Managing Groups in Dynamic Networks

B. Gruschke, S. Heilbronner, N. Wienold
Munich Network Management Team
Department of Computer Science, University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
Phone: +49-89-2178-{2170,2168,2171}, Fax: -2262
E-Mail: {gruschke,heilbronner,wienold}@informatik.uni-muenchen.de

Abstract

Traditionally networks are structured for operation in quite a static manner. Even if routing protocols support flexible and self-organizing networks, other configurative tasks like forming LAN segments on (OSI) layer two and subnets on layer three as well as most application-specific configuration are not intended to be frequently modified. Reorganizing them therefore often requires larger management efforts.

In recent years several technologies such as Virtual LANs, configuration bootstrapping and directory protocols have evolved to allow the implementation of dynamic configuration in the networking and computing infrastructure that in turn can support the dynamics needed for nomadic computing and frequently changing, flexible organisational structures. Each of these technologies defines groups at a different layer. This leads to new requirements for integrated management solutions covering all networking, system and application entities and motivates the necessity of a new kind of more generic tools.

We will derive user requirements starting from a concrete scenario which demonstrate the various needs for defining groups of resources in a dynamic and flexible manner. After investigating some emerging technologies, with respect to their grouping aspects, e.g. related to the frequency of changes, multitude and layering of groups, we will focus on the idea of a generic *group object model* as an important management abstraction. The motivation of this abstraction is supported by its application to event correlation and fault diagnosis and for a policy–based configuration management, especially in support of nomadic computing.

Keywords

Groups, Domains, Event Correlation, Fault Diagnosis, Nomadic Computing, Policies, Virtual LANs, Corba

1. Introduction

Currently, remarkable complexity is added to networks and distributed computing infrastructures: One trend is to add new layers to networks, i.e. to introduce *virtual networks*. Another is to automate configurative tasks like assigning IP addresses and

host names as well as basic system and application configuration. Both increase *dynamics* that the management has to deal with.

From a technical point of view these trends are driven by many new and diverse technologies to be used alternatively or in combination. From a user's point of view there are various reasons to apply such technologies. For example: Traffic separation enabled by virtual networks is performed to allow various levels of quality of service, to assign traffic costs to customers (i.e. accounting), for security reasons and to bridge between technologies.

From a management point of view — always looking for generics — these different technologies and motivations to apply them share common aspects: They *group* certain kinds of managed objects. Since this is a widely used term our use requires some clarification: While in the OSI management [10] and [22] groups of arbitrary managed objects (termed *domains*) are formed to define the scope of a management *policy*, the grouping here is imposed by the **managed system**, i.e. the infrastructure, itself. A group membership then typically allows for exploitation of services. Therefore the group concept reflects technical restrictions, e.g. concerning the kind of members of a group and the relation between groups. In other words, we define a *group* to be one of the results of the refinement process of a policy on a domain. This refinement is the mapping of higher-level (strategic or goal-oriented) policies down to the *operational* level, where they can finally be implemented. This terminology is a variation of [22] (and later) and is similar to the one used in [12] and [7]*. [24] describes additional criteria for policy classification and refinement requirements.

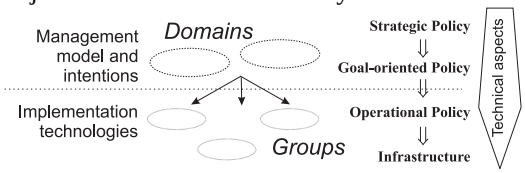


Figure 1: Policy implementation processs

In section 2 we introduce a scenario that motivates why it is necessary to be able to create groups in a dynamic manner on lower (OSI) layers as well as on the application layer. Additionally the scenario describes some of the more prominent technologies being used to form groups in networks, systems and applications. Their requirements lead to the definition of our *group object model* in section 3. In section 4 we apply our model for integrated fault management with event correlation and fault diagnosis using dependency graphs and for management procedures to support nomadic computing. Section 5 gives an overview of the implementation and provides a summary and an outlook.

2. Scenario

In this section we illustrate how a certain class of user requirements on the management of the IT infrastructure maps to the issue of defining groups of managed objects. The user's point of view is derived from a scenario called the *Office Park*. This scenario reflects the increasing trend towards deeper IT provider hierarchies, e.g.

