# Using Multicast-SNMP to Coordinate Distributed Management Agents

J. Schönwälder
Department of Computer Science
University of Twente
P.O. Box 217, 7500 AE Enschede
The Netherlands

**Abstract**

*This paper presents a mechanism that allows to implement coordination primitives for cooperating management agents based on the Simple Network Management Protocol (SNMP) [12]. The multicast extensions presented in this paper allow to maintain group information of cooperating SNMP agents. We also show how an election algorithm can be implemented by multicasting SNMP messages. Election algorithms can be seen as a base functionality for the implementation of fully distributed management applications.*

## 1 Introduction

Different approaches to build distributed network management systems have been proposed in the last recent years. Early work in this area has focused on the standardization of management functions which are build into specialized agents and can be used by management applications to distribute some of the processing load. Examples of this approach can be found in both, the OSI and the Internet network management worlds (e.g. the RMON MIB [13] or the OSI event report and log control functions [6, 7]).

Standardizing management functions is a difficult task because the designers have to consider some important trade-offs. One the one hand, it is desirable to add powerful parameters so that the management system has enough flexibility to describe and select the required services. However, accepting too much options makes the implementation of the management functions complex although in most cases only a subset of the available functions are actually used.

The Management By Delegation (MBD) model [15] is a completely different approach to distributed network management because it allows to assign management functions dynamically. This approach has several advantages for a great number of advanced management applications. For example, you can implement and distribute site specific policies without being restricted to predefined parameter sets. The MBD model also allows to dynamically locate management functions at different nodes in the network depending on e.g. changes of the network topology or the current daytime. Some criteria which kind of application may benefit from decentralized control and intelligence are discussed in [9].

## 2 Organizational Models

While the MBD model introduces a very powerful mechanism, little discussion has so far been done on the underlying organizational structure. Many distributed management applications are based on a hierarchical organization where lower level functions (e.g. data aggregation or threshold monitoring) are delegated to run "near" the network elements that generate the statistics (figure 1). Higher level data is send up the management tree while control operations are send in the opposite direction [1]. Hierarchical models can be mapped on existing network topologies easily which makes them attractive and easy to understand.

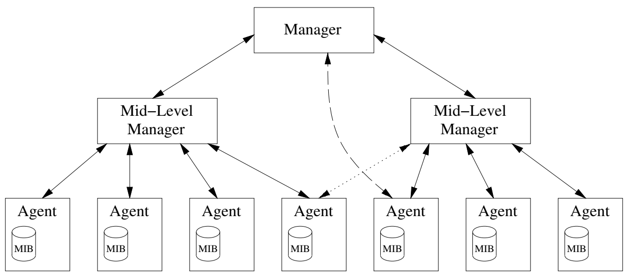


Figure 1: Hierarchical model.

Figure 1 also shows some possible variations that can be found in hierarchical models. For example, it might or might not be allowed to exercise direct control over a device from a higher level manager once a fault has been reported up through the management hierarchy. A similar question is whether it is allowed to access a single node in the hierarchy from two superior nodes or if the system enforces a strict hierarchy.

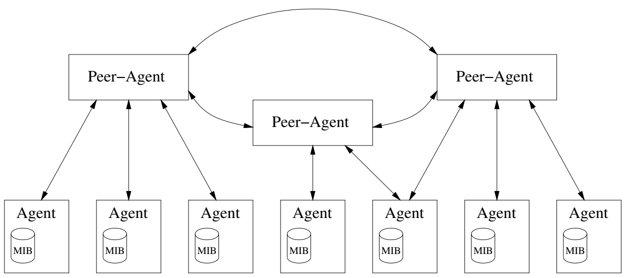


Figure 2: Two level peer model.

In contrast to the hierarchical organization, we propose to use a peer like organizations wherever possible (figure 2). Based on this organizational model, we implement network management applications as distributed applications executed by cooperating management agents.

Distributing management tasks to a number of cooperating agents provides some benefits. The most important one is the ability to replicate functions to increase the fault tolerance of the management system itself. The placement of non-replicated network management functions is a challenging problem because you have to consider all kind of network failures. A good decision for the normal case (e.g. a lightly loaded machine on a fast network) might become a bad choice in case this network gets swamped with large, broken frames. Replication allows to build some fault tolerance into the management system without worrying about the right place to execute management applications.

