Implementation and Performance Evaluation of Multicast Control Protocol

Takeshi Takahashi[†], Miikka Tammi[†], Heikki Vatiainen[†], Rami Lehtonen[‡], Jarmo Harju[†]

[†]Tampere University of Technology
Institute of Communications Engineering
P.O.Box 553 FIN-33101 Tampere, Finland
{takahash, tammi3, hessu, harju}@cs.tut.fi

[‡]TeliaSonera Corporation
Hatanpään valtatie 18, 33100 Tampere, Finland
rami.lehtonen@teliasonera.com

Abstract

Multicast Control Protocol has been proposed to control multicast traffic transparently from the sources and the receivers. We have been developing and implementing this protocol, and this paper introduces the framework of our implementation in the Linux environment and proposes some performance enhancements brought by implementation specific features. This paper also shows the performance evaluation of our implementation. Finally, we clarify the effectiveness and the scalability of this protocol.

1. Introduction

Multicast enables one-to-many communication in data networks. Compared to unicast, it is very efficient from the network resource viewpoint, and plenty of applications have been deployed over IP network. However, the current IP multicast architecture has insufficient dynamic control capabilities over the multicast networks. The lack of admission control exposes crucial issues to be solved. For instance, multicast network without control is vulnerable to the Denial of Service (DoS) attacks [1] from malicious nodes. In the IP multicast model, any host can transmit packets to a multicast group without registering itself with the group. Therefore, network operators are still waiting for better multicast control schemes before offering more services.

To cope with this problem, Multicast Control Protocol (MCOP) [2, 3] was proposed that facilitates multicast network management. MCOP provides access control to multicast networks within one domain or autonomous system (AS) by remote access-lists and controls multicast traffic by selectively filtering the Internet Group Management Protocol [4] (IGMP) messages and UDP multicast traffic. Moreover, MCOP is transparent from multicast clients. The fundamental functionality of MCOP is multicast receiver control and multicast source control. The former controls the reception of multicast traffic coming into the administrative network while the latter controls the transmission of multicast traffic going out of the administrative network. This feature may be used to block misbehaving hosts from sourcing traffic to certain multicast group. MCOP allows gradual, group and network specific multicast network deployment.

As a related work, Internet Group membership Authentication Protocol (IGAP) [5] provides IGMP's group membership control between hosts and their first-hop routers, with the addition of user authentication and accounting. Its control policies are managed by centralized database, which enables dynamic policy changes. However, it is still vulnerable against DoS attacks since it does not support source control. Moreover, it requires specific features to all the hosts. Another way to control multicast receivers and sources is static router configuration such as Cisco's access lists [6]. However, since the database is stored in the local router, this scheme is not suitable for dynamic policy changes and is not scalable in the larger network. Compared to those schemes, MCOP controls multicast traffic transparently from the sources and receivers with centralized database.

In this paper, we show our implementation of MCOP and evaluate the performance of the protocol. In Section 2, we give an overview of MCOP, in section 3, we describe the framework of our MCOP implementation, in Section 4, we evaluate the performance, and finally, we summarize this paper in Section 5.

2. Overview of MCOP

This chapter describes the basic operation of MCOP. MCOP requires Multicast Control Client (MCC) and Multicast Control Server (MCS) for its operation. MCS contains the database of the multicast policies while MCC filters multicast traffic according to the policies from MCS.

2.1. Multicast Control Server

MCS contains information about MCOP controlled multicast addresses and valid receivers and sources for MCOP controlled groups. MCC validates the group members against the information stored within MCS. If information changes for MCOP controlled group in MCS, MCS is re-

sponsible for updating the information to MCCs that have validated the group status earlier.

The fact that MCS contains information about configured multicast groups prevents unwanted multicast state to be appearing in the multicast routers, since the multicast tree is not created if the multicast information is not found from the MCS.

2.2. Multicast Control Client

2.2.1. Initialization

MCC controls multicast traffic before the traffic is given to multicast processing or before the MCC processes IGMP packets. Multicast control is applied to a specific multicast address range that is specified by MCS. The MCC behavior discussed within this chapter is applied only to the multicast groups that pertain to the MCOP controlled multicast address space. MCOP processing can be separated from the router operation and implemented as a transparent filtering bridge between router and directly connected hosts.

