Context Driven Access Control to SNMP MIB objects in multi-homed environments

Radu State, Olivier Festor, Isabelle Chrisment

Context Driven Access Control to SNMP MIB objects in multi-homed environments

R. State, O. Festor, I. Chrisment

INRIA-LORIA
615, Rue du Jardin Botanique
54600 Villers-les-Nancy
France
{state, festor, ichris}@loria.fr

Abstract. The advent of multi-technology networks offering the service continuum over multiple network infrastructures implies new challenges to integrated management. One of these challenges is the auto-configuration of the management plane needed to allow dynamic relationships among several managers and one management agent. This paper proposes the use of provisional policies in order to dynamically auto-configure the access control part of a management agent. This allows simple management based on agent location and time as well as the cooperative behavior of several managers.

1 Introduction

Recent advances in providing multiple services over heterogeneous networks demand a new approach for constructing the management plane. The management paradigms used in today networks are based on the manager agent model (designed in the mid 80') which defines a protocol for exchanging management information, specified according to an information model and stored in a conceptual repository, called Management Information Base (MIB).

In this approach and for the last decade, the management plane was configured in a static manner. For instance a basic configuration of the management protocol established at agent boot time is always assumed to exist. Such an assumption does not hold any more when we are trying to manage nomadic equipment, multi-homed sites and dynamic service infrastructures. In this paper we address the issue of providing a management approach which extends the SNMPv3 framework allowing the dynamic configuration of the access control to MIB objects using a context specific access control. The automatic configuration of the management plane is an essential step towards fully integrated management. Including self-management features within the management plane is important for plug and play type of management, where minimal human interactions are requested. On the other hand, existing management frameworks should be easily extended/integrated without demanding a total conceptual and implementation change. Our paper is structured as follows: section 2 describes the business case and motivates our work. An introduction to provisional authorizations and their usage for the configuration of the management stack is given in

section 3. The management framework is described in section 4, while pointers to related work are given in section 5. Section 6 concludes the paper by highlighting future work.

2 The Business Case for Self-Configuration of MIB Access

This section presents two simplified business cases (see figure 1) motivating the potential of the self-configuration of the management stack. In this first case, Bob owning a WIFI enabled laptop uses his laptop both at home and work. While Bob is at the office, his laptop is under the management responsibility of the enterprise management platform (EMP). For instance, Bob might not even be allowed to auto-administer his laptop. However, as soon as Bob leaves the office and gets home, the same laptop which is connected to the home network should be considered under the management responsibility of the home management platform (HMP). At this moment, the enterprise management platform should not be allowed to perform any management operations on Bob's laptop. An extension to the previous case consists in adding some additional constraints. Bob's laptop is owned by the enterprise. If Bob is working at the office then his laptop is under the total control of the enterprise management applications. However, when Bob connects his laptop to his home network, the home network management application might perform management operations if and only if the enterprise network management platform approves it.

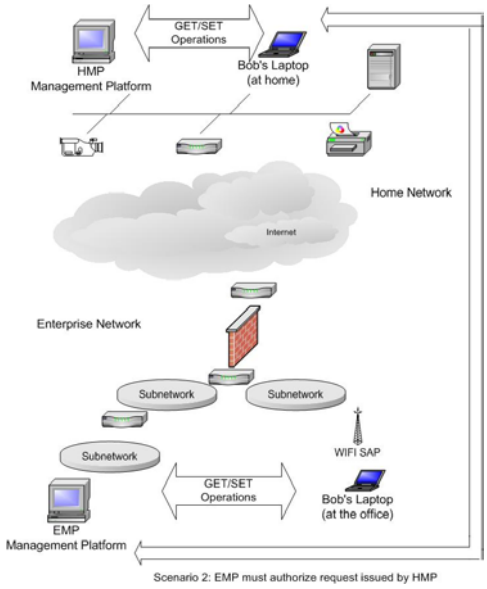

Fig. 1. Simplified Business Case

Looking at the two scenarios in order to abstract the fundamental requirements we can claim the following requirements:

1. Management needs to be context driven. A context can be defined as the overall ensemble of parameters characterizing the instant connectivity. For instance the context can be the collection of network interfaces, their IP addresses, netmask, and the DNS used.
2. Dynamic interactions among several managers need to be supported. It should be possible to a manager to approve or deny operations performed between an agent and another manager, without requiring direct manager to manager communication.