Another benefit of using cooperating agents in a distributed management system is the ability to implement load-balancing algorithms. Once a group of cooperating agents is established, you can easily move management scripts from an overloaded agent to another member of the group. It is also possible to use the recursive form of delegation where management procedures move like worms from agent to agent in order to monitor or control the system.

It is also possible to combine the hierarchical and the peer model described above by implementing the nodes of a hierarchical structure using a set of cooperating peers. This approach allows to combine the advantages of both models.

However, to implement a network of cooperating management agents, you need coordination primitives that are currently not part of standard management protocols. In this paper, we show how multicast SNMP messages can be used to establish and support groups of cooperating management agents. An election algorithm is used to select a master agent which has the responsibility to supervise the execution of delegated management scripts. The election algorithm can take care of network partitions due to link failures.

Section 3 introduces the environment which we use to experiment with distributed network management applications. Section 4 shows how we use multicast SNMP messages to establish group membership information between cooperating agents. We further explain how we adapted a simple election algorithm to fit into the Internet network management framework. We conclude with a brief discussion of related work in section 5.

## 3 An Environment for Distributed Network Management

The architecture of our prototype is based on the concept of autonomous management agents that cooperate to solve management tasks. Our implementation follows the management by delegation paradigm [15]. Figure 3 shows the internal structure of our peer agents. It is divided into three parts. The upper part contains a delegation server, a delegation client and a storage for management procedures. A management procedure is usually part of a management application which can be delegated to other agents. For example, a management procedure may define a policy which controls routing changes to take advantage of varying tariffs of different network providers. The delegation server allows to load management procedures into the agent while the delegation client can be used to distribute management procedures to remote agents.

The middle part of figure 3 comprises the runtime environment which is used to execute management threads[1]. Management threads can be seen as instantiations of management procedures. The lower part of figure 3 contains a network management agent and manager (dual-role entity). The embedded manager is used to access remote management information while the embedded agent provides access to local MIB information. Management threads can access the local MIB and the delegation client to create new (remote) management procedures/threads. An important feature of our architecture are symmetric interfaces. They allows us to establish arbitrary topologies of cooperating agents.

Our prototype is build around the Internet network management framework. The manager and agent

[1]The term thread is used in a conceptual sense - our current implementation is event driven.

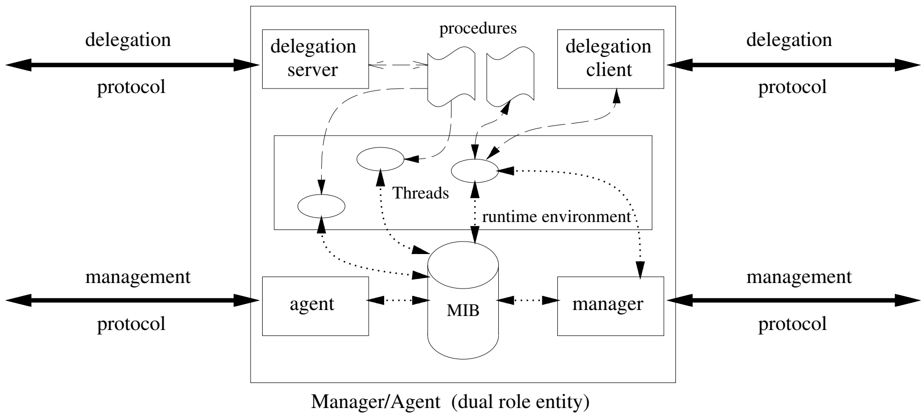


Figure 3: Peer agent architecture.

modules support SNMPv1 and SNMPv2. The runtime environment is based on the Tool Command Language (Tcl) [10] and allows to send SNMP requests to other agents and to manipulate the local MIB. There are also a number of Tcl commands to access services provided by other Internet protocols like ICMP, NTP, DNS as well as some well-known SUN RPC services [11]. The delegation server interacts with the delegation client using the HTTP protocol [2] which makes transfers of large management procedures efficient.[2]

The implementation of a software environment which allows to experiment with different organizational models requires to solve a number of problems (e.g. access control, authentication, resource control, naming, group management, synchronization). In this paper, we focus on the problem of group management, which covers the mechanism needed to establish a group of cooperating agents and to select a master within this group, who is responsible to provide synchronization and decision support.

