RC22114 (W0107-013) July 17, 2001
Computer Science

IBM Research Report

Simplifying Network Administration Using Policy Based Management

Dinesh Verma
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Simplifying Network Administration using Policy based Management

Dinesh Verma
IBM Thomas J. Watson Research Center
30 Saw Mill River Road, Hawthorne, NY 10532

Email: *dverma@ us.ibm.com*

ABSTRACT

The management of network infrastructure in an enterprise is complex and daunting affair. In an era of increasing technical complexity, it is becoming difficult to find trained personnel that can manage the new features that are introduced into the various servers, routers and switches. Policy based network management provides a means by which the administration process can be simplified and automated to a large extent. In this paper, we look at a general policy based architecture that can be used to simplify several new technologies emerging in context of IP networks. We explain how network administration can be simplified by defining two levels of policies, a business level and a technology level. We discuss how the business level policies are validated and transformed into the technology level policies, and present some algorithms that can be used to check for policy conflicts and unreachable policies We then show how to apply this architecture to two policy disciplines - managing performance service level agreements, and supporting enterprise extranets using IP-sec communication.

1. Introduction

Present day IP networks are large complex systems consisting of many different devices. Ensuring that all these devices inter-operate smoothly is not a trivial task. New technologies that have emerged to address some of limitations of the traditional IP protocols have added to the complexity of the network infrastructure. There is an acute shortage of experts who understand the new technologies and are able to manage and deploy them on a large network. For many emerging technologies, the management costs associated with deploying the technology outweighs the advantage conferred by that technology. As an example, many pragmatic network operators choose to over-engineer their networks to address any performance concerns rather than deploy bandwidth savings Quality of Service techniques. This is because the manpower cost associated with learning the new technologies and managing them is much higher than the savings in bandwidth related costs that would result from deploying these technologies.

In this environment, there is a clear need to make new and emerging technologies easier to manage. The policy framework being standardized within the IETF [1] holds the promise to deliver this ease of management. The simplification and automation of network management process is one of the key applications of the policy framework. This paper explains how the policy management framework can help network administrators attain this simplification, and demonstrates its applicability in three different disciplines.

The next section of the paper provides an overview of the general policy based administration architecture. The third section discusses structure of a management tool that can apply to many policy disciplines, and the algorithms needed within the management tool. The section also discusses some policy validation algorithms that can be used to check for consistency and usability of network policies. Section four show how the generic framework can be applied to the areas of service level agreements enforcement and secure IP communications. Finally, we present our summaries and conclusions.

2. General Policy based Administration Architecture.

The general policy based administration framework that we present can be considered an adaptation of the IETF policy framework to apply to the area of network provisioning and configuration. The IETF policy framework is shown in Figure 1, and consists of four elements, the policy management tool, the policy repository, the policy decision point and the policy enforcement point.

An administrator uses the *policy management tool* to define the policies that are to be enforced within the network. A device that can apply and execute the different policies are known as the *policy enforcement point* (PEP). The preferred way for the management tool and PEPs to communicate is through a *policy repository*. The policy repository is used to store the policies generated by the management tool. In order to ensure interoperability across products from different vendors, information stored in the repository must correspond to an information model specified by the Policy Framework Working Group. Instead of communicating directly with the repository, a policy enforcement point uses an intermediary known as the *policy decision point* (PDP). The PDP is responsible for interpreting the policies stored in the repository and communicating them to the policy target. The PEP or PDP may be in a single device or different physical device. Different protocols are to be used for various parts of the architecture, e.g. the COPS protocol [2] or the SNMP protocol [3] can be used for communication between the PDPs and the PEPs. A repository could be a network directory server and accessed using the LDAP protocol [4].

The structure of the management tool is not defined by the IETF standards since a standard format is not needed for interoperability between different machines. In this paper, we focus on the policy management tool and how it can leverage the power of policies to simplify the provisioning and configuration of the different devices within the network. This simplification of the management functions is obtained via two elements of the policy management tool and the policy architecture, namely centralization and business level abstractions.

Centralization refers to the process of defining all the device provisioning and configuration at a single point (the management tool) rather than provisioning and configuring each device itself. In a system with a large number of machines, the centralization of configuration at a single reduces the manual effort required of an administrator. The administrator inputs the policies needed for network operation into the management tool which populates the repository. The information in the repository is specified in terms of the technology being deployed within the network, e.g. using terms and concepts available for managing performance using the Differentiated Services technology. The PDPs retrieve the policy defined in the technology-specific notation, and convert it into the appropriate configuration of the PEP that can enforce the desired policies. The benefits of centralization on reducing manual tedium can easily be seen. In a network which requires configuring a 1000 machines where configuring each machine requires 10 minutes, an administrator would need to spend over a week working non-stop to configure the devices manually. In a policy based solution, the administrator would need to spend only about 15 minutes populating the repository with the appropriate policies. The system of PEPs and PDPs would cause the devices to get configured in an automated manner. The centralized approach is likely to get the configuration done much faster, with a smaller likelihood of erroneous configuration.