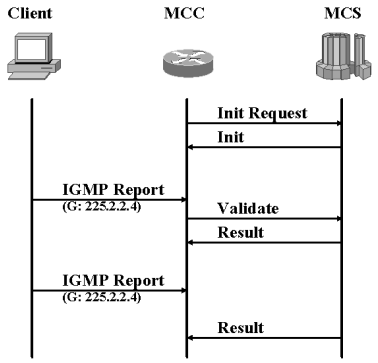


Figure 1. MCOP messages

Fig. 1 shows the message exchange procedure between MCC and MCS so that they can control multicast traffic. At the initialization phase, MCC is responsible for the connection to MCS. MCC informs MCS about all its directly connected networks by transmitting Init Request message and, in response, receives Init message that contains all MCOP controlled multicast addresses from MCS. After this initialization phase, MCC is ready to filter IGMP messages as well as multicast traffic.

When MCC receives IGMP Report transmitted from a directly connected host to an MCOP controlled multicast group, MCC transmits Validate message to MCS in order to check the validity of the receiver, and it, in response, receives Result message that contains valid receiver and source information. This validation procedure is called remote validation. According to the policy information obtained from the Result message, the MCC either forwards or drops the the IGMP Report (local validation). Here, upon receiving Result message, MCC creates multicast group information cache by storing the information obtained from remote validation (cache entry update). Cache entry update is mandatory so that MCC does not have to redo remote validation. Once the specific clients are allowed to transmit packets according to the Result message, MCC continues to forward IGMP Reports transmitted from the receiver unless the controlling policy changes for the receiver. If the receiver is not found from the list of valid receivers, MCC starts discarding IGMP Reports.

Likewise, when MCC receives multicast traffic from a directly connected host to an MCOP controlled group, MCC performs remote validation and cache entry update. If the multicast group cache entry exists in MCC, the remote validation and cache entry update are omitted. If the source is found from the valid source list for this multicast group, the traffic can be forwarded for further processing (multicast traffic forwarding). MCC continues to forward the multicast traffic unless the group status changes for the source. If the source is not found from the list, the MCC discards the multicast traffic originated from the source and destined to this multicast group. MCC continues to discard the traffic unless the policy changes for the source.

2.2.2. Update and Reset

The locally stored multicast control information cannot be changed except of updating and resetting.

The policy updates are informed by unsolicited Result messages transmitted by MCS. Updates are invoked when there are changes to valid sources or receivers for that multicast group. The change in group member information affects directly the management of active receivers and sources, which are either set to passing or filtering state by this message. If the policy is changed so that some MCOP multicast clients are not allowed to receive specific multicast traffic any more, the MCC generates IGMP Leave and transmits it to the upstream multicast router on behalf of the clients. This IGMP Leave injection feature minimizes the policy changing delay. The MCC discards further IGMP packets for this multicast group originating from the receiver.

The policy resets are invoked by timer expiration. When MCC judges that MCOP controlled clients are neither receiving nor transmitting, i.e. inactive, MCC transmits Reset message to MCS so that MCC can delete all the controlling information concerning the MCOP controlled clients provided cache entry timer is not set. The judgement for inactivity is based on the timeout expiration, i.e., if MCC doesn't receive any IGMP packets from these clients for a certain timeout duration and if MCC does not receive any UDP packets from these clients for a certain timeout duration, MCC judges that those clients are not active any more, and transmits Reset message. If the MCC receives IGMP messages or UDP multicast traffic from the client that is deleted from local database, it will perform remote validation from the beginning. In case that the cache entry timer is set, MCC does not immediately send Reset message upon judging its inactivity and, instead, awaits the clients' activity for another duration specified by cache entry timer before sending Reset message. The value of cache entry timer is specified by Init message.

2.3. Example Service Scenario

MCOP is intended for network management purposes as we mentioned above, and one possible usage of MCOP is to connect the control mechanisms together with some application and allow billing of multicast services such as streaming. Fig. 2 shows the example network. First, we have one client that is listening specific multicast traffic. To avoid unnecessary traffic flooding, the network may utilize virtual LANs, which may enable individual control of each receiver in switch based network.