*We deem this differentiation to be useful for proper description of policy implementation.

originated by the outsourcing of IT services, and the implied introduction of new and more customer–provider relationships. It also motivates the demand for flexible management systems which leads to the necessity of a simple and standardized forming of groups. We do not provide concepts specifically for customer network management but assume multiple administrative authorities.

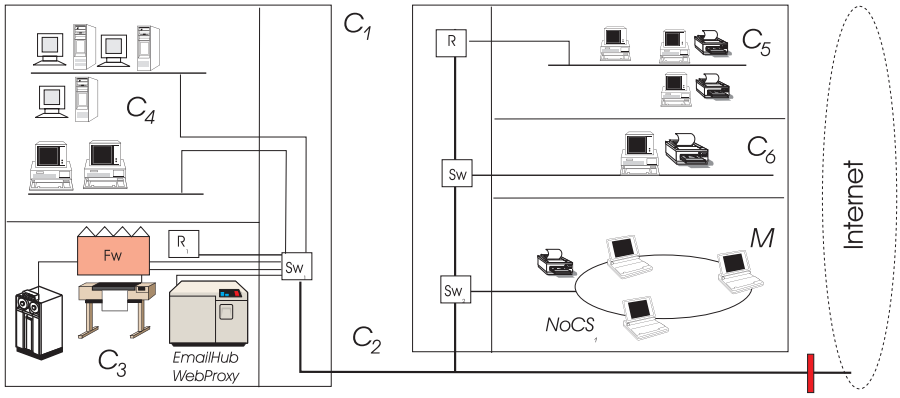


Figure 2 The *Office Park* scenario

Within the Office Park (figure 2) companies C_{1-6} are characterized by their computing resp. outsourcing needs as follows (simplified for the sake of brevity): C_1 is the company responsible for the office management within the park. It has chosen C_2 for maintaining the cabling between the ports (physical layer) as well as running the switches (data link layer) that provide the backbone between the ports for certain customers. C_3 is a subsidiary office (branch) of a company specialized in providing essential computing services to customers on site, such as Email and Web server operation, backup and storage management as well as secure Internet access, e.g. firewall operation (Fw)*. C_4 is a larger software development company that has chosen C_3 for its outsourcing needs related to outside communications, but nevertheless keeps the operation of their local computing infrastructure inhouse. C_5 is a small publishing company which prefers to have some printing equipment inhouse but nevertheless shares the high–performance color printer and other equipment offered by C_3. Finally, C_6 is a single-person consulting firm.

Nowadays, the complexity of the described scenario is often complicated further by the appearance of *nomadic systems* and *users* who also need to access the provided resources. In our case we expect nomadic usage to appear in two forms: (a) Users from different companies from within the Office Park gather in meeting rooms, such as M. (b) Users from outside (and therefore unknown to the Office Park systems)

*It is therefore also the only customer towards the company providing access to the Internet backbone.

appear. In both cases interoperability is to be established among these systems as well as with surrounding equipment and resources.

The service provisioning scheme of the Office Park is depicted in Figure 3. As it is increasingly common to formally specify Service Level Agreements (SLAs) the provider–user interaction at the service access points is supplemented by a management–related interaction (expressed by the gray crossbars). For the sake of brevity, we will stem our presentation on the requirements from the interaction related on fault and configuration management. Of course, the other functional areas such as accounting, security and performance are more or less affected, too.

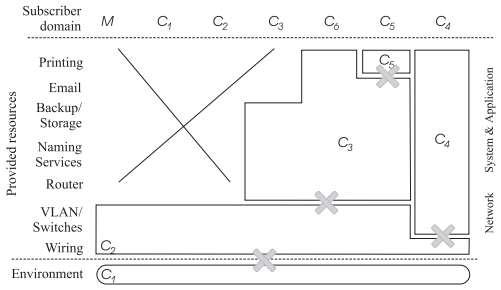


Figure 3: Service provisioning scheme within the scenario

When regarding the services offered within this scenario it is widely seen that the interaction may be governed by a set of rules, i.e. *policies*, for each combination of (management) subjects such as service users, accessed servers and resources. The large quantity of possible combinations and the expected change rate of the rules necessitate a flexible, dynamically updateable policy enforcing system [7]. By further examination of the requirements of nomadic computing different kinds of policies have been identified: (a) Policies governing the default configuration of a system. (b) Policies governing access in terms of specific rights to resources. (c) Policies governing the *groups* to which these access rights are granted.