Business Level Abstractions make the job of policy administrator simpler by defining the policies in terms of a language that is closer to the business needs of an organization rather than in terms of the specific technology needed to deploy it. The administrator need not very conversant with the details of the technology that supports the desired business need. As an example, let us consider the case of a network operator that needs to define two levels of customers, one being premium and the other being normal. It is fairly simple for an administrator to identify each customer and to define which level they may map to. One way to support the business need for bi-level customer support to use IP Differentiated Service technology within the network. If the administrator wants to define policies in using the technology level abstractions, he needs to be familiar with the jargon of the technology, e.g. be aware that differentiation is obtained by assigning traffic to different PHBs (Per Hop Behaviors) and that the premium customer be mapped to a specific PHB (e.g. EF-PHB "Expedited Forwarding"), and then define some parameters for this PHB. The jargon of the technology (PHB, EF-PHB, configuration parameter details) requires special knowledge that is harder to find than the familiarity with the business needs of the organization.

The business level abstractions depend on the business needs and the technology that the policies are being defined for. The business needs of an organization may be satisfied by many different technologies. A policy discipline is a two-tuple consisting of a business need and the technology that supports it. A business need, such as supporting performance SLAs on a network may be satisfied by technologies such as capacity planning [5], Integrated Services [6], Differentiated Services [7] or Content Distribution [8]. A business need such as establishing a secure virtual private network may be satisfied using IP-security [9] or TLS protocol [10]. While each discipline would have some aspects of policies that are specific to its details, many operations related to policies can be performed in a generic manner.

3. The Policy Management Tool

As discussed above, the policy management tool needs to support the notion of policies specified as business level abstractions as opposed to the policies specified as technology-level abstractions. A generic policy management tool that will support these two levels of policies can be constructed out of the four basic components as shown in Figure 2.

The *user interface* is the means by which an administrator can input the business-level policies within the network. These policies are the ones that need to be enforced in the network, and the configuration of the network must be made conformant to these high-level policies. The interface may consist of command lines or a graphical tool that can be used by the administrator. Command lines permit programmatic manipulation of the policy management tools.

The *resource discovery* component determines the topology of the network and the users and applications that are operational in the network. In order to generate the configuration for the various devices in the network, the capabilities and topology of the network must be known. For any moderately sized networked system, such topology and capabilities must be discovered automatically.

The *policy transformation logic* component is responsible for ensuring that the high-level policies that are specified by the network administrator are mutually consistent, correct, and feasible with the existing capacity and topology of the network. It also translates the business-level policies into technology-level policies that can be distributed to the different devices in the network.

The *policy distributor* is responsible for ensuring that the technology-level policies are distributed to the various devices in the network. If the network complies with the IETF policy architecture, policy distribution consists simply of writing the technology level policies (low-level policies) to the repository. If there are devices that do not conform to the IETF architecture, the distributor has to perform extra work. One possible way to support non-conformant devices is to convert the low-level policies into the appropriate device configuration, and send the configuration to the device over a command line interface.

The user interface, resource discovery and the policy distributor (in-accordance with IETF guidelines) are components that have a relative straight-forward design. This is not to discount their importance; a good user interface can make the difference between an unusable tool and a great tool. However, the heart of policy management lies in the policy translation logic, as to how the policies will be represented and how they will be managed.

3.1 The Policy Translation Logic

The policy transformation logic module validates the information provided in the high-level policies and transforms them into the configuration of devices in the network. The logic furthermore ensures that the policies specified are mutually consistent and that they cover all aspects of interest to the network administrator. The validation process must incorporate syntactical checks as well as semantic checks. The semantic validation of high-level policies consists of various types of checks:

- **Bounds checks:** validates that values taken attribute in the policy specification are within specific limits that are determined by the network administrator.

- **Relation checks:** validates that the value taken by any two parameters in the policy specification satisfy a relationship determined by the specific technology.

- **Consistency Checks:** This validates that any two policies defined by the administrator do not conflict with each other.

- **Dominance Checks**: This checks for “unreachable policies”, i.e. policies that are defined by an administrator but will never become active in the network because they have rendered ineffective by the definition of other policies.

- **Feasibility Checks**: This checks that the set of policies desired by an administrator for a network are feasible in the operating environment provided by the network.

Some of these checks can be made in a generic manner independent of the policy discipline. The translation process and the feasibility checks depend on the policy discipline that is being supported. However, given a suitable policy representation, it is possible to perform the bounds, relations, consistency and dominance checks in a discipline-independent manner.

3.2 Policy Representation

The high-level and low-level policies required for network management can be specified in many different ways. Among the researchers who are involved in specifying policies, multiple approaches for policy specification have been proposed. These approaches range from an interpretation of policies as programs to an interpretation of policies as simple entries in a directory or database.

From a human input standpoint, the best way to specify a high-level policy would be in terms of a natural-language input. You would like to specify the objective of operating a computer network in a simple English sentence such as "All communication between the research site and the engineering site needs tight security." Although these policies are very easy to specify, the current state of natural-language processing, a special area within the field of artificial intelligence, needs to improve significantly before such policies can be expressed in this manner. For example, it is unclear what the term "tight security" in a policy implies.

The next approach is to specify policies in a special language that can be processed and interpreted by a computer. This maps a policy to a piece of software that can be executed by a computer under certain conditions. Examples of this approach include CacheL, a language for specifying caching policies [11] and LaSCO [12], a language for specifying security policies in the network. Another approach is to specify the policy using a formal specification language. Examples of such an approach include GRAIL [13] and PONDER [14]. When policies are specified as a computer-interpretable program, it is possible to execute them. However, in general it is quite difficult to determine if the policies specified by two different programs are mutually consistent.