Does the current management SNMP framework meet these requirements? The answer is definitely not. For instance, the most advanced SNMP version (SNMP v3) [6] [17] allows to offer several views and access rights to the MIB, but these ones are defined statically independent of the context. This is done using an enhanced access control module, also known as the View Based Access Control Model (VACM) [6]. In the case of the simple scenario presented above, using the current VACM model defined by SNMPv3, would imply that the EMP could perform some management operations even when Bob is connected from home. This is based on the assumption that Bob will not manually configure the VACM every time he connects to a network. The extended scenario (where the EMP must authorize management requests performed by HMP) is definitely more complicated and can not be implemented with current management paradigms. In the following section we will introduce provisional policies and show how an extension of the current VACM can be based on provisional authorizations in order to meet the requirements. Two more additional requirements must be addressed:

1. Backward compatibility with SNMP [5].
2. Whatever works right now, should also work in the future. That is, existing management applications should be valid and fully working.

3 Provisional Authorizations for Dynamic Access Control to MIB Objects

Provisional authorizations have been introduced in [7] and [8] as a solution towards additional semantics for controlling access requests. Traditional access control systems considered that access requests can be modeled as a demand to authorize a particular action (read/write/delete) made by a subject within a given context on a particular object. A provisional authorization is a generalization of this scheme, modeling a conditional authorization, ie, an access request is granted if and only if additional conditions hold. This approach can be applied at the agent side. MIB objects are entities to which read/write requests are made. Provisional authorizations are stored on the agent and regulate the access to the MIB objects. The provisional framework for management can be described as follows:

1. The set S represents all possible subjects requesting access to objects. In our case, these are all managers: For instance in terms of the previously described simple scenario $S = \{EMP, HMP\}$
2. The set of all possible objects is O. This corresponds to all OIDs in the MIB.

3. A is the set of access modes permitted on objects. $A = \{get, set\}$
4. There is the notion of context, c, representing time and location. The set of contexts is C. A particular context is given by the collection of ipaddress/netmask/DNS used on each network interface. In this paper, we will use the term context having this definition in mind. The VACM aware reader should not confuse this term with the one used to specify a MIB view in SNMPv3.
5. A permission is either to grant or to deny access. The set of permissions is Perm = {grant, denied}
6. A set FM of formulas. An individual formula f is a logical conjunction of equalities and/or inequalities. For instance t>18 means current time after 18h. An equality or inequality is constituted of constant terms (string or numbers) as well system accessible variables. System accessible variables are all variables in the MIB. Other system specific variables might also be presented. If a formula holds, then the authorization policy is active (see in the following for the definition of an authorization policy). Examples of formulas could be :
    - "`1.3.6.1.2.1.ip.ipInAddrErrors>1000`", allowing a manager to get/set values in a subtree, whenever the number of datagrams errors due to address errors is higher than a threshold. While visiting a foreign network, Bob could allow management access to his machine, if and only if it's starting to have functioning errors.
    - "`1.3.6.1.2.1.interfaces.ifTable.ifEntry.ifIndex.1.ifOperStatus==down`". This formula expresses the fact that the first network interface of an agent's node does not work. For a multihomed node, one can use this formula to allow a manager connecting via the second interface to fully manage the agent if the first network interface is not working. As a typical business case, let us consider an access router with two interfaces, used by a home network. Two managers do exist. The first one belongs to the Internet Service Provider, while the second one is used on the home network. Such a formula could be used to allow the home manager full control over the access router whenever no connection to the ISP's manager is possible.
7. A set of provisional actions PRV. A provisional action is used to add new semantics on an authorization policy. We provide one single extension *verify*, meaning that another subject agrees with this operation. For instance *verify(HMP)* means that HMP has to agree with a particular request in order to authorize this access. We can model the agreement of several managers using a list of provisional actions. Several comma separated verify clauses are equivalent to a conjunction of authorizations. All subjects must authorize the request. Semi-colon separated list of provisional actions represent the fact that at least one entity must authorize the request. These types of actions are used in situations when a subset of managers must all agree on a set of ac-

tions. The second case is used when any manager out of a subset has to authorize a request.