When carried out, the refinement of these policies to management procedures in our scenario frequently leads to management operations in various areas; yet one is predominant due to the frequent changes in user–provider relationships: establishing, querying, updating and deleting **groups** of management objects with respect to their relationships to other management objects. Examples for group objects are:

- groups of **hosts** within a VLAN, e.g. **office rooms** of C_4, identified by their corresponding switch ports or their Ethernet addresses,
- groups of IP clients with common routing characteristics, e.g. **subnets** within C_5,
- groups of hosts with common settings for basic system services, e.g. in M,
- groups of **mail clients** granted access to a mail relay or a message store,
- groups of printers that may only be accessed by a certain group of users, e.g. C_5 or **authenticated customers** in M,
- groups of clients allowed to reserve and consume bandwidth and communication resources across the WAN link with certain QoS characteristics, e.g. C_6 for the transfer of multimedia data and
- groups of **users** under common security policies, e.g. with respect to their information rights and the required authentication strength, e.g. executive staff of C_5.

An additional challenge in our type of scenario is the fact that rights like to add

a member to a group may lay at different entities for each group and each resource affected. This is an important issue since it requires that access control lists on groups have to be specific according to the type of operations.

Technologies Implementing Groups

All user requirements with respect to form groups described in the previous section can only be met when taking technologies at several layers (of the OSI model) into account. Some of these technologies are presented in more detail later in order to make it easier for the reader to understand the motivation and the correctness of the object model developed.

Virtual LANs (VLANs) VLANs are groups forming distinct logical Local Area Network (LAN) segments (generally at the physical and data link layer) on top of a single physical network. Historically, the concept is not primarily motivated by privacy concerns but by performance, fault and scalability aspects in growing LANs. Communication between VLANs requires a router connected to both VLANs. The VLAN membership is defined using certain grouping criteria, e.g. the switch port a DTE is attachted to and/or a certain MAC address*. The grouping criteria can be used alone or in combination with others to implicitly or explicitly define the VLAN to which the host (interface) belongs to. When frames are transmitted through the layer two network, the switches spanning the physical LAN need to know to which of their ports other hosts of the VLAN are connected to. For this purpose the switches exchange VLAN management information. Besides some proprietary protocols from CISCO (ISL and a modified version on IEEE 802.10) or 3COM (VTP) a VLAN standard is developed by the IEEE 802 working group covering these kinds of protocols and some management aspects [9].

Dynamic DNS and DHCP The Domain Name System (DNS) [14] is the Internet standard technology for providing basic directory services, e.g. mapping host names to IP addresses, as well as organizing the directory entries in hierarchical name spaces. A rudimentary service location is supported by mapping names from a set of predefined (wellknown) aliases to actual host entries, e.g. *www*.mnmteam.de to hpheger0.nm.informatik.uni-muenchen.de. Provided these mappings can be changed programmatically, the DNS may be used for implementing very simple grouping schemes for application and network entities. Currently, some protocol extensions are under development to support dynamic updates of the DNS, however there is still no management information defined. The Dynamic Host Configuration Protocol (DHCP) [3] provides support for dynamic (re-) configuration of end systems by supplying them with network interface characteristics and addresses of basic Internet services, e.g. DNS and SMTP server and router address. If DNS and DHCP servers are connected via *Dynamic DNS (D-DNS)*, a standardized basic configuration scheme in intranets can be established as required for implementing the flexibility required in the intranet scenario described before.

*As another example an IP network address could be used to define a VLAN combining all hosts using this network address.

Applications Grouping at the application layer most often only lends itself to areas in which membership must be specifically authenticated and in which performance of the communication and the provisions to support the authentication are not of primary concern. If a single network is used by multiple subscribers or by nomadic systems from other subscribers these concerns are usually outweighed by security and accounting requirements. Examples would be specific rights to access printing or email relaying services or to make reservations on networking resources.