A simpler approach is to interpret the policy as a sequence of rules, in which each rule is in the form of a simple condition-action pair (in a “if-then-else” format). The rules are evaluated on specific triggers, such as the passage of time or the arrival of a new packet within the network. If a rule's condition is true, the action is executed. A sample specification of the policy in this format would be "If the packet's source or destination IP address belongs to the research or engineering subnet, encrypt the packet." Policies specified in this fashion are easier to analyze than policies specified as full-blown computer programs or by a formal specification language.

Representing policies using "if-then-else" semantics may lead to the conclusion that policy representations should be evaluated using an expert system or theorem-proving approach. Although such an approach toward network policies is feasible, this may not be the appropriate approach for the problem. For the most important problems that are of relevance in an application of policies within the network, expert systems-based approaches have several limitations. As an example, checking the mutual consistency of a set of policies using the theorem-proving approach requires exponential running time in many instances.

An alternative specification of policies is to represent them simply as entries in a table. The table consists of multiple attributes. Some of these attributes constitute the condition part, and others constitute the action part. Different types of tables need to be specified if the condition components or action components of different rules vary. Such a tabular representation is rich enough to express most of the policies that can be specified with a rule-based notation. Furthermore, it is easier to analyze for dominance and consistency.

The IETF [1] has chosen a rule-based policy representation in its specification. However, due to the need to store this representation in an LDAP directory or database, this representation essentially follows the tabular specification just described. For a variety of policy disciplines that arise in the field of TCP/IP networks, we have been able to use such a tabular specification of policies to capture most of the practical scenarios one may encounter.

3.3 Policy Validation Algorithms.

When we use a tabular representation for the policies, each policy discipline can be characterized by a description of the set of tables in the policy definition for the discipline, and the set of columns that make up each table. We refer to this description as a policy schema. A column of a scheme defines an attribute of the policy which could be a simple (textual or numerical) attribute, a structured attribute consisting of multiple attributes (e.g. an IP Subnet attribute consists of the simple attributes of SubnetAddress and Prefix Length), or a nested table (e.g. the table of interfaces at a computer). Different types of validation criteria can be associated with each table and column of the attribute.

The validation criteria enable some of the checks to be performed in a very simple manner. By associating a limit checking criteria with each column, the bounds checks can be performed rather trivially. Similarly, the relations checks can be performed by defining a relationship criteria associated with a table. Each row in the table is validated against the relationship criteria.

The checking for policy conflicts and dominance needs to be performed across all of the rows that of a table. The consistency criterion is that if two rules can both apply under some conditions, the actions to be performed must be uniquely identified and be doable simultaneously.

3.3.1 Conflict Resolution

As an example of conflicts among different policies, consider a simple example, where two classes of service (Gold & Silver) are defined within the network. An application called WebServer is defined to operate on TCP protocol and the port number of 80. A set of users with IP addresses in the subnet 9.2.34/24 is defined as HighPowerUsers. Two policies are defined as follows:

P1: Any access to WebServer gets Silver service.

P2: Any use of the network by HighPowerUsers gets Gold service.

Both of these rules are perfectly okay by themselves, but there's a conflict when the two are taken together. In the case of a HighPowerUser who is trying to access WebServer, it is unclear whether the Gold service or the Silver service should be provided.

One approach to detecting conflicts among the different rules is to look upon each policy rule as consisting of multiple independent terms and one or more derived terms. A policy rule consists of the generic form if-condition-then-action. Let us impose the restriction that the terms that define the condition be distinct from the terms that define the action portion of the rule.. Business SLA policies as well as security policies are often defined in terms of classes of service (dealing with performance or security). Each class of service combines several actions that can be taken together.

Consider the different terms make up the condition part of a policy expressed in the format, if condition then action. Each of the independent terms can be looked upon as independent axis in a hyper-dimensional space. Each rule defines a region in the hyper-dimensional space. Each such region can be associated with a dependent term (such as the service class) that is identified by the rule. If any point in space has multiple dependent terms that conflict with each other, you have a potential conflict. For example, consider the case of policy definitions that have two independent terms. Each of the policy definitions would carve out a two-dimensional space. If the regions defined by two policies do not overlap, they do not conflict. If two regions overlap, the corresponding policies might have a potential conflict if the dependent terms in the policy definition can't be done together.

For a more specific example, consider the simple example previously mentioned about HighPowerUsers and Webserver. The two independent axes in this case are the applications (identified by their port numbers) and the users, identified by their IP address. These two rules define regions in a two-dimensional space, with the first dimension being IP address and the second dimension being the port numbers. The first rule carves out a region defined by the line obtained by keeping the port dimension fixed at 80. The second rule is the square region carved out by the subnet 9.2.34/24 (with a lower of range 9.2.34.0 and upper range of 9.2.34.255). The line intersects with the region and the common region contains a conflict with two different classes of service.

More realistic cases of policy definition would tend to have more independent terms and corresponding dimensions defining the policies. However, if one defines the dependent and independent terms for each policy table, along with a function for each independent term that checks whether there is an overlap between two values of that term, the algorithm can be implemented in a very simple fashion with a running time of $O(n^2)$ where n is the number of policies. If a conflict is found, one way to resolve them is to assign them different priorities. Since the priority can be considered an independent term for conflict resolution, policies with different priorities will not result in overlapping regions in the hyper-dimensional space.