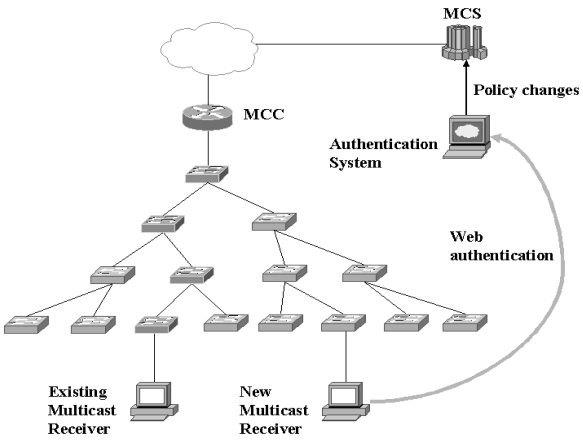


Figure 2. Billing System

When a new client wants to join a specific multicast service like TV broadcasting, the client has to authenticate himself via web authentication system. The web authentication system sends the receiver policy change to the MCS. Then MCS transmits unsolicited Result message to MCC to start the forwarding of the multicast traffic to the client.

3. Framework of MCOP Implementation

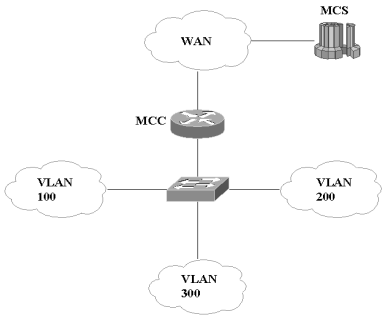


Figure 3. Our implementation network

Fig. 3 shows our experimental network. Our MCOP implementation is based on Linux environment, and our software shall become open source [7]. In this network, three virtual LANs are connected to the router, where MCC is also implemented. This MCC has TCP connection with the MCS that resides in the WAN network. Although the location of MCS is not restricted as long as MCC has TCP connection with MCS, the closer MCS is to MCC, the faster the validation procedure is. In this network, MCS is two hops away from MCC. Here, we utilize IGMP version 2 [4] over IP version 4 network.

3.1. Packet filtering

MCC controls UDP multicast traffic and IGMP traffic before their routing decisions are made. From the software point of view, MCC functions just after IP packet processing. In our implementation, the IP packet processing is done by the tool "iptables" [8], which makes it possible to process IP header of packets so that it can filter (ACCEPT, DROP, or QUEUE) packets. When the filtering policy is ACCEPT, packets are further processed for the routing decision. When the filtering policy is DROP, packets are simply discarded. When the filtering policy is QUEUE, packets are lifted up to the MCC application.

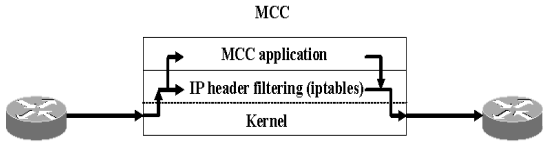


Figure 4. MCC layers

Therefore packets are processed in three different layers : kernel, IP header filtering (iptables) and MCC application (Fig. 4). Incoming packets are caught by kernel, and iptables processes the IP header of these packets. Although iptables handles IP header processing including filtering as a part of kernel, we describe it as the uppermost layer of kernel since it provides user application interfaces.

When a packet destined to a new multicast group arrives to MCC, iptables lifts it up to the MCC application layer since no filtering policy for this multicast traffic is specified yet. Then the MCC application invokes remote validation by communicating with its MCS. Those packets arriving to the MCC application during the remote validation are just discarded in the iptables layer. Upon finishing the remote validation, the MCC application sets new policies directly to the iptables layer according to the Result message so that any traffic transmitted to the same multicast group can be filtered in the iptables layer. This multicast traffic is then verdicted by the iptables instead of MCOP application layer.

When an IGMP packet destined to new multicast group arrives to MCC, iptables lifts it up to the MCC application layer as in the case of multicast UDP packets. Then

the MCC application invokes remote validation by communicating with MCS. Upon finishing the remote validation, the MCC application sets new policies directly to the iptables according to the Result message so that IGMP packets related to the same multicast group can be filtered by the iptables. Here, only IGMP Report can be caught by its destination address. IGMP Leave and IGMP Query have the values of ALL-SYSTEMS.MCAST.NET (224.0.0.1) and ALL-ROUTERS.MCAST.NET (224.0.0.2) respectively in the destination field of IP header regardless of the multicast group address field of the IGMP header. Therefore MCC is required to catch all the IGMP Leaves so that MCC can keep track on which clients are still active in the specific multicast group. Yet, MCC does not have to inspect IGMP Query. Therefore, IGMP Queries are simply passed without any specific action.