Naming Services At almost every layer a sort of naming service is usually employed that already allows for a sort of grouping by using a common "prefix" in the naming hierarchy, e.g. NIS, WINS and DNS as well as the Corba Naming Service within a (Corba) application. This prefix implicitly defines a group.

A very recent development is the wish to support nomadic computing as well as Ad-Hoc-Networking*. With the increasing appearance of team–oriented as well as teleworking structures there is an additional need of proper exploitation of an existing infrastructure without causing too much administrative load for the necessary change management [7].

Additionally, an increasing need for properly configuring application services and resources with respect to the allowed users comes from accounting and security requirements. An example from the application layer consists of the group of email client entities that might be allowed to use a given Mail Transfer Agent (MTA) as a relay host. In this case not only the client must be told which MTA to use, but also the MTA must be configured to accept email from a previously unknown client (see section 4.2). Further examples may be drawn from the area of software license, printer management and from the scenarios for which the *Service Location Protocol* (SLP) [23] is being developed for.

The standards described above only provide the technologies to implement management policies. The lack of standardised management interfaces for the entities implementing the technologies together with the lack of commonly understood semantics in policies used by different management domains renders automated management for complex subscriber–user–provider relationships impossible.

If, however, semantic of management operations is defined (which is beyond the scope of this paper) there will be the need to find adequate definitions for generic management information to be exchanged between managing entities with different responsibilities as well as within a complex, heterogeneous managed environment.

3. The Group Object Model

Since there is a need for grouping managed objects in a dynamic, flexible manner in the transport system and in distributed client/server applications and since there are multiple implementation technologies on various layers (as shown in section 2)

*This term describes the interconnection of systems physically located close to each other for purpose of forming a temporary network, e.g. during an meeting or at a fare.

we can expect to be confronted with a number of management systems addressing specific technologies or vendors resp. a combination of those.

This heterogenity already arises: Cisco VlanDirector manages VLANs formed by Cisco switches, Fore ATM switches want their VLAN capabilities being managed by ForeView VLAN Manager, 3Com has the Transcend Enterprise Manager and so on. Every vendor of a commercial DHCP/D–DNS server (like Microsoft, Quadritek, Join Systems and Metainfo) has a management tool of its own and the same holds for management of applications (email, storage, printing, etc.). None of these management tools provides a standardized management interface. They are technology–specific and hardly work across customer–provider boundaries that are present in the office park scenario. Taking the evolution of technology–specific management tools as a given fact we focus on integrative and generic aspects by presenting a generic management model for the management of groups.

In the following, requirements for the group object model arising from technologies and applications described in section 2 are used as a basis for its development. As mentioned in section 1, [10] defines an *management domain model* using *jurisdictions* to associate *policies* with a set of management objects called management domains. We built upon that object model and modified it by appropriate objects, relations and notifications.

Another standardization approach, the OMG Topology Service [16], has been abandoned due to the lack of interest on the side of manufacturers of management platforms. Nevertheless, it provides valuable concepts for implementing associations between managed objects.

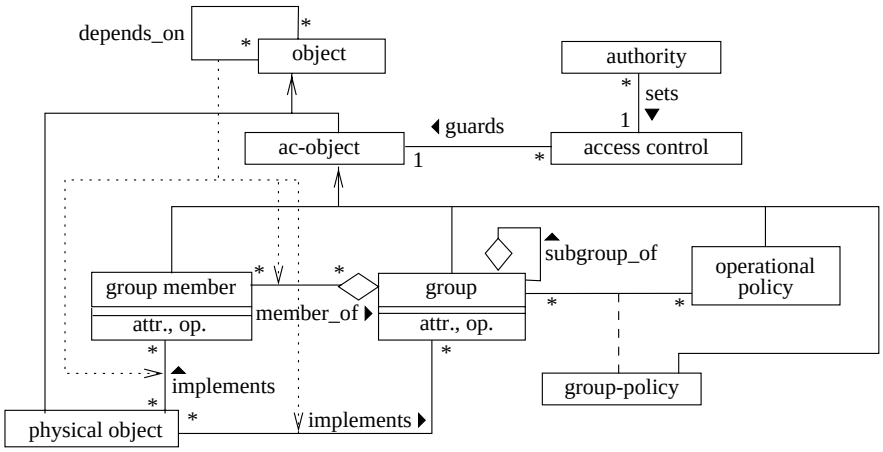


Figure 4 The *group* object model

Figure 4 gives an overview of the group object model in UML notation [20]. The main classes within the model are *group* and *group member*. A group can contain

other groups and group members. Thereby hierarchies of groups can be modeled. An operational policy can be applied to a group; A group–policy object is used. For each association between a policy and a group. This way the group defines on which objects a policy is enforced.