3.3.2 Dominance Checks:

The dominance criterion checks that a policy is actually applicable in some condition that can arise during operation. The dominance checks are also designed around the concept of the hyper-dimensional space. To check for this, we map each policy into the independent and dependent terms as before. We also assume that there is a function that takes two policies and determines which of them will dominate in a region of the overlap. With this information, the dominance check for a single rule consists of comparing it against all the other policies that overlap with it and dominate it.

We start with a list of hyper-dimensional regions initially consisting of only one hyper-dimensional region that is defined by the policy rule that we are checking for dominance. Then we remove the region described by each dominating and overlapping policy from all the regions in the list successively. After all the policies have been compared, we examine the resulting list. If the list of hyper-dimensional regions is empty, the policy is unreachable and can be removed without any penalties. Otherwise, the final regions represent the combination of independent terms where the policy would remain applicable.

The worst case running time of this algorithm is $O(n^{k+1})$ where n is the number of policies to be compared, and k is the types of independent terms that are used to define the hyper-dimensional space. While this may appear rather inefficient algorithm, practical management systems are not likely to find it a problem. The number of independent term is usually in single digits, and the algorithm is thus polynomial. Another factor which helps considerably is the fact that the running time for a single policy dominance is $O(n_2 + n_1^K)$ where n_1 is the number of policies that overlap with the given policy and n_2 is the number of policies that do not overlap with the given policy. Since the number of overlapping policies is only a small fraction of the total number of policies, the expected time for checking the dominance of all policies is $O(n^2)$.

3.3.3 Discipline Specific Procedures:

The translation of business level policies to a technology level policy and the feasibility checks are discipline-specific procedures. The exact method to translate the business level abstractions to a specific technology has to be defined on a per-discipline basis. However, the policy management tool provides a common framework within which the translation procedure can be performed. The common framework consists of defining the rules which provide the mapping from the tables defined as business level policies to the tables that define the technology level policies. Once these rules are defined for a discipline, the policy management tool can use them to perform the translation in a generic fashion.

As an example, let us assume that the business level tables as well as technology level translation are represented in XML[15]. A representation of the table driven policies is straightforward to do in XML. A different XML rule specification is needed for the business level abstractions and the technology level abstractions. The translation rules consist of XSLT [16] mappings which can be applied by the policy management tool. While the XSLT rules have to be defined on a discipline-specific basis, the management tool performs the translation in a generic fashion.

The feasibility checks also need to be performed in a discipline specific manner. For the case of performance related SLAs, the feasibility check require ensuring that the current network topology and the features of the technology being used can meet the desired performance goals. This may require the use of performance evaluation schemes. For the case of secure communications, the feasibility checks would require checking the compatibility of IP-sec capabilities at two devices.

4. Some Example Policy Disciplines

Having provided an overview of the generic policy management, we will now consider how the policy management tool for a couple of disciplines. The two policy disciplines that we will consider are (a) the support of performance based service level agreements using the IP Differentiated Services Technology and (b) The support of Enterprise Extranets using IP-sec protocol suite.

For each of the policy discipline that we want to apply the generic management architecture to, we need to do the following tasks (i) define the policy schema for the business level policies (ii) define the policy schema for the technology level policies and (iii) define the discipline specific translation rules and (iv) define the nature of any discipline specific feasibility tests.

For demonstrating the examples with policy discipline, we would consider the environment as that of an enterprise network. The enterprise network consists of several campus networks that are connected together by means of wide area or backbone links. We would use some simple policy schemas for both the business level and technology level policies, and illustrate them in the UML notation.

The policy management architecture can be used for many other disciplines and in business environments such as a network services provider or an application services provider. For a more detailed discussion of the other disciplines and the environment, please refer to [17].

4.1 Service Level Agreement using Differentiated Services.

Within an enterprise environment, one of the components of the business SLAs enforced upon the I/T department specifies desired objectives for *application performance*. The objectives for application performance would typically require bounds on the response time of applications running on the various servers. As an example, an performance objective may look like “A mail message of less than 100 Kilobytes should become available to a client in less than half a second”. In general, a SLA consists of many such performance objectives that need to be satisfied simultaneously.

In Figure 3, a very simple object model of the business level policies that can be used to support business SLAs in an enterprise environment is shown using UML notation [18]. The business level policies include the following types of objects:, namely clients, servers, applications, SLA, performance objectives and class of service.

A SLA or Service Level Agreement is an aggregation of several performance objectives. Each objective defines a association between a client, an application, a server, and a class of service. Semantically, a performance objective states that traffic flows used by a client to access an application running on a specific server be mapped to one of many classes of service. As an example, consider a client called Accounting, which is accessing an application called SAP running on a server called business-server. A performance objective may state that accounting’s access to SAP running on business-server be given Gold class of service.

In the object model of Figure 3, a *client* represents users within the network. Each client has two properties: a name (clientName) that identifies it uniquely, and the subnet address, identifying the location of the machines used by these users. A *server* is a machine where applications run, and its properties include a name and IP address(es) of its interfaces. We assume that applications run on well-known port numbers or range of port-numbers on each of the servers. Thus, an *application* has the properties of name, a port-range, and a protocol. A *class of service* defines a level of performance. Its properties include a name, response time and an evaluation period. The response time is the expected application response time for any traffic flows that map into this class of service. The evaluation period states how long measurements must be taken in order to determine the response time. As an example, the Gold class of service may have a response time of 500 ms, for an evaluation period of 1 hour. Any traffic flow that maps into the Gold class of service is required to have a response time of 500 ms or less when averaged over intervals of an hour or more.

