received. B receives the report of A and determines that A is its parent since B's IP address is smaller than A's IP address. The parent-child relationship is shown by a filled arrow pointing out of B towards A. A is the leader of the spanning tree A-B, see Figure 1(b). B sends a MAP control packet by unicast to A. Meanwhile host C joins the multicast group and starts its timer T_1. A becomes the child of B and sends a MAP control packet to B. B receives the packet from A and becomes leader and multicasts a report, see Figure 1(c). Immediately B becomes the child of A and sends a MAP control packet to A. C receives the report of B. As soon as A receives the MAP packet from B, A becomes leader and multicasts a report, see Figure 1(d) . C receives the report of A.

Timer T_1 at C expires and C becomes child of A (Figure 1(e)) because $d(\mathsf{A},\mathsf{C}) < d(\mathsf{B},\mathsf{C})$.



(a) A and B multicast report

(b) A is leader and parent of B

(c) B is leader and multicasts *report*, C joins the group

(d) A is leader and sends report

(e) C becomes child of A because $d(A, C) < d(B, C)$

Figure 1 Construction of Tree A-B-C

As more hosts join the multicast group, the spanning tree is dynamically updated as in the previous example.

A MAP control packet contains a 2 bit sequence number.

A member keeps two state variables:

sn a sequence number that takes the values 0,1 and 2, and that take initially value 0.

ln the last value of *sn* at which the member was a leader.

The access control to the multicast tree is shown in more details in Figure 2. The spanning tree consists of five hosts A, B, C, D and E. A is parent of B and C and B is parent of E and D. A is root of the tree A-B-C-D-E. Initially *sn* and *ln* take the value 0, see Figure 2(a). A increments *sn* modulo 3 since no neighbor has its *sn* equal to 1 and 2, see Figure 2(b). A is now the leader since $ln = (sn + 1) \bmod 3$, sends a report and sets $ln := sn$, see Section 3. The state values of A are showed in white on a black background as long as A is a leader. A sends a control packet with *sn* to C, see Figure 2(c). This control packet is a grant [5]. C receives the grant from A, becomes the root, sends a control packet to A to acknowledge the reception of the grant that carries the same sequence number as the grant. C increments its *sn*, becomes a leader and sends a report, see Figure 2(d). C sends a grant to A since it has no other