The configuration of the access control scheme is based on providing a set of authorization policies : $Auth\,\mathrm{Pr} \subset O \times S \times A \times C \times Perm \times FM \times PRV^*$

An authorization policy, for instance is:

$$(1.3.6.1.*, EMP, get,$$
$$context(address = 194.224.3.23, netmask = 255.255.255.0, dns = 195.224.3.1),$$
$$grant, time > 18, verify(HMP)) \in AuthPR$$

modeling the fact that : Manager HMP is allowed to read all elements under the subtree 1.3.6.1, if the agent is connected with IP address 194.2234.3.23, netmask 255.255.255.0 and DNS server 194.224.3.1, and current time is past 18h, ***provided that EMP agrees***.

An authorization request is a 4 tuple : $ar = (o, s, a, c) \in O \times S \times A \times C$ meaning that action a in context c is to be performed by s on object o. For the previous example, such a request might look like:

$$ar = (1.3.6.1.*, EMP, get,$$
$$context(address = 194.224.3.23, netmask = 255.255.255.0, dns = 195.224.3.1))$$

This request models the read operation on OID starting with 1.3.6.1, performed by manager EMP, where the current connectivity configuration of the laptop is given by the address, netmask and DNS. For every authorization request, an authorization decision is computed based on the set of authorization policies. Basically, a decision is to either grant or deny an action. Allowing an action can be either unconditionally or conditionally. A conditional allow is associated to a list of provisional actions. These actions must be performed in order to fully allow the operation. An example based on the previous scenario is the following decision:

$$decision = (1.3.6.1.*, EMP, get,$$
$$context(address = 194.224.3.23, netmask = 255.255.255.0, dns = 195.224.3.1),$$
$$grant, verify(HMP))$$

This decision allows the access for EMP to read all objects in the subtree 1.3.6.1, if the agent is on a network (ip address of the agent=194.224.3.23 with netmask 255.255.255.0) if and only if HMP agrees.

The provisional action can include several actions. "*verify(HMP), verify(Bob)*" is equivalent to both Bob and HMP must agree with this decision, while "*verify(HMP);verify(Bob)*" means that at least one of them must agree. This framework is a generalization of the SNMPv3 VACM architecture. However, this generalized approach allows us to model context driven and conditional management, as well as more advanced interactions between managers as described in the two business cases. Another applicability of this framework lies in providing a potential solution towards autonomous management.

For the moment, we postpone the presentation of the management architecture needed to support this authorization framework, and focus on the semantic expressiveness of this model.

Let us first see if the two described scenarios can be implemented by the model. Obviously, for the fist scenario, we can use the following four authorization policies:

1. $(*, EMP, get, context(address = 152.224.3.23, netmask = 255.255.255.255, dns = 152.224.3.1), grant, null)$
2. $(*, EMP, set, context(address = 152.224.3.23, netmask = 255.255.255.255, dns = 152.224.3.1), grant, null)$
3. $(*, HMP, get, context(address = 194.224.3.23, netmask = 255.255.255.255, dns = 195.224.3.1), grant, null)$
4. $(*, HMP, set, context(address = 194.224.3.23, netmask = 255.255.255.255, dns = 195.224.3.1), grant, null)$

Policies 1 and 2 state that the enterprise management platform in allowed to get/set any object in the MIB as long as Bob's laptop is on the enterprise network. Policies 3 and 4 model the equivalent for the home network. Obviously, dealing with dynamic allocated addresses is done by using wildcards in the specifications. For this scenario, no formulas are requested. It's also easy to see that Bob's terminal cannot be managed by his office manager, when Bob is at home.

The second scenario, in which Bob's enterprise manager must agree with the home management application, is already presented when we have introduced the provisional authorizations.