For modeling the relation between members and groups, we use the *member_of* association which is a refinement of the *depends_on* association. The dependency between a group and its implementation is modeled by an association to a physical object implementing the group. A physical object could be a switch implementing VLANs. Of course, some more detailed object models for the physical object will exist but are out of scope for this work. From the viewpoint of event correlation and generic configuration the association between logical objects and physical objects is required. Additionally, as dependencies may possibly occur between all objects, the *depends_on* association is modeled at the generic object class and specialised by several associations describing the concrete dependencies between the derived object classes. This technique can also be found in the Common Information Model [1].

To restrict access to attributes and operations all objects mentioned above are derived directly from the generic object class *ac–object* associated with the access control object class defining the access control rules and indirectly with the authority object class describing the responsibility for defining the access control rules. The responsibilities for defining groups, policies and group–policies are defined indirectly by setting access rights for their operations. A realization of the access control could be based upon the Corba Security Service [17].

Attributes and operations are presented only for the classes *group* and *group member* being in the center of our research: a *group object* is characterised by a *group object identifier*, a membership specifier for group members and subgroups* and a reference for an access control object class instance. The *group object class* provides operations as *create*, *delete*, *enter group member*, *remove group member*, *create subgroup*, *list direct group members*, *list subgroups* and *verify group membership*. If a group member is added to the group object the latter sends a notification *membership_changed* with the parameters *group object identifier*, *group member identifier* and *reason* describing whether the group member joined or left the group. This notification is useful for detecting new group members, e.g. nomadic systems appearing in the network.

The *group member* is characterised by an *identifier* and a reference to a *physical object*. When thinking of centralised management operations of adding or removing group members to the group are modeled by group object operations. For enabling decentralised management we define operations in the group member class, i.e. *join group* and *leave group* that may be alternatives for *enter group member* and *remove group member* of the group class.

*This can be list of group members and/or a filter defined by predicates selecting group members.

4. Application of the group object model

The group object model provides a generic interface towards management applications. It needs to be refined in order to be instantiated. A generic implementation beyond some frameworking is not possible because groups are always reflected in concrete technologies. So an instantation of the classes for the VLAN technology would result in an VLAN management application providing the group object model.

In order to demonstrate the benefits of providing the group object model we present management applications from the areas of fault management and configuration management.

4.1 Event Correlation and Fault Diagnosis

Fault isolation is a critical task in networks that allow for flexible, dynamic groups by providing its enabling technologies. Virtual layers hide the real cause of a fault and the search for a fault has to track changes in dynamically formed entities, i.e. groups.

Many approaches from this area rely on the functional *dependencies*, usually described by a graph, between managed objects for their algorithms. Group membership imposes a dependency: The group member needs the group or more precisely, services offered from the group as a whole resp. services offered by other group members to group members only. Therefore the group object model contributes to the problem of gaining the dependency structure from the management system in a generic manner.

For this purpose the *member_of* association is declared as a kind of dependency. Additionally, two associations were introduced connecting group members and groups to implementing entities termed *physical objects*. This is a cut through the level of abstraction in order to model dependencies to managed objects **outside** the logical group structure itself. They shall enable isolation of faults *across* technological and administrative boundaries. Using *physical object* a fault isolation tool can connect the dependencies gained from the group object model with dependencies from other sources, especially topological data from lower layers of the managed system, where in turn the group object model may be applied.

The relation between groups, dependency graphs and fault isolation upon them is illustrated by an example from the Office Park scenario involving VLANs.