As can be seen, MCC application layer catches only the first IGMP packet and the first multicast UDP packet of any multicast group to invoke remote validation. MCC application layer never catches traffic from iptables layer after the remote validation since local validation is performed by the iptables according to the access list instead. Therefore, the performance of local validation depends on the filtering performance itself. In this case, we can expect the same scalability level of the filtering software which has been reliable in the view of scalability for years. Here, the iptables filters only the traffic coming from directly connected multicast clients. No packets coming from outside the network will be filtered by the iptables.

In this implementation, in order to enhance the performance, we developed some implementation specific features: IGMP message injection, specific address block ordering, and suppressed packet inspection.

3.2. IGMP message injection

Upon receiving Result message (including unsolicited Result message) from MCS, MCC analyzes the message and stores the objects obtained from the Result message. Simultaneously, the MCC modifies the iptables configuration to take load off the application program. If the message indicates the status change of one receiver to ”deny”, the MCC transmits an IGMP Leave on behalf of the multicast client. This feature significantly reduces the policy updating delay and is mandated by the draft.

Likewise, when MCC receives unsolicited Result message that indicates the status change of one receiver as ”allowed”, the MCC transmits an IGMP Query to the clients. If the client still wants to join the multicast group, the client immediately transmits the IGMP Report to the router. Although this is not specified in the draft, it minimizes the policy updating delay.

3.3. Specific address block ordering

In our MCC implementation, the most time consuming functionality is the local database updating. To minimize the local database updating time, we established a specific rule for the Group Member object of Result Message. In our implementation, Group Member object is constructed per interface policy though the draft allows the Group Member object to contain information for several interface policies. Moreover, the order of the address blocks in the Group Member object must be in ascending order so that further sorting can be avoided inside MCC.

Since MCC can expect that the default filtering rule of the whole interface appears at the end of the address blocks of the Group Member object of Result message, MCC does not search the default interface policy information from the Group Member object.

More specific network filtering rule must appear in the earlier stage of the filtering table. Since the blocks in the Group Member object were already sorted in ascending order, we can subsequently add that information to the filtering table instead of sorting the blocks.

3.4. Active client inspection

In order to judge the inactivity of MCOP multicast clients, MCC is required to manage timer values. Therefore, upon receiving a packet from validated MCOP multicast clients, MCC is required to update the timer value. Compared to the normal routers that simply inspect the source address and destination address provided the packet is destined to the router itself, this feature has too much burden for each packet transaction. Although these inspections are acceptable for IGMP packets since these packets are arriving at MCC in each constant interval, they are undesirable for UDP packets since UDP packets are arriving at MCC successively with very little packet interval.

Our implementation does not inspect the arrival of UDP packets upon receiving each UDP multicast packet but inspect it in each constant time, i.e. window time (default value is 1 second). When UDP packets are arriving, iptables just filters the packet according to the policy without checking and updating timer value. Instead, MCC checks the packet counters once per each window time. If MCC received packets during the last window time, MCC updates the timer value concerning to the multicast traffic. Note that MCC application does not catch the packet but simply inspects the packet counters. This procedure reduces the burden of MCC and reduces the packet transaction time when receiving UDP packets.

4. Evaluation

4.1. Packet transaction time

Here, we evaluate the packet transaction time in the application layer at MCC and MCS. Our implementation is based on Linux boxes. MCC is a Linux box composed of Celeron 633 MHz processor and 320 MB memory while MCS is a Linux box composed of the same processor as the MCC and 256 MB memory.

4.1.1. IGMP packets (Receiver case)

When MCC has multicast clients in its directly connected networks, MCC must control IGMP packets transmitted from them. Therefore, transaction time for one IGMP packet was measured here. Here, 500 different IGMP packets were transmitted from the directly connected clients. Those 500 IGMP packets were intercepted by the MCC, which invoked the remote validation with its MCS.