Without detailing an implementation specific verification mechanism, let us consider the following authorization policy:

$(1.3.6.1.4.1.2, *, read, context(address = *, netmask = *, dns = *), grant, null, verify(Bob))$

This policy models the fact that Bob must be asked by the management agent whenever a get request is made on the subtree 1.3.6.1.4.1.2. Such flexibility is interesting in the management of mobile devices, in which users could be prompted to explicitly authorize operations performed on their terminals (like for instance reading their configuration/agenda by a foreign management application). It could be considered as an user-interactive SNMP agent.

Finally, it's natural to ask if such an authorization policy based approach can be used for total self management. Without pretending to have a clear definition of total self management, we argue that at least it might give you illusion of self management. Imagine the very simple case of one authorization policy:

$(*, *, *, context(address = *, netmask = *, dns = *), grant, null, verify(manager))$

The intelligence or auto-configuration feature is actually hidden by the existence of a manager who authorizes every request. An agent using such a policy can be easily plugged into any network and provide the illusion of being auto-configurable. The auto-configuration in this case is actually outsourced and delegated. This is obviously an extreme case, having a lot of overhead in terms of management communication, but it shows that provided the existence of an authorization manager and a convenient set of authorization policies, a relative degree of autonomy can be achieved. Such a mechanism is very useful for the management of nomadic equipment. The management platform to which the equipment belongs can specify limited management operations allowed whenever the equipment is not on the home network and might agree

to extended management provided it is consulted. As far as we know, no previous management framework is capable of similar features.

4 Implementation Framework

This section describes the management framework based on the authorization policies. One of the requirements was not to depend on a new management protocol. Taking SNMPv3 as a major building block, our approach consists in providing a new Access Control subsystem and an optional extension within an SNMP agent. Figure 2 (adapted from [6], [19]) shows where the new access control subsystem and the optional extension fit into the block functional architecture of an SNMP agent. One issue that was not addressed in this paper yet relates to the dynamic interactions among an agent and one or several managers. The issue is how do managers and agents discover reciprocally, and how to provide a security model for such a framework.

The existing security model defined in SNMPv3 is implemented in the USM (User Security Model) [18] allowing the authentication of the manager as well as the optional privacy of the communication. For the authentication and privacy, two secret keys are shared between the agent and the manager. The first one is used to authenticate the manager, while the second one is used to encrypt/decrypt the SNMP operations. While the existence of these two keys can be assumed by an off-line exchange among the managers, we consider that full autonomy for the management plane is based on a dynamical manager to agent secret key exchange. Figure 2 shows two additional components and their interactions. The first one is a Manager Discoverer. Its main functionality is to discover network managers. Several choices are possible:

1. The manager learns through topology monitoring/DHCP the existence of a new node, and checking UDP port 161 on the node, discovers the agent.
2. The address of the manager is included in an extension of the DHCP configuration 10. An IETF proposition [15] suggests the use of DHCP to detect a list of IP addresses to which notifications have to be sent.
3. The agent uses the DNS to look for a manager located in the same domain
4. A management service containing the description of the manager location is advertised over SAP (Session Announcement Protocol) [12]
5. The agent discovers via the Service Location Protocol (SLP) [11] the manager.

The information about the manager contains also a public key *Public(M)* on which the latter is listening. The manager uses the RSA (Rivest, Shamir, Adleman algorithm) [13] with the public key *Public(M)* and an associated private key *Private(M)*.

As soon as a new manager is discovered, two secret keys *authk, privk* are generated by the agent and stored in the MIB of the agent. Although, the existing SNMPv3 standard discourages the storing of secret keys in the MIB, we consider that proper access control performed by the agent can assure the required security. The Manager obtains the two keys by issuing a Get Request. This Request must be encapsulated in a SNMPv3 packet having *msgSecurityModel* equal to 4. This field describes the security model to be used by the agent. Normally, 1 is used for SNMPv1, 2, SNMPv3 uses the value of 3 and SNMPv2c uses 2 [6].

The value 4 corresponds to our PK security model. This PK security model checks the authenticity of the issued requester using the manager's public key *Public(M)*. In fact, our extension provides a possible public key exchange facility to a SNMPv3 agent. The assumption is here, that the manager discovery process detects a genuine trusted service.