Figure 5 shows an excerpt from the layer two topology of the Office Park. It includes a nomadic system connected in the meeting room and some of C_3's servers. Figure 6(a) shows a simplified dependency graph derived from that topology. It can be used to find potential explanations why the nomadic system $NoCS_1$ cannot communicate with C_3's mail hub: The problem may either has its origin in one of the participating end systems or in one of the switches between them. The topology would allow

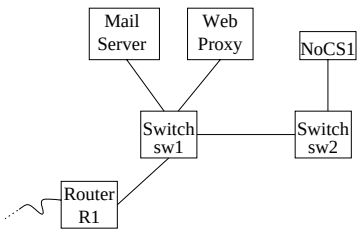
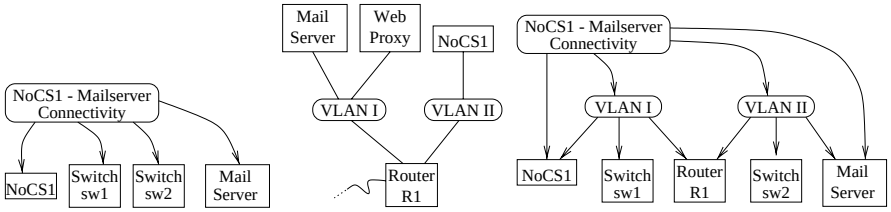


Figure 5: Physical Topology

to derive many more dependencies but this is left out for the sake of simplicity. Note that the router R_1 plays no role in this dependency graph.



(a) (Physical) Dependency Graph (b) VLAN Topology (c) VLAN Dependency Graph

Figure 6 Example: Dependency graphs for a sample VLAN

Figure 6(b) shows the same scenario. But this time knowledge on groups — in this case Virtual LANs — was used as a basis for the model: C_3 grouped its servers in VLAN I and the ports of the meeting room belong to VLAN II. The resulting dependency graph in figure 6(c) now reflects the fact that a problem in router R_1 can cause a connectivity problem between $NoCS$ and the mail hub, since inter–VLAN–traffic is routed through R_1.

An in–depth discussion of the dependency graph is out of the scope of this paper. In short: Here, the VLAN node depends on the systems being connected to the VLAN. The main argument to model it this way is that there are examples where an end system can disturb other systems within its LAN segment especially if protocols using layer two broadcasts are involved.

The VLAN dependency graph is naturally mappable to the group object model: VLANs are group objects, the systems and network devices are physical objects and the dependencies between the node representing the connectivtiy service and VLANs are derived from group membership.

Using the dependency graph there are two basic strategies to isolate faults [*]:

- Fault diagnosis tools compute and execute a sequence of diagnostic tests whenever a fault is suspected. [11] uses a dependency graph to compute the sequence of tests: When a test returns "faulty" for a given node its dependent nodes are investigated. Note that in the VLAN sample router R_1 is owned by C_3 and the switches are owned by C_2, so obtaining management access to all the resources may be a problem.
- Event correlators [25] condense (many) symptoms, i.e. input events, to (few) events indicating the actual fault accurately. In [5] an algorithm based on dependency graphs is outlined: After mapping events to nodes of the graph the corre-

[*] Other applications of dependency graphs are for example the measurement of availability as outlined in [2].

lation engine searches along dependencies for common dependend nodes that are reported as potential causes.
Since we look at a scenario with a decentralized management, even fault detection may become problematic: The event correlator should be fed with many events – potentially from multiple administrative domains – in order to correlate close to the actual fault. Therefore we need standardized events in terms of protocol and meaning. This issue is addressed in management architectures, however, not yet sufficiently at all. Another aspect differentiating management architectures is the flexibility in the routing of management events. Corba promises (and partly provides) a higher degree of flexibility compared to SNMP due to decoupling event source and sink using the generic Event Service extended by filtering and QoS mechanisms in the TelecomDTF's Notification Service [19] *.

Due to the simplicity of the example the need for a *generic* model was not demonstrated yet. A good VLAN / layer two management tool could implement these algorithms and perform fault isolation for this example. However, it is somewhat unrealistic to assume that connectivity problems between a notebook and its mail server are realized and solved within layer two. A "real" event correlation would get input events from a mail tool or the user itself and would have to correlate through operating systems, the mail system, DHCP configuration and so on. This is where the strength of the generics of the group object model comes into account.