The *performance objective* provides an association between a client, an application, a server and a class of service. An objective has an association with exactly one class of service, but the association could be with more than one clients, applications or servers. An objective could only be valid at specific times of day, which is one of the attributes shown for the performance objective in Figure 3.

The low level policies for the mapping would consist of the policy schemas that are defined for the differentiated services technology. An example of such a specification for Differentiated Services is shown in Figure 4. Multiple devices with the same set of policies are mapped into a device role. For a device role, multiple network levels are defined, each network level corresponding to one of the many DiffServ PHBs that can be used within the network. The DiffServ policies map the traditional IP 5-tuple (consisting of the source and destination addresses, ports, and protocol) to one of the network level. While this set of DiffServ policies are very different than the framework defined by the IETF, the set of policies conformant to the object model shown in Figure 4 can be easily mapped into the notation of the IETF workgroup [19].

Let us consider that an expert user has defined the rules that specify the mapping of the classes of services defined as per Figure 3 into the network levels defined as per Fiure 4. The characteristics of a Gold application, as well as the amount of bandwidth to be allocated to this class, is determined by the expert user. Thus, an expert user may define that one should use the class selector PHBs within the network, with the Gold class of service to correspond to the highest priority service, the Silver class of service to correspond to the medium priority service, and the Bronze class of service to correspond to the default service, with a maximum bandwidth limit of 80% of a link's capacity to be used by applications in the default class of service.

The policy tool uses the network topology to determine the set of access routers and servers that are relevant for each business level policy. For each device, the relevant rules are collected together. Devices with the same set of policies are collected together in a common device role. The management tool then uses the translation tables provided by the expert user to determine the correct marking behavior for all the servers and the access routers. For the core routers themselves, the policy management tool must generate a consistent mapping of the ToS encoding to the right priority level along the forwarding paths of the routers.

Supporting Enterprise Extranets using IP-security.

Enterprises establish extranets in order to automate their business processes with other enterprises, e.g. with their contractors and suppliers. An *extranet* allows a business partner to access part of the enterprise infrastructure. We assume that the following entities are involved in establishing the extranets which we are examining here.

A Extranet Client Application: This client application runs in the demilitarized zone (DMZ) of a business partner, which is a supplier to the Enterprise in this example. It runs on a machine that has the support for IPsec. These types of machines are the only entities in the supplier's environment which are allowed to communicate to a set of servers within the enterprise.

A Extranet Server Application: This application runs in the DMZ of the enterprise, and communicates with the extranet client applications that are operational in the supplier's DMZ. The extranet server application runs on extranet server machines.

Furthermore, a policy management tool and a policy repository are required in compliance with the IETF policy architecture. It is assume that the machines running the extranet client application and the extranet server application implement the IETF PEP and PDP functionality.

The UML diagram illustrating the business level policy schema for enterprise extranets is shown in Figure 5. An *extranet* definition allows a set of business partners to access a set of applications that are running on some servers within the enterprise. A *business partner* may have more than one machines at its site where the extranet client application is operational. Thus, the business partner contains an association to multiple *extranet client* machines. Each extranet client houses the extranet client application, and is identified by its name and IP address. The extranet definition allows some machines to become accessible to external business partners. These machines (the *extranet servers*) have a name and an IP address as their attributes. Only some *applications* running on the servers may be made accessible to external business partners. These applications are identified by their name, and their other attributes include the ports they run on, and the protocol these applications use for communication. An extranet is associated with one or more business partners, one or more extranet servers, and one or more applications. Each extranet definition allows extranet clients belonging to associated business partners to associated applications that are executing on associated extranet servers.

Each extranet is associated with exactly one security class. The security class defines the type of security that needs to be provided to the traffic flows that form part of the extranet definition. The details of the security class are provided within the low level technology specific definitions, and are described in more detail in the following subsection.

All the extranet definitions taken together constitute the high level policy in this environment.

Figure 6 shows a object model to represent the technology level (low-level) policies for IP-sec. An instance of *security policy rule* maps an instance of *communication tunnel* to an instance of *security class*. Using the “if condition then action” representation of the policy rule, the security policy rule uses the associated communication tunnel as the condition part, and the associated security class as the action part.

Each instance of a security class is associated with one instance of *phase one parameters*, and one or more instances of *phase one transform*. Similarly, it is associated with one instance of *phase two parameters* and one or more instances of *phase two transform*. Each instance of a phase one transform defines a set of acceptable encryption/authentication algorithms that can be used for phase one communication within IP-sec. Each instance of phase one parameters contains values of various parameters such as the duration after which keys for phase-one communication must be renegotiated. An analogous explanation holds for the phase two counterpart of the transforms and parameters. Depending on the phase of communication with a remote party, the associated instances of transforms and parameters dictate the operation of the IPsec protocol engine. The description of phase 1 and phase two of IP-sec can be found in the RFCs describing them [9].