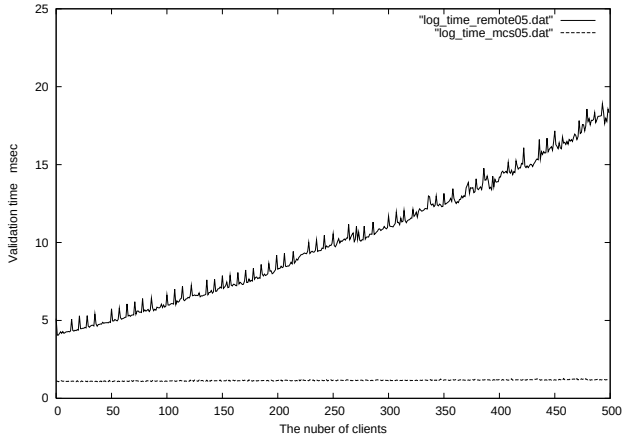


Figure 5. Remote validation time for IGMP packets

The solid line in Fig. 5 shows the remote validation time for each IGMP packet. The X-axis shows the number of IGMP packet, which is equivalent to the amount of stored Group Member object information. The Y-axis shows the transaction time for an IGMP packet. This time is measured from the time a packet arrives at MCC application to the time the packet leaves MCC application including MCOP message transaction time inside MCS and network delay between MCC and MCS. In this evaluation, the round-trip time between MCC and MCS was around 0.43 millisecond. As can be seen, the remote validation time is slightly increasing depending on the number of the clients. This is because MCC has to search policy information in the updating situation. The remote validation time is formulated in equation (1).

$$f(x) \simeq 0.0275x + 4.06[msec] \quad (1)$$

Here, x denotes the number of clients while $f(x)$ denotes validation time. Although the remote validation time is increasing as the number of clients increases, the inclination of the validation time is rather small. Moreover, the remote validation happens only one time per a new client for a multicast group. Therefore, MCC is working in a scalable way in the case of IGMP packet transaction.

The broken line in Fig. 5 shows the MCOP message transaction time inside MCS during the remote validation. This time was measured from the time that MCS application received Validation message to the time that MCS returned Result message to MCC. As can be seen, the packet transaction time of MCS does not change. Since MCS database contains only the information of the allowed hosts, the amount of the information inside the database is not proportional to the number of multicast clients. Moreover, since MCS is storing policy information by aggregating addresses with network mask value, the amount of the information stored in the database is not proportional to the number of the allowed multicast clients. Therefore MCS shows good scalability.

After the first set of 500 IGMP packets, the same 500 IGMP packets were transmitted from those directly connected multicast clients. Since MCC has policy information for those clients and MCC has already set policy in the iptables layer, none of those packets are caught by the MCC application layer and local validation was performed by the iptables.

4.1.2. UDP packets (Sender Case)

We also evaluated the packet transaction time for UDP packets. The result were quite identical to Fig. 5. Similar to the case of IGMP packet transaction time, the remote validation time is slightly increasing depending on the number of the clients. This is also because MCC has to search policy information in the updating situation. The remote validation time is formulated in equation (2).

$$f(x) \simeq 0.0275x + 3.74[msec] \quad (2)$$

Here, x denotes the number of clients while $f(x)$ denotes validation time. Equation (1) and equation (2) are similar though the intercept value is slightly different. The slight difference comes from the difference of the header fields. MCOP needs to inspect IP header for UDP packet while it needs to inspect IP header and IGMP header for IGMP packet. Although the remote validation time is increasing as the number of clients increases, the inclination of the validation time is rather small. Moreover, the remote validation happens only at the start of traffic in a multicast group. Therefore, MCC does not have any problems for the scalability in the case of multicast UDP packet transaction.

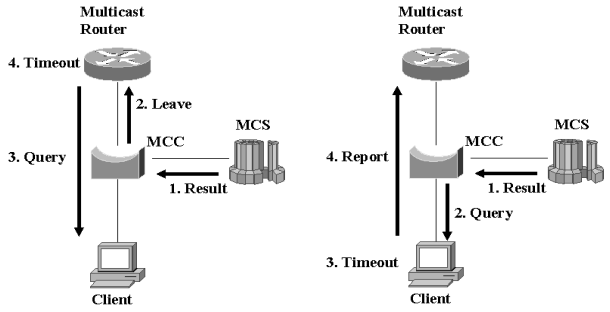
Similar to the case of IGMP packet transaction, the packet transaction time inside MCS is constant. Likewise, local validation was performed by the iptables and our MCC application does not catch any packets. Therefore, the performance follows the iptables though it is not the main concern of this paper.