If authentication is valid (this process is based on the fact that only the legitimate manager has the key used for encryption: *Private(M)*), the two keys (*authk, privk*) are sent to the manager. The response is encrypted using the public key of the manager: *Public(M)*.

The next following interactions between the manager and the agent are performed using the traditional security model (*msgSecurityMode*l=3), which is based on symmetric keys (*authk, privk*) and therefore more efficient.

The second extension consists in a new Access control subsystem, which is used by a SNMP engine to check that particular request is authorized.

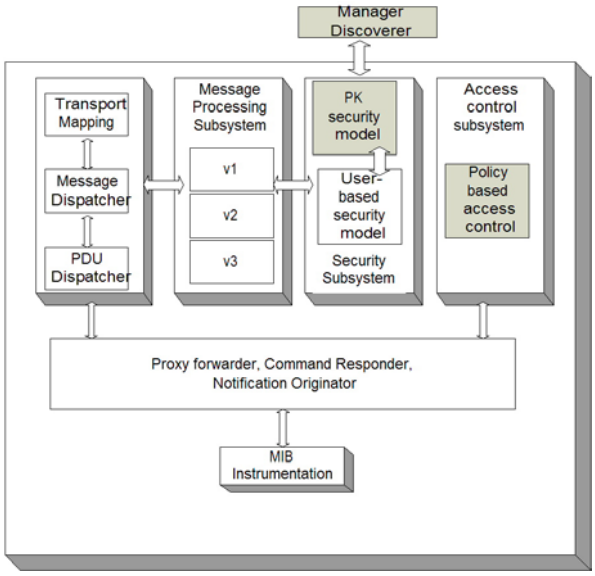

Fig. 2. SNMP agent architecture

A functional block decomposition of the Authorization based Access control is shown in figure 3.

Authorization Policies are stored in a Policy store. This store is implementation specific. A policy management console pushes authorization on this store. This configuration can be either based on SNMP, similar to the VACM configuration in SNMP3. In this case an extended MIB containing tables corresponding to the formulas and the provisional actions are needed. Note, that for the address and netmask variables used in the context, the existing MIB2 entries can be used.

Another configuration of these policies can be done using COPS-PR [4]. One of the advantages of COPS-PR lies in its working over TCP. If the self-management ca-

pable agent is connected to a distinct network then the one containing the policy console, most firewalls will drop SNMP traffic, leaving however TCP connections originating from the agent.
The Policy Validator is responsible to check the coherence of deployed policies. Its main objective is to resolve conflicting authorizations.
A Request Processor is the entity where an access request is made. The request processor uses the policy store and computes the authorization decision. If access is either granted without any provisional actions, or denied, this decision is returned to the SNMP engine.
To summarize, authorization policies are configured in the Policy Access Control module of the SNMP engine. When such a policy is triggered, an authorization might be requested from an authorization manager. This request can be implemented via COPS, in which a explicit accept decision is requested from a PDP. Therefore, the concept of policy is twofold. We have authorization policies regulating the access rights to the SNMP agent and we have authorization requests and replies triggered by an authorization policy.