Each security policy rule is associated with only one security class, and each communication tunnel is associated with only one security policy rule. However, a security class may be associated with more than one security policy. Similarly, instances of phase one transforms, phase one parameters, phase two transform and phase two parameters can be shared across multiple instances of security classes. The multiplicity values shown on the various associations have been

Please note that the IPsec object model shown in Figure 6 is a simple model intended for illustrative use within this paper, and can be mapped to the standard representation used by the IETF [19], but does not follow the standard definitions verbatim.

In order to translate the high level policies as expressed in Figure 5 to the low level policies as expressed in Figure 6, we need to map the definitions of the extranets to a set of secure communication tunnels, and then generate the right associations between the communication tunnels and the phase one and phase two parameters and transforms.

As in the case of the enterprise SLA, we presume that an expert user (e.g. the Chief Security Officer of an enterprise) would determine an appropriate definition for a security class. As an example, a security class named “secure” might be defined as using the IP-sec Authentication header protocol without encryption of packets while a security class named “ultrasecure” might be defined as using IP-sec Encapsulating Security Payload protocol with both authentication and encryption. These definitions have to be based on an object model as well. This description will essentially map a security class to a set of policies used for the low level policy definition.

In order to translate the definition of extranets into secure communication tunnels, the policy translation tool creates one secure tunnel between each participating extranet client application machine and the named participating extranet server application machine. It then determines the appropriate mode in which the security transformation must take place. If the point where IP-sec transformations happen is the same machine as the end-points of the communication, one can use the transport mode. Otherwise tunnel mode needs to be used.

Once the set of secure communication tunnels to be established has been determined, we can proceed with determining the relevant set of tunnels for each firewall/machine involved in the extranet.. From the set of relevant tunnels at each device, one could determine the right set of phase one and phase two tunnel descriptions to be used for IP-sec policies that will then be populated into the policy repository. The different firewalls involved in the process can then reconfigure themselves.

Summary

We have provided an overview of a policy management tool that can simplify the task of managing network configurations by centralizing the policy definitions at a single location, and by providing business level abstractions for defining policies rather than using technology specific policies. We have explained the structure of the management tool, and examined the different validation algorithms that can be used within the tool. We have also illustrated how the policy management tool can be used to support extranets and performance based SLAs in an enterprise environment. A similar paradigm can be used to support security and performance policies in many different business environments.

References

[1] The IETF Policy Framework Working Group:. Charter available at the URL http://www.ietf.org/html.charters/policy-charter.html.

[2] The IETF Resource Allocation Protocol Working Group. Charter available at the URL http://www.ietf.org/html.charters/rap-charter.html.

[3] William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* Addison Wesley, ISBN 0201485346, 1999.

[4] Tim Howes, Mark C Smith and Gordon S Good, “Understanding and Deploying LDAP Directory Services”, MTP, ISBN 1578700701, 1999.

[5] Robert S. Cahn, Wide Area Network Design, Morgan Kaufmann Publishers, ISBN 1558604588, 1998

[6] Paul Ferguson and Geoff Huston, *Quality of Service: Delivering QoS on the Internet and in Corporate Networks,* John Wiley & Sons, ISBN 0471243582, 1998

[7] Kalevi Kilkki, *Differentiated Services for the Internet,* Mcmillan Technical Publishing, Indianapolis, IN, 1999

[8] Akamai Technologies Inc., "*FreeFlow content distribution service,*" www.akamai.com.

[9] D. Maughan et al, Internet Security Association and Key Management Protocol (ISAKMP), Internet RFC 2408, November 1998.

[10] T. Dierks and C. Allen, *The TLS Protocol Version 1.0*, Internet RFC 2246, January 1999.
[11] J. Fritz Barnes and Raju Pandey. ``CacheL: Language Support for Customizable Caching Policies. Proceedings of the Fourth International Web Caching Workshop (WCW '99), March 1999, San Diego, California, USA.
[12] James Hoagland, "Specifying and Implementing Security Policies Using LaSCO, the Language for Security Constraints on Objects". Ph.D. Dissertation, University of California, Davis, March 2000.
[13] R. Darimont, E. Dalor, P. Massonet and A. Van Lamsweerde, "GRAIL/KAOS: An Environment for Goal Driven Requirements Engineering", Proceedings of the 20th International Conference on Software Engineering, Kyoto, April 1998, pp. 58-62.
[14] N. Damianou, N. Dulay, E. Lupu, and M Sloman, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems", Imperial College, UK, Research Report DoC 2001, Jan. 2000.
[15] Extensible Markup Language, Specification available at URL http://www.w3.org/XML.
[16] Extensible Style-sheet language Transformations, Specification available at URL http://www.w3.org/TR/xslt.
[17] D. Verma, Policy enabled Networking, New Riders Publications,
[18] M. Page-Jones, "Fundamentals of Object Oriented Design in UML", Addison-Wesley, ISBN 020169946X, 2000.
[19] B. Moore et. al., *Policy Core Information Model -- Version 1 Specification*, RFC 3060, February 2001.