## 4 Coordinating Agents using Multicast-SNMP

Distributed systems based on cooperating processes require mechanisms to make the participating processes known to all group members. Once the list of group members is known, one can start to implement algorithms which solve problems by exchanging messages. Group communication protocols or multicast protocols[3] are frequently used for this purpose because they provide the programmer with a simple mechanism to exchange information between all members of a group. There are a number of group communication protocols with different characteristics described in the literature [5]. Most of them are build on top of simple message passing protocols.

The Internet protocol suite includes a multicast extension [4] which in contrast to other multicast protocols focuses on world wide multicast routing problems. The protocol itself does not ensure reliability or message ordering. Instead, it supports fast and scalable message delivery to multiple recipients over the whole world. It is supported by many modern operating systems today.

Although IP multicasts lack some power, we feel that IP multicasts can very well serve as a mechanism to implement efficient group communication in a distributed network management system. We therefore started experiments with SNMP on top of IP/UDP multicasts. The reason to use SNMP for group communications is obvious: We can reuse an already existing protocol stack which is needed in our agents anyway. The drawback of this approach is that we limit our distributed management system to use SNMP over IP/UDP. Our management system also relies on IP multicast routing in addition to IP routing. We don't consider these restrictions any serious limitation given the fact that IP multicast routers are becoming more and more reliable.

SNMP is a request/response protocol with the exception of trap messages, which do not trigger responses. This makes SNMP trap messages well suited to be send to a potentially large number of recipients. On the other hand, SNMP is a protocol with a high overhead because each MIB variable carried in a PDU is addressed by a complete object identifier. This can easily lead to PDU sizes around 1500 bytes for PDUs that contain only 60 MIB variables.

[2] Using HTTP allows us to use existing HTTP servers as a storage system for management procedures.

[3] The term broadcast protocol is sometimes used in the literature.

## 4.1 Group Membership

Our first goal is to establish a group membership table using SNMP trap messages encapsulated in IP/UDP multicast messages. The protocol is very simple: Every agent periodically sends a `mcAliveTrap` trap message ($M_a$) to show that he is alive. This message is send to a well known multicast address every $T_a$ seconds.

Every agent which receives multicast SNMP traps builds a MIB table containing all locally known group members. Every $T_e$ seconds, all group member are marked expired if the last alive message was received more than $T_e$ seconds away. An entry in the group member table is discarded if there was no alive message during the last $T_d$ seconds. Obviously, $T_d$ should be much larger than $T_e$.

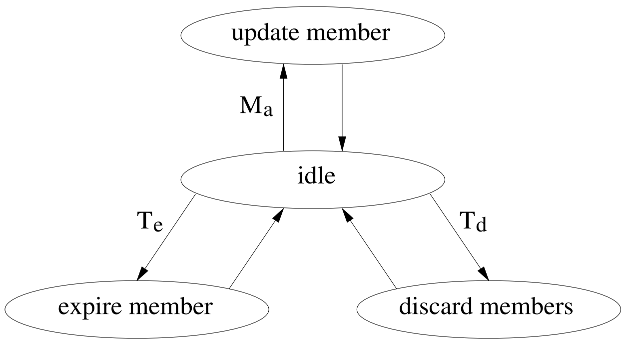


Figure 4: Group management protocol.

Figure 4 shows the state diagram for this protocol. The table containing information about group members can be used to delegate scripts to other management agents or to migrate management threads to agents which are better suited to execute a given thread. Using a group communication protocol avoids the need to configure new agents because they will be recognized automatically once they join the IP multicast group and start to send `mcAliveTrap` trap messages.

## 4.2 Master Election

The next desirable feature is the election of a master agent, which can be used to synchronize operations or to aid in resolving conflicts among cooperating agents. Therefore, we added an election algorithm which guarantees that every group of peer agents has a known master agent. This algorithm ensures, that all active members of a group agree on a single master agent.