4.2 Nomadic computing

Support for configuration management related to nomadic computing across multiple management domains requires generic interfaces between the managing entities of the involved systems, e.g. servers and nomadic clients. How can one ease the process of *entering* or *leaving a group* of service users, of clients of basic computing services, of users allowed to access certain resources, and others? We see the need to apply a variety of standards to exchange management–related information. However, the IETF as the standard–setting body for application and network protocols in TCP/IP networks rarely pays attention to management issues. Nevertheless, it has developed several protocols that aid infrastructure support for nomadic computing as well, namely DHCP and SLP (as mentioned in section 2). Configuration protocols — such as the Application Configuration Access Protocol [15] — are making some progress. Still, management interfaces to servers implementing these protocols are mostly proprietary. Even in areas where standard MIBs have been defined (such as [4]), the lack of expressiveness in the Internet Management SMI keeps vendors from basing management upon standard MIBs for more than monitoring purposes.

When managing the nomadic computing infrastructure there is a frequent need to exchange information between a new group member, the physical object implementing the group and the managing entities. Figure 7 shows the application of the object model within the scenario, limited to the main methods for the sake of brevity. Three (interleaved) configuration procedures are given in a simplified fashion. No fixed

* In the future the administration of Event Networks (RFP [18]) may be standardized as well.

underlying management object model is assumed although employing the Corba architecture would be beneficial due to its richness of standardised services. The technology–specific interpretation of the arguments to the operation *addmemberfilter* is not shown here for brevity, we have followed the OSI guidelines related to modeling implementation–specific details.

```
// Initialisation of the infrastructure:
gidmailsources := EmailHub.create(MailSources) // gid = group identifier
gidwebclients := WebProxy.create(ProxyClients)
...
// Upon Connection of a nomadic system:
// (a) create new VLAN
gidvlan := Switch.create(VLAN)
gidvlan.addmember(MailHub.interface) // Membership defined by enumeration
gidvlan.addmember(WebProxy.interface) // -- chose if address instead of port
gidvlan.addmember(Switch_on_floor_2.port[6]) // Point of attachment of NoCS1
// (b) configure mail server and web proxy to accept requests
gidmailsources.addmember(NoCS1) gidproxyclients.addmember(NoCS1)
// (c) configure nomadic system
gidmailrelays.addmember(NoCS1) // groups live in nomadic system
gidwebproxies.addmember(NoCS1)
```

Figure 7 Sample application of the group object model towards support of nomadic systems

These procedures implement low–level configuration policies that are intended to be carried out upon appearance of a nomadic system, e.g. of $NoCS_1$ in meeting room M of the Office Park. The abstraction layer that the group object model provides helps bridging the different management systems that might be employed for configuration of the heterogeneous network and application equipment employed in this scenario.

5. Conclusion and Outlook

From an instance of an increasingly common set of scenarios we have shown the need for further standardization of generic management objects related to configuration and fault management in a heterogeneous and frequently changing environment with multiple managing entities. It is increasingly common that these entities belong to different providers and therefore do not share a common set of policies to which they conform. This necessitates common definitions for certain management objects that are sufficiently generic yet comprehensive enough to form the basis for the exchange of management information between systems of different providers. The group object model provides this definition for a well–defined management area. At a lower level it may also be applied to abstract vendor– and technology–specific management models for their use in more generic management applications.

The group management concept is currently being applied towards the design of a vendor–independent VLAN management system. The concept will be carried on to the integration of standardized management interfaces for Intranet configuration software such as DHCP and DNS in large, heterogenous infrastructures such as the

Leibniz Supercomputing Center and BMW in Munich. Immediate benefits for an event correlation prototype under development with a research cooperation and configuration for nomadic computing in these environments are expected. The interfaces are mapped to Corba–IDL and implemented by mobile Java agents.

We are currently examining the design of standardized interfaces for the exchange of policy information with a standardized set of semantics between different management entities for further improving support of nomadic computing as well as application of the generic group object model in autodiscovery/autotopology processes.

Acknowledgment

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences.

References

[1] Common Information Model (CIM) Version 2.0. Specification, March 1998.