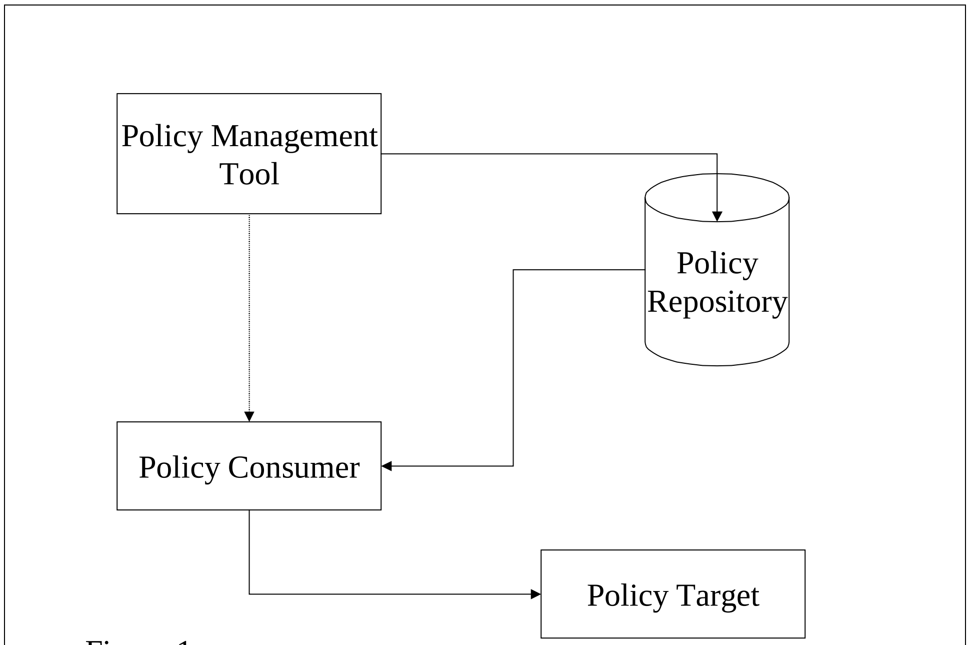

Figure 1

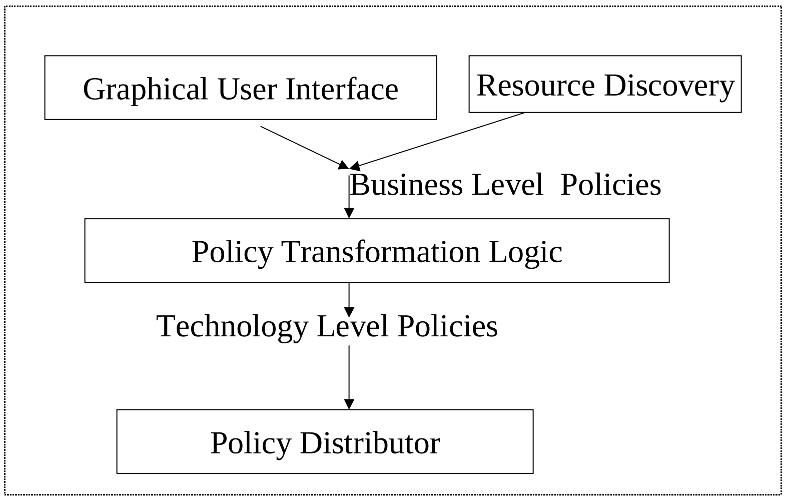

Figure 2

-x-

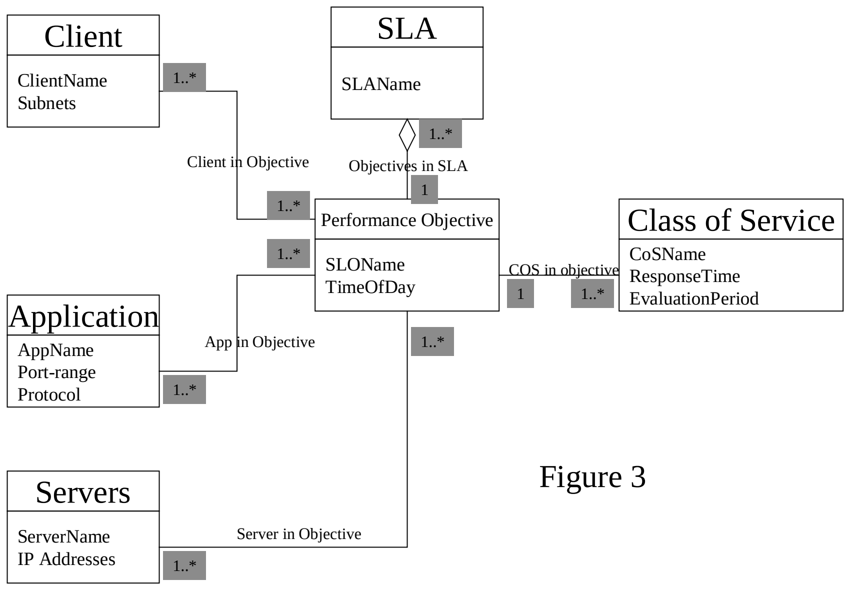

Figure 3

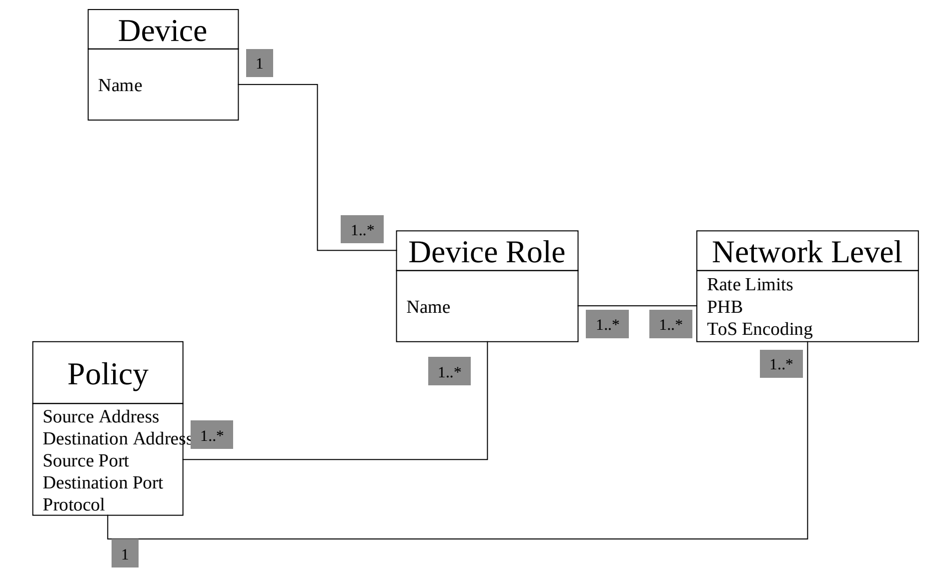
Device
Name
1
1..*
Device Role
Name
1..*
1..*