4.2. Policy updating delay

When multicast controlling policy is changed, MCS transmits Result message to MCC, which immediately changes the filtering policy according to the message. Fig. 6a shows the case in which the policy is changed so that one specific client cannot receive the multicast traffic any more. Upon receiving Result message, MCC transmits IGMP Leave to the router on behalf of the client. This

feature of MCC is designed to minimize the policy updating delay. Here, MCC can be implemented over the multicast router as our implementation is. Upon receiving IGMP Leave, the router transmits several (Default: 2) [4] Query messages and stops forwarding further multicast traffic unless any IGMP Reports are received before the response timer(Default: 1 second) [4] of the last IGMP Query expires. Here, we define the policy changing delay as the time after MCC changes its database and until clients do not receive any multicast traffic. In our implementation, since we configured the router so that it transmits two Queries with Max Response time of 0.5 second, the total policy changing delay was 1 second. Here, the packet transmission delay in the network can be ignored since it was less than a millisecond and was very tiny value compared to the IGMP protocol timers as mentioned above.

Since the router does not have to wait for a certain amount of time specified by Group Membership Interval (default: 260 seconds) [4] to stop forwarding multicast traffic, the IGMP Leave injection feature significantly minimizes the policy updating delay. However, since multicast multimedia receiver usually has some decoding buffer, the decoding process continues for a while until the decoding buffer becomes empty. For instance, VLC [9] sets 300 millisecond buffering time as a default value.



a) IGMP Leave b) IGMP Query

Figure 6. IGMP message injection

Fig. 6b shows the case in which the policy is changed so that a specific client is allowed to receive the multicast traffic, then MCC transmits IGMP Query to the multicast client on behalf of the router. Upon receiving IGMP Query, the client has to wait for a certain amount of time that is selected from the range between 0 to Max Response Time (Default:1 second) [4]. Then it transmits IGMP Report to the multicast router. Here, we define the policy changing delay as the time after MCC changed its database and until clients start receiving the multicast traffic. In our implementation, since MCC transmits IGMP Query with default Max Response Time value, the total policy changing delay was 0.5 second in average. Here, the packet transmission delay in the network can be ignored since it was less than a millisecond and was very tiny value compared to the timeout value.

Since the router is not required to wait for a certain amount of time specified by Query Interval (default: 125 seconds) [4] to receive IGMP Report from multicast clients, the IGMP Query injection feature significantly minimizes the policy update delay. However, since multicast multimedia receiver usually needs to buffer some packets before decoding, the actual decoding process starts after a couple of seconds depending on the decoding buffer size.

5. Conclusion

This paper introduced the framework of our MCOP implementation in Linux environment and proposed some implementation specific features that enhance the MCOP performance. Our evaluation clarified that MCOP is working effectively and in a scalable way. MCOP is traffic controlling protocol and is able to cooperate with plenty of other protocols such as authentication protocols and security protocols. It is also possible to establish fault tolerant system so that we can avoid MCS to be the single point of failure.

Our current implementation is based on IGMP version 2 over IP version 4 network. Since IGMP version 3 supports additional features, some modifications are necessary in order to make our software function correctly. Especially, source filtering feature of IGMP version 3 enables MCOP to control multicast traffic by sender's source address. We will implement those features as our future work.

References

[1] Felix Lau, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajkovic, "Distributed Denial of Service Attacks," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2275–2280, October 2000.

[2] R.Lehtonen, J.Soini, J.Majalainen, H.Vatiainen, and M.Tammi, "MCOP operation for first hop routers," *Internet draft draft-lehtonen-mboned-mcop-operation-00.txt*, November 2003, Work in progress.

[3] H.Vatiainen, R.Lehtonen, and J.Soini, "Multicast Control Protocol (MCOP) over COPS," *Internet draft draft-vatiainen-mcop-cops-00.txt*, December 2003, Work in progress.

[4] W. Fenner, "Internet Group Management Protocol, Version 2," *RFC 2236*, November 1997.

[5] Tsunemasa Hayashi, Daisuke Andou, Haixiang He, and Teruki Niki Wassim Tawbi, "Internet Group membership Authentication Protocol," *Internet draft draft-hayashi-igap-03.txt*, August 2003, Work in progress.

[6] Jeff Sedayao, *Cisco IOS Access Lists*, O'Reilly & Associates, Inc, 2001.

[7] "MAD/TUT Project," *http://www.atm.tut.fi/mad/*.

[8] "The netfilter/iptables project," *http://www.netfilter.org*.

[9] "VideoLAN," *http://www.videolan.org*.