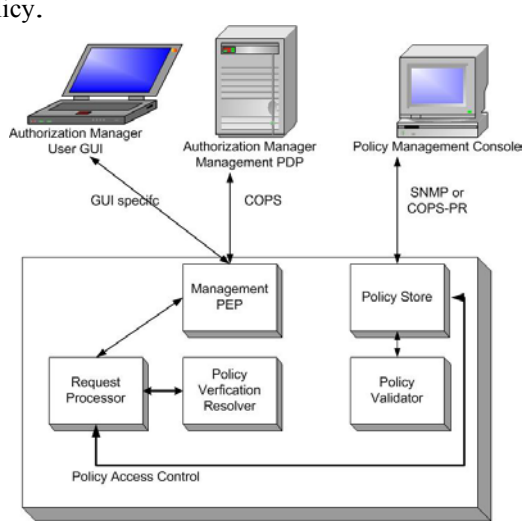

Fig. 3. Functional Architecture of the Access Control Subsystem

One interesting example consists in a management approach for a multi-homed host. This case, (shown in figure 4) assumes that both managers trust a third manager T. The latter is responsible to assure that requests issued by one manager do not alter/expose sensible management information. For requests issued to particular OIDs, T must explicitly approve them. The assumption that a common trusted manager exists might be relatively optimistic. If this assumption does not hold, both A and B should be asked to allow these requests. This can be expressed in a provisional action might be "*verify(A), verify(B*)". This means, that both manager have to approve this request. Obviously, this implies an overhead, but provides a solution for either not disclosing A's confidential monitoring information to B, or allowing B to configure A's related entities.

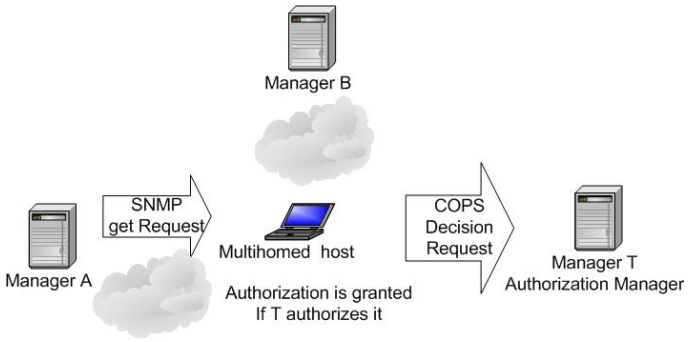

Fig. 4. Management of a Multi-homed host

To see the usage of COPS for authorization checking, assume that A issues a Set on a OID and that the Control Subsystem decides to grant it, provided T confirms.

The Control subsystem issues a COPS Request Message [3] to T. The COPS-context in this message is set to "Set", (it would have been "Get" for a Get Request). Additional COPS-type objects in this message, called *ClientSI* (client specific information object) represent the OIDs and the *snmpEngineId* of the requester.

If T decides that the request is granted, it replies with a COPS-Decision having a command INSTALL. From now one, whenever A issues the same request, (within the same context and formulas still holding), this is granted by the agent. If after some time, T issues a Remove Decision, the grant decision of the agent, must be re-confirmed as previously explained.

5 Related Work

We started our work based on the general management agent security configuration proposed in the SNMPv3 [6], [19] specifications. The View Access Control Model [17] allows the definition of view and allowed operations for a set of managers. Its user security model USM [18] provides the supporting mechanisms for manager-agent privacy and authentication. These mechanisms must be configured statically and are difficult if management of nomadic or multi-homed equipments has to be performed. These approaches are extended by our work with context driven and conditional management as well as a security model configurable dynamically. One of the first initiative towards simplified and automatic host configuration was started by the zero-configuration group at the IETF [9], where individual (private network) addresses are automatically assigned without requesting static configuration or DHCP [10] support. Policy based network management has been applied for QoS management in both Diffserv as well as Intserv [1] types of networks, without however addressing the self-management issue. A policy based approach for the auto-configuration of networks is proposed in the Nestor [14] platform used to manage an active network platform. Change operations are managed via policy rules and inte-

grated within a larger network self configurability architecture. The research community working in access control mechanisms proposed a large variety of security architectures and policy specification methods. However, no direct applications towards extending the management functionality based on these concepts have been proposed. Our approach is different with respect to the previously mentioned work in several aspects. We start with the main objective to use SNMPv3 the standard management protocol in an autonomic way. We propose a novel architecture for an SNMP agent capable to integrate within existing deployed networks and without requiring a complex additional infrastructure at the network site. We have not yet decided to use a specific policy specification framework. Our current prototype uses a proprietary solution. We are looking into applying the PONDER specification framework [16] for this purpose. PONDER provides an extremely powerful language which could be capable to express management authorizations as well as context related information.