Every agent starts with an empty member table and a timer $T_p$. It collects `mcAliveTrap` trap messages ($M_a$). In addition, it also accepts `mcMasterTrap` messages ($M_m$) which are send by the master agent. Once an agent receives a $M_m$ message, the timer $T_p$ is stopped and the peer enters the idle state discussed in section 4.1. Otherwise, if the timer expires and there was no $M_m$ message, the peer starts a new election by changing its state and sending a `mcPanicTrap` message ($M_p$). The agent now starts a timer $T_m$ and waits for votes. The timer $T_m$ is stopped as soon as the peer receives a vote `mcVoteTrap` ($M_v$) send by a peer with a higher number. In this case, the peer enters the idle state again. Otherwise, if the peer did not receive a vote with a higher number, the peer declares itself to be the master and starts to send `mcMasterTrap` messages ($M_m$).

An agent in the idle state which receives a `mcPanicTrap` message ($M_p$) enters the send vote state and starts the timer $T_m$. After sending the `mcVoteTrap`, the agent starts to collect incoming votes. The agent goes back into idle state if the highest number of all received votes is higher than the id of the agent. Otherwise, the agent will become the master if it does not receive a response with a value higher than the agent id.

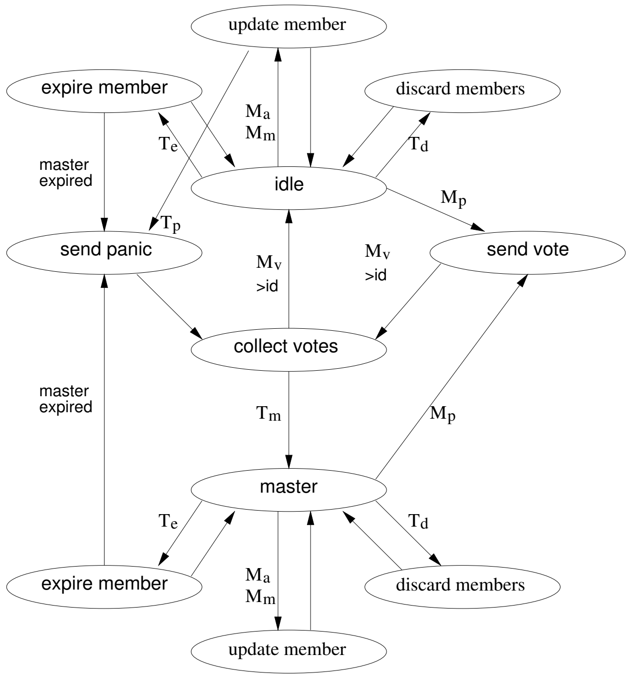


Figure 5: Election protocol.

Figure 5 shows a slightly simplified version of the state machine. Some possible state changes are not listed which can happen if agents send arbitrary messages.

The election procedure can be enhanced to count the number of received votes. In this case, the transition into the master state is only allowed if the number of votes fulfills a quorum criteria, e.g. you must have more than 50% of the votes. This ensures that network partitions will not result into multiple masters running in different network partitions. However, this requires that the discard timeout values are larger than the lifetime of the network partition.

### 4.3 Implementation

The implementation of the algorithm described above is straight forward. The required lower level changes to the SNMP protocol engine add the ability to send/receive SNMP messages to/from IP/UDP multicast addresses. The effort to implement the state machine depends largely on how well timers are supported in the target environment.

Our implementation provides a MIB interface which allows to access the current state of the group management and election protocol described above. The heart of this MIB is a table called `MbdMemberTable` which includes an entry for every group member (figure 6).

```
MbdMemberEntry ::= SEQUENCE {
    mbdMemberGroup          Integer32,
    mbdMemberId             Unsigned32,
    mbdMemberState          MemberState,
    mbdMemberAddress        TAddress,
    mbdMemberLastHeard      TimeTicks
}
```

Figure 6: Group membership MIB table.

The table is indexed by the group identifier `mbdMemberGroup` and the member identifier `mbdMemberId`. The current state of the group member (`mbdMemberState`) identifies if the member is a normal member, the master of the group or currently unavailable. The transport address of a group member is given by the `mbdMemberAddress` field and the time stamp at which the last `mbdAliveTrap` was received is given relative to the local system uptime by `mbdMemberLastHeard`.