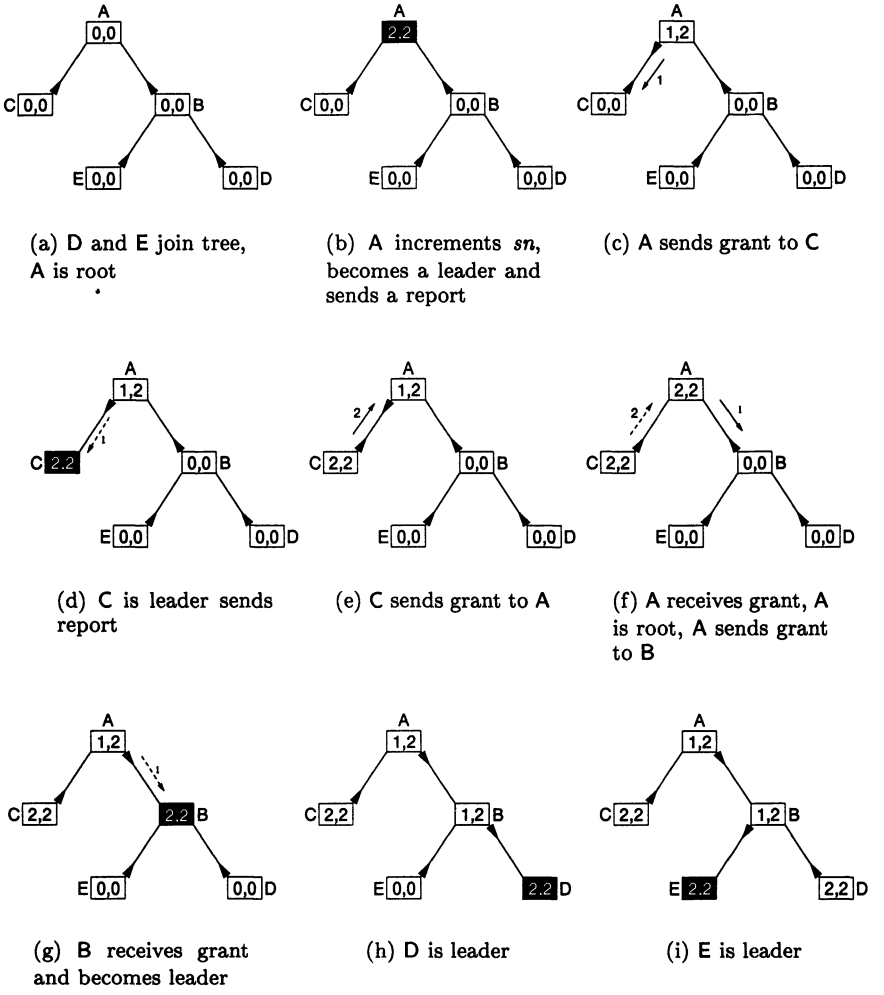


Figure 2 Single Access Control on Tree A-B-C-D-E

neighbor, see Figure 2(e). A receives the grant, becomes the root and sends an acknowledgement to C. A sends a grant to B since B has a sequence number equal to (2+1) mod 3, see Figure 2(f). B receives grant, becomes a root, sends an acknowledgement to A, increments its *sn* twice modulo 3 and becomes a leader, see Figure 2(g). Similarly D and E become leader and send a report, see Figure 2(h) and Figure 2(i). This example shows how the protocol ensures that every member sends a report in turn.

3 SPECIFICATION OF THE PROTOCOL

This section presents a formal specification of MAP for the case where only one report can be sent at a time.

We recall from Section 2 that a member keeps two state variables:

sn a sequence number that takes the values 0,1 and 2, and that take intitially value 0.
ln the last value of *sn* at which the member was a leader.

A member keeps also a safety timer T_1 which is set to value that represents the maximum delay tolerated by a member to send a report. A second timer T_2 is used to resend a control packet to a parent if the grant or its acknowledgement were lost. In addition a member keeps a distance table which consists per neighbor of:

- the IP address of the neighbor.
- d(member,neighbor) in number of hops.
- a parent flag (one bit) that is set if neighbor is parent and else is reset.
- a 2 bit sequence number received in the last MAP control packet from this neighbor.

The four following functions are associated with the distance table.

- neighbor(x) searches the table and sets the parent flag for the first neighbor it finds with a sequence number equal to x and returns true, else it returns false.
- no_neighbor(x) searches the table and returns true if no sequence number equal to x was found, else it returns false.
- reset_parent(x) searches the table, if there is a neighbor with parent flag set and sequence number equal to x, returns true and finds the neighbor with the parent flag set and d(member,neighbor) is minimum, resets this neighbor flag and delete all other neighbors with parent flag set, else returns false.
- delete_parent searches the table for a neighbor with the parent flag set and deletes the neighbor from the table.

Each member that implements MAP is modelled as a finite state machine. A transition has a label of the form: **if** (condition) **then** actions. A transition is fireable if the condition is true. Fireable transitions are executed one at a time. When a transition is executed all its actions are executed as one atomic action. If several transitions are fireable at the same time one is chosen at random. The state machine is shown in Figure 3.

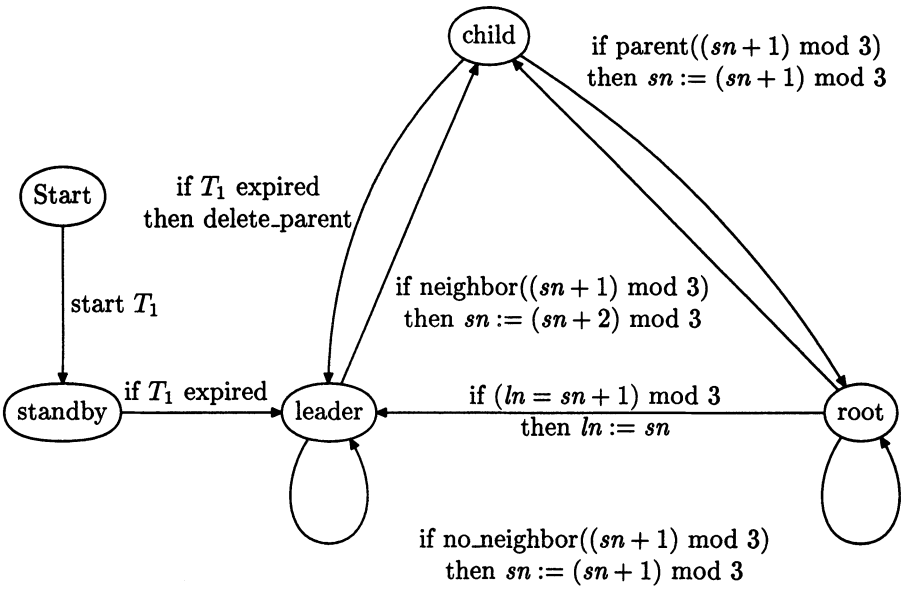


Figure 3 The state machine per spanning tree

State standby. When a new member joins the multicast group, it starts timer T_1 and enters state standby. If a member receives a report while in state standby and its distance table is empty it inserts the sender as a new neighbor with sequence number equal to 0 and a parent flag reset. If the member receives a report and the distance table has one entry, and d(member,sender) < d(member,neighbor), the neighbor is deleted and the sender is inserted as a new neighbor with sequence number equal to 0 and a parent flag reset, else the distance table remains the same. When state is standby and timer T_1 expires, the state changes to leader and only if the distance table is empty the member multicasts a report.

