MASTER OF SCIENCE THESIS

Telia ProSoft AB

Directory Enabled Networks, DEN

by

Per-Erik Andersson
(per-erik.a@home.se)

Stockholm, February 2000

Supervisor:
Arne Lindgren and Rune Wahlgren
Telia Prosoft AB
Vitsandsgatan 9, 12386 Farsta
E-mail: Arne.S.Lindgren@telia.se
Rune.H.Wahlgren@telia.se

Examiner and Supervisor:
Gunnar Karlsson
KTH Teleinformatics
Electrum 204, 16440 Kista
E-mail: gk@it.kth.se

Abstract

A directory is a special purpose database that contains information about the nodes, or devices, attached to a network. Directories offer a potentially powerful tool in helping to simplify and automate many of the complex tasks involved in managing a large network. At the Microsoft Professional Developers Conference in September 1997, Cisco Systems Inc. and Microsoft Corp. announced the Directory Enabled Networks initiative (DEN-initiative). It is based on the Common Information Model (CIM) and X.500 and uses LDAP as core access protocol. One important part of the DEN-initiative is the directory support for Policy-Powered-Networking, which makes it possible to manage the network with certain rules (i.e. policies).

This thesis gives you an overview of the DEN-initiative and its protocols. Policy-powered-networking and the protocols, which should become standard for policy transactions are topics that are thoroughly discussed. The thesis ends up in a proposal for how Telia could design their network based on the DEN-initiative.

Table of Contents

1. Introduction

1.1. Background

A directory is a certain type of database that contains information about nodes, users, clients and addresses belonging to one or more networks. For a long time, directory services were considered in the realm of electronic mail, as finding the email address for a person. This was perhaps the most obvious application of a directory. The role of a directory has expanded over the last years to encompass many more functions than simply mapping a name to an email address. There is a strong need among Telia and other telecommunication operators to be able to integrate directories and network components to enhance the standard in their networks according to security, quality of service (QoS) and reservation of resources. There is also a strong interest of exchanging information for making new services possible.

Several big vendors, with Microsoft in the lead have since 1997, tried to make a useful global directory structure starting from X.500. They have developed a simplified access protocol named lightweight directory access protocol (LDAP) and are now trying to replace X.500's directory structure with something called directory enabled networks (DEN).

Telia ProSoft wanted to investigate were the Distributed Management Task Force (DMTF) and the Internet Engineering Task Force (IETF), (Appendix B) stands in the standardisation progress and the result is this thesis.

1.2. Goal

The goal of the project is to give an overview of the DEN-initiative and its core access protocol LDAP, and to investigate if DEN could be a solution for future services, like pay-per view. The thesis shall also result in a proposal for how Telia should position itself with respect to the initiative.

1.3. Outline

Chapters 2 to 6 serve as overview of the background to why the DEN initiative was announced and its functions. It ends with a summary.

Chapters 7 to10 describe the problem part where requirements on new services and missing functionalities in Telia's net are defined and solved based on the DEN-initiative.

Chapter 11 contains the conclusions where I give my personal view of the forthcoming work with DEN inside and outside Telia.

Appendix:
Appendix A contains useful acronyms and abbreviations. Appendix B presents the different organizations that are, or have been working with the development of DEN and LDAP.

2. X.500

A directory is a special purpose database that contains information about the nodes, or devices, attached to an enterprise network. Directories offer a potentially powerful tool in helping to simplify and automate many of the complex tasks involved in managing a large corporate network. Directory services are optimized for storing information that is frequently read, but are weak at managing data that are constantly changing [10]. In 1988 the International Telecommunication Union-Telecommunication Standardisation Sector (ITU-T) released a specification of how information can be stored and accessed in a global directory. It is called X.500 [9]. An updated version came in 1993. The focus is on the communication between a server and a client and the structure of that information. X.500 was meant to serve as a universal, standards-based directory service, but it proved overly complicated and ran only on high-powered Unix machines. This resulted in X.500 never gaining any market shares.