[2] G. Dreo Rodosek and T. Kaiser. Determining the Availability of Distributed Applications. In Lazar and Saracco [13].

[3] R. Droms. RFC 1541: Dynamic host configuration protocol. Rfc, IETF, October 1993.

[4] N. Freed and S. Kille. RFC 2248: Network services monitoring MIB. Rfc, IETF, January 1998.

[5] B. Gruschke. Integrated Event Management: Event Correlation using Dependency Graphs. In Sethi [21].

[6] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application.* Morgan Kaufmann, 1999.

[7] S. Heilbronner. Requirements for Policy-based Management of Nomadic Computing Systems. In Sethi [21].

[8] Gilbert Held. *Virtual LANs – Construction, Implementation, and Management*. Wiley, 1997.

[9] IEEE-802.1-WG. Draft Standard P802.1Q/D11 - IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. Technical report, IEEE, July 1998.

[10] Information Technology – Open Systems Interconnection – Systems Management – Part 19: Management Domain and Management Policy Management Function. DIS 10164-19, International Organization for Standardization and International Electrotechnical Committee, January 1996.

[11] S. Kaetker and M. Paterok. Fault Isolation and Event Correlation for Integrated Fault Management. In Lazar et al. [13], pages 278–289.

[12] T. Koch, C. Krell, and B Krämer. Policy Definition Language for Automated Management of Distributed Systems. In *IEEE Workshop on Systems Management*. IEEE, 1996.

[13] A. Lazar, R. Saracco, and R. Stadler, editors. *Integrated Network Management V (IM'97)*, San Diego, USA, May 1997. Chapman & Hall.

[14] P. V. Mockapetris. RFC 1034: Domain names — concepts and facilities. Rfc, IETF, November 1987.
[15] C. Newman and J. G. Myers. RFC 2244: ACAP — Application Configuration Access Protocol. Rfc, IETF, November 1997.
[16] Topology Service (RFP). Document 97-01-02, Draft 2, Object Management Group, January 1997.
[17] CORBAservices - Security Service. Document 97-12-22, Object Management Group, December 1997.
[18] Notification Service (Joint Revised Submission). TC Document telecom 98-06-15, Object Management Group, June 1998.
[19] Management of Event Networks (RFP). Document telecom 98-06-17, Object Management Group, June 1998.
[20] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language — Reference Manual*. Addison-Wesley, 1998.
[21] A. S. Sethi, editor. *9th Annual IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, Newark, DE, USA, October 1998.
[22] Morris S. Sloman, editor. *Network and Distributed Systems Management*. Addison Wesley, 1994.
[23] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. RFC 2165: Service location protocol. Rfc, IETF, June 1997.
[24] R. Wies. Using a Classification of Management Policies for Policy Specification and Policy Transformation. In Y. Raynaud and A. Sethi, editors, *Proceedings of the 4th International IFIP/IEEE Symposium on Integrated Network Management*, May 1995.
[25] S. Yemini, S. Kliger, E. Mozes, et al. High speed and robust event correlation. *IEEE Communications Magazine*, May 1996.

Biographies

Boris Gruschke, **Stephen Heilbronner** and **Norbert Wienold** studied Computer Science at the Technical University of Munich where they graduated in 1996, 1994 and 1995 resp. with a Masters Degree (Diplom-Informatiker, M.S.). They are currently persuing their Ph. D. degrees in various areas of integrated management. All authors are members of the MNM–Team and belong to the research staff of the University of Munich.

Boris Gruschke is persuing his Ph. D. in the area of event correlation using dependency graphs. His work is sponsored by Siemens. From 1993 to 1995 he worked with the Siemens R&D division on projects in the field of parallel programming.

Stephen Heilbronner worked in 1992 with Microsoft on product internationalization and from 1990 to 1994 with BMW on several projects employing distributed computing techniques in mission critical applications. His Ph. D. is in the area of infrastructure management for nomadic computing. He is a member of the GI (Gesellschaft für Informatik) and the ACM.

Norbert Wienold worked from 1993 to 1994 with ICS, a network integrator, on a network management project integrating LAN network components with the management platform SPECTRUM. His Ph. D. is in the area of management for virtual networks in cooperation with Siemens. He is a member of the GI.