Network Level
Rate Limits
PHB
ToS Encoding
1..*
1..*
Policy
Source Address
Destination Address
Source Port
Destination Port
Protocol
1..*
1

Figure 4

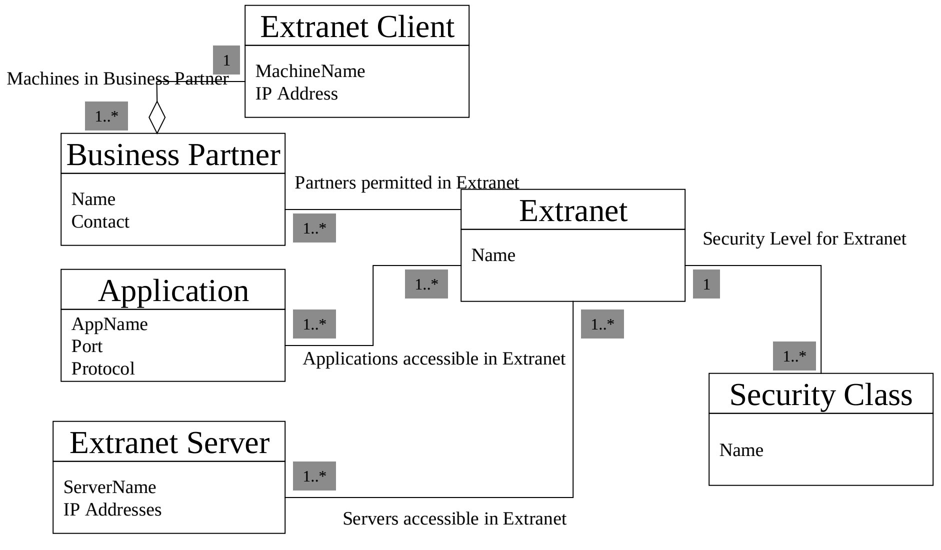

Figure 5

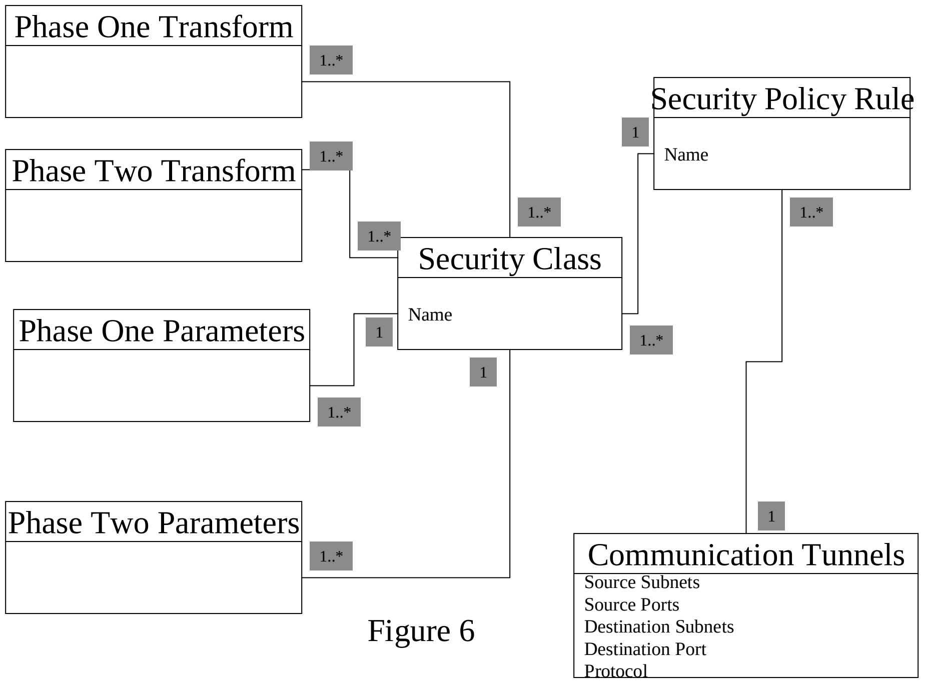

Figure 6