## 5 Conclusions and Related Work

This paper describes some ongoing research to provide group communication facilities on top of the Internet network management protocol SNMP. We have shown how group membership information and a simple master election algorithm can be build by using SNMP trap messages encapsulated in IP/UDP multicasts.

Group communication primitives allow to build fully distributed management applications that are executed by autonomous cooperating agents. Fault tolerance can be achieved by replicating management functions in a set of cooperating agents. The delegation mechanism adds mobility to the management threads which allows to balance the management load over a set of agents.

There is a lot of work available in the literature dealing with different multicast mechanisms. However, there is only little work done in the context of distributed network management systems. A group communication protocol for a distributed network management system that provides ordered delivery of messages has been proposed by K.H. Lee [8]. More work is needed to understand which properties of group communication protocols are actually needed by management applications and how the protocols behave in situations where the network is unstable. One of the main purposes of a network management system is to provide help in situations where the network does not operate as designed. It might therefore be useful to lower the consistency requirements usually found in distributed systems.

There are a number of papers dealing with election protocols. Some newer work tries to use formal techniques to analyze the correctness of election protocols [3]. A similar formal analysis of the protocol described in this paper could be useful to better understand its limitations.

Another important topic is the security of network management applications. The proposal to add a user-based security model to SNMPv2 [14] fits nicely into the multicast SNMP scheme presented in this paper because it allows to share security parameters between SNMP entities. It is therefore possible to authenticate multicast trap messages between cooperating agents. This is an improvement over previous approaches to SNMP security which did not allow to share so-called SNMP parties.

## References

[1] V. Baggiolini, J. Harms, M. Ramluckun, E. Solana, W. Spahni, and C. Tschudin. Management of Distributed Applications: Management of Electronic Mail Systems. In *Proc. Workshop of the Priority Program in Informatics Research*, Bern, November 1994.

[2] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. Internet Draft 05, MIT/LCS, UC Irvine, MIT/LCS, February 1996.

[3] J.J. Brunekreef, J.P. Katoen, R.L.C. Koymans, and S. Mauw. Design and analysis of dynamic leader election protocols in broadcast networks. *Distributed Computing*, 9(4):157–171, 1996.

[4] S. Deering. Host Extensions for IP Multicasting. RFC 1112, Stanford University, August 1989.

[5] V. Hadzilacos and S. Toueg. Fault-Tolerant Broadcasts and Related Procedures. In S. Mullender, editor, *Distributed Systems*, chapter 5, pages 97–145. Addison-Wesley, 1993.

[6] ISO. Information technology — Open Systems Interconnection — Systems-Management Functions — Event Report Management Function. International Standard ISO 10164-5, ISO, 1993.

[7] ISO. Information technology — Open Systems Interconnection — Systems-Management Functions — Log Control Function. International Standard ISO 10164-6, ISO, 1993.

[8] K.-H. Lee. A Group Communication Protocol for Distributed Network Management Systems. In *Proc. ICCC 95*, pages 363–368, Seoul, 1995.

[9] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralizing Control and Intelligence in Network Management. In *Proc. International Symposium on Integrated Network Management*, May 1995.

[10] J.K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, April 1994.

[11] J. Schönwälder and H. Langendörfer. Tcl Extensions for Network Management Applications. In *Proc. 3rd Tcl/Tk Workshop*, pages 279–288, Toronto (Canada), July 1995.

[12] W. Stallings. *SNMP, SNMPv2 and CMIP: The Practical Guide to Network Management Standards.* Addison-Wesley, 1993.

[13] S. Waldbusser. Remote Network Monitoring Management Information Base. RFC 1757, Carnegie Mellon University, February 1995.

[14] G. Waters. User-based Security Model for SNMPv2. RFC 1910, Bell-Northern Research Ltd., February 1996.

[15] Y. Yemini, G. Goldszmidt, and S. Yemini. Network Management by Delegation. In *Proc. International Symposium on Integrated Network Management*, pages 95–107, April 1991.