2.1. Components of the X.500 directory service

The standards contain the following components, which any global directory service is required to support:

- **The hierarchical namespace**, which determines how information is referenced and organized.
- **The information model**, which describes the format and structure, called the schema, of information maintained in the directory.
- **The functional model**, which specifies the directory access protocol and specific operations in the directory (e.g. read, write and authenticate).
- **The distributed operation model**, which determines how data is distributed and the operations that must be performed to synchronize and maintain the global directory across thousands of servers.

2.1.1. Data model

Figure 2.1 shows a possible directory information tree for an X.500 directory service.

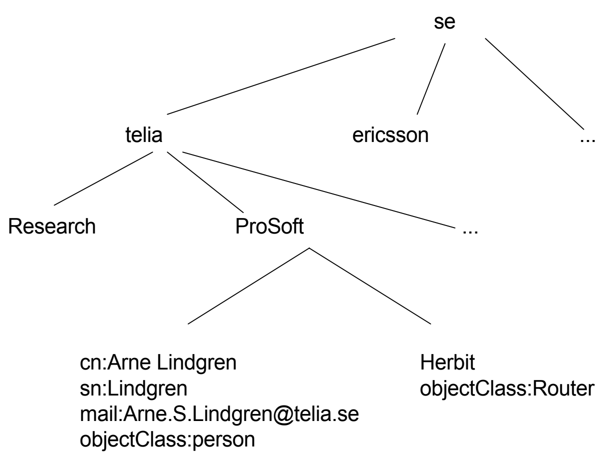

Figure 2.1 An X.500 directory information tree, DIT

The tree is made up from entries. Each element in the directory information tree (DIT) has a globally unique name, called distinguished name (DN). For example Arne Lindgren's DN would be "telia; ProSoft; Arne Lindgren". DN provides access to a unique object in the global directory. The DN written as strings of dotted decimal numbers are called Object Identifiers (OID). For example, all the user attribute types defined by the X.500 standards begin with 2.5.4.

Each object is represented by an entry in the DIT and contains a set of attributes. The attributes hold the information stored for each object. An example of an attribute is "mail." Each entry must have an objectClass attribute. The objectClass identifies the type of entry (e.g. person, switch, router, etc.) and determines which subsequent attributes are required and which are optional. The collection of attribute type definitions, objectClass definitions and other information is known as the object's schema.

2.1.2. Functional model

The X.500 directory is maintained by a set of directory system agents, DSAs. Each DSA holds information for only a part of the complete directory information base, DIB. A given DSA can either respond or forward the user requests (Figure 2.2). This is what ties together the X.500 distributed directory service.

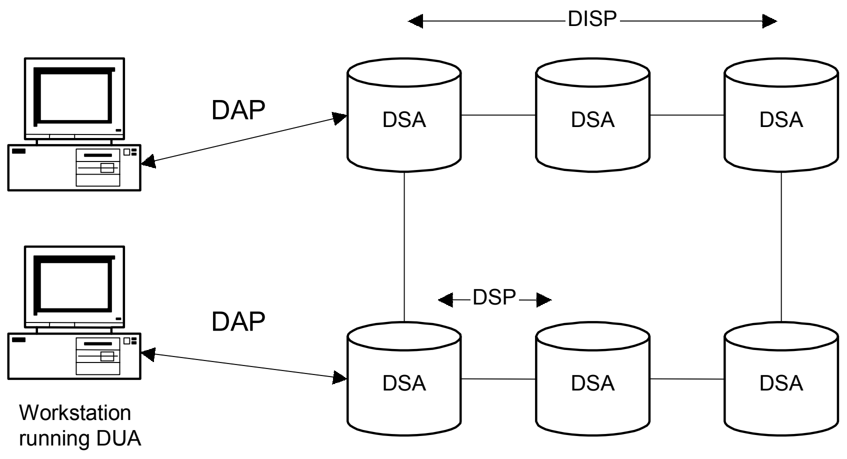

Figure 2.2 Functional model for an X.500 directory

The directory user agents, DUAs, are standalone programs within an application that provide the interface between the desktop and the local DSA. A DUA communicates with a DSA via the directory access protocol, DAP. Between DSAs, a second protocol, directory service protocol, DSP, is used. Information that is maintained in one DSA can be shadowed (i.e. copied) to other DSAs with a third protocol, called directory information shadowing protocol, DISP.