State leader. While in state leader if there is no neighbor in the table with sequence number equal to $(sn + 1)$ mod 3, sn is incremented modulo 3. If the state is leader and there is a neighbor in the table with sequence number equal to $(sn + 1)$ mod 3, sn is incremented twice modulo 3, timer T_1 is started and send a control packet to neighbor with sn and the state changes to child.

State child. If state is child and T_1 expires, the parent neighbor is deleted from the table and the state changes to standby. While in state child if a report is received and d(member,sender) < d(member,parent), the sender is added as a new neighbor in the table with sequence number equal to 0 and a parent flag set. If in state child and there is a neighbor with parent flag set and sequence number equal to $(sn + 1)$ mod 3 then send a packet to the neighbor with sequence number $(sn + 1)$ mod 3 and starts timer T_2. Member

finds the neighbor with the parent flag set and d(member,neighbor) minimum, resets this neighbor flag, delete all other neighbors with parent flag set, sn is incremented modulo 3, the state changes to root. If in state child and timer T_2 expires resend a packet with sn to parent. If in state child and receive packet from parent with sequence number equal to sn, stop T_2.

State root. While in state root if there is no neighbor in the table with sequence number equal to $(sn + 1)$ mod 3, sn is incremented modulo 3. If the state is root and there is a neighbor in the table with sequence number equal to $(sn + 1)$ mod 3, sn is incremented twice modulo 3, send a control packet to neighbor with sn and the state changes to child. If the state is root and $ln = (sn + 1)$ mod 3, $ln := sn$ and the state changes to leader.

4 OPEN ISSUES

MAP presents an alternative to the session message mechanism in SRM [4]. The two approaches should be compared in a flat hierarchy scenario as well as for a two-level hierarchy. It is not clear also which of the two alternatives will be more robust to packet losses. Also many interesting questions remains to be answered about MAP such as the overhead for all members to send one report each as well as the associated latency.

5 CONCLUSIONS

We have proposed in this paper a protocol that controls the members of a multicast group that send periodically status reports to all members. The protocol, called Multicast Access Protocol (MAP), limits the number of concurrent multicast reports as the group size becomes large. MAP is a decentralised protocol that provides an access control mechanism to an IP multicast group. The protocol supports members joining and leaving the group dynamically as well as changes in the underlying network topology. MAP is a self-configuring mechanism and requires every member to keep only local information independent of the group size without using random timers. Assuming that no member join or leave the group from time $t = 0$, if $n(i,t)$ is the number of times member i sends a report during $[0,t]$, then MAP ensures that $|n(i,t) - n(j,t)| \leq 1$ for all members i, j, and time t. MAP was showed to be free of deadlocks by extensive simulations using the model checker SPIN [7],[8]. including in case of message loss, duplication and reordering. An implementation was tested successfully in a LAN environment. A complete proof of correctness of a protocol is left for further study.

REFERENCES

[1] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liv, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," in *Proc.of ACM SIGCOMM'95*, USA, August 1995, pp. 342–356.

[2] S. Deering, "Host Extensionsfor IP Multicasting," Internet RFC1112, Aug. 1989.

[3] M. Handley, "SAP: Session Announcement Protocol," Internet Draft, 1996.

[4] P. Sharma, D. Estrin, and S. Floyd, "Scalable Session Messages in SRM using Self-configuration," submitted to SIGCOMM, 1998.

[5] E. Gauthier, J.-Y. Le Boudec, and Ph. Oechslin, "A many-to-many multicast protocol for atm," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 458–472, April 1997.

[6] J.-Y. Le Boudec, "private communication," .

[7] G. J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, 1991.

[8] G. J. Holzmann, "The Model Checker SPIN," http://netlib.att.com/netlib/att/cs/home/holzmann-spin.html.

6 BIOGRAPHY

Ljubica Blazević received the B.S. degree in 1993 from the Faculty of Electrical Engineering, Belgrade, Yugoslavia. From 1993 to 1996 she worked as a R&D engineer at Institute "Mihajlo Pupin" in Belgrade. She is currently a Research Assistant at ICA, Swiss Federal Institute of Technology Lausanne, where she is engaged in research on multicast routing and traffic control.

Eric Gauthier is born in Zurich, Switzerland, in 1970. He received the B.Eng. degree in electrical engineering from McGill University, Montreal, Canada, in 1992, and the Master degree from INRS-Télécommunications, Verdun, Canada, in 1995. He is currently working toward the Ph.D. degree at the Swiss Federal Institute of Technology Lausanne.