6 Conclusions

We addressed in this paper the issue of flexible network management, respectively self-management. We started with the observation that current standardized network management frameworks do not offer enough support for enhanced agent autonomy. The simple observation is that an already configured SNMPv3 agent, taken out from his home network and put under the management control of one or several foreign management applications without any additional human interaction is not functional. This is due to several factors. Firstly, a fixed security Model (USM) and a fixed Access Module (VACM) are incapable to configure on their own. A second factor, which is more conceptual, is related to the existing management paradigm, in which a manager interacts directly with an agent. We extend this paradigm by allowing other parties (managers) to express their agreement within such an interaction. We provide a framework allowing context/location driven management and conditional management. We argue that current SNMPv3 specifications provide an excellent authorization and access control mechanism for a fixed environment. This is the case with most existing target environments. However, nomadic environments where more and more users are mobile require new management frameworks. We addressed this issue by assuming the standard management protocol SNMPv3 and we proposed an extension of the SNMPv3 management framework. A new security model and an access control are proposed. The proposed security model allows the exchange of the authentication and privacy keys. These keys are used by the SNMPv3 user security model. Our management framework is based on authorization policies and provisional actions in which context driven management can be performed. The context is defined by the network connectivity properties used on the managed equipment site. Our approach generalizes the View based Access Control subsystem proposed in the SNMPv3 architecture, without requiring changes at the SNMP protocol level. Thus, our extension is transparent for already existing management applications. Existing management agents should be easily modified in order to enable dynamic access control to MIB objects. We consider that self-management is strongly related to the auto-configuration of the management plane. Management of mobile devices as well as

environments where multiple management applications interact dynamically are the primary immediate business targets. Our approach is based on managing the management stack within a policy based solution. This integration shows also the complementary nature of these two types of management and motivates the necessity of having both of them within a single management stack. We are currently implementing the proposed architecture within a Net-SNMP framework. It will be validated within our IPv6 testbed.

References

1. D. Verma. Policy-Based Networking. New Riders Publishing.2000.
2. RFC 3159. Structure of Provisioning Information (SPPI). IETF. 2001
3. RFC 2748. The COPS (Common Open Policy Service). IETF. 2000
4. RFC 3084. COPS Usage for Policy Provisioning (COPS-PR). IETF.2001.
5. W. Stallings. SNMP, SNMPv2, SNMPv3 and RMON1 and 2 Addison-Wesley Pub Co; 3rd edition 1998)
6. W. Stallings. Network Security Essentials, Prentice Hall, 2nd edition 2002.
7. M. Kuda , S.Hata. XML Document security based on provisional authorization Proc. 7th ACM Conference on Computer and Communication Security (CCS 2000), Nov. 2000.
8. S. Jajodia, M. Kuda, V.S. Subrahmanian. Provisional authorizations. Workshop on Security and Privacy in E-Commerce (WSPEC), Nov. 2000, Recent Advances in Secure and Private E-Commerce, published by Kluwer Academic Publishers in 2001.
9. E. Guttman. Autoconfiguration for IP Networking: Enabling Local Communication, IEEE Internet computing. 2001.
10. R. Droms. "The DHCP Handbook". Sams. 2nd edition. 2002.
11. RFC 2608. Service Location Protocol , Version 2. IETF 1999.
12. RFC 2974. Session Announcement Protocol. IETF 2000.
13. B. Schneier. Applied Cryptography: Protocols, Algorithms and Source Code in C, Second Edition. John Wiley and Sons. 1995
14. A. V. Konstantinou, Y. Yemini, and D. Florissi. Towards Self-Configuring Networks. DARPA Active Networks Conference and Exposition (DANCE), May 2002, San Franscisco, CA.
15. M. Bakke. DHCP Option for SNMP Notifications. draft-bakke-dhc-snmp-trap-01.txt. Internet draft IETF. Work in progress 2003.
16. N. Damianou. A Policy Framework for Management of Distributed Systems. Ph.D thesis. Faculty of Engineering of the University of London and Diploma of the Imperial College of London. London. December 2002.
17. RFC 3415. View Based Access Control Module (VACM) for the Simple Network Management Protocol. IETF 2002
18. RFC 2274. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). IETF 2002
19. RFC 3411. An architecture for describing simple network management protocol (SNMP) Management Frameworks. IETF 2002