3. The DEN initiative

At the Microsoft Professional Developers Conference in September 1997, Cisco Systems Inc. and Microsoft Corp. announced an industrywide initiative to integrate directory services and networks. This enables the development of network applications that will work with offerings from a variety of network and directory vendors. This draft was the first to integrate user profiles, applications and network services. This was achieved by providing a standard schema for storing network state and an extensible information model for exposing network information. **DEN** uses some base classes from X.500 and is a complement to the DMTF-sponsored CIM specification that uses LDAP as core access protocol (Chapter 4).

3.1. Common Information Model, CIM

CIM is a common data model of an implementation neutral schema for describing overall management information in a network/enterprise environment. CIM is comprised of a specification and a schema. The specification defines the details for integration with other management models (i.e. SNMP's MIBs or the DMTF's MIFs) while the schema provides the actual model descriptions [5].

3.2. DEN Base Schema

The schema of a directory defines the set of objects that can be created in that directory and the set of attributes that can be used to describe those objects. Figure 3.1 shows the functional structure of the DEN base classes, with key classes defined by CIM, X.500 and DEN listed separately.

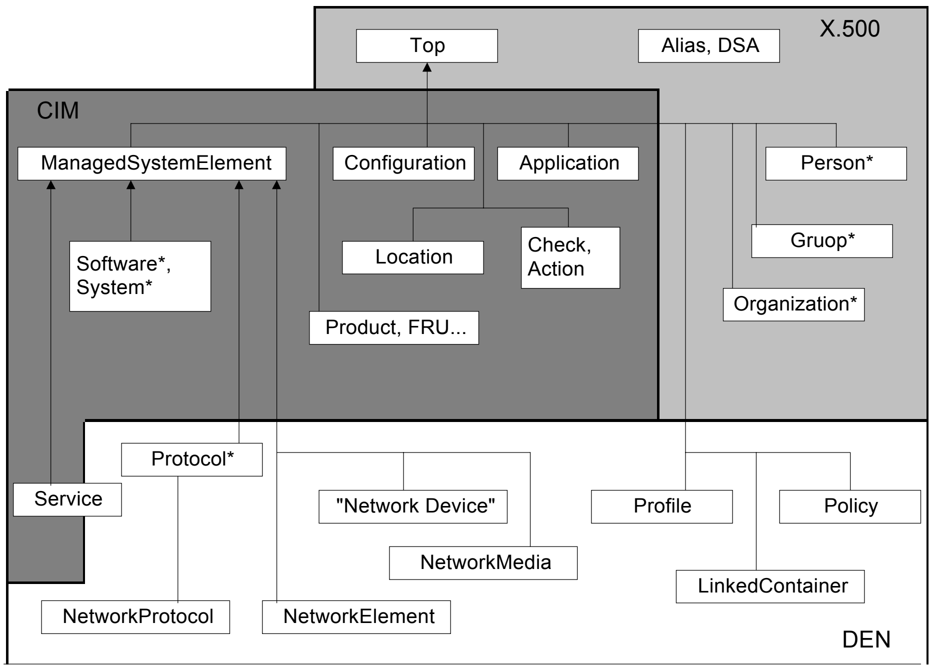

Figure 3.1 Functional structure of the DEN base classes

Note that "Network Device" is not an actual class, but rather the abstraction of changes made to existing CIM classes to realize the physical charateristics of network devices. In addition, the classes marked with an * represent a set of classes that correspond to that general function or service.

Following is just a list of short descriptions of the base classes. Interested readers should refer to [11] for more details.

3.2.1. Overview of Base Classes derived from X.500

- **Top**: Root of the directory tree.
- **Person**: Generic concept of a person, an employee, a person with a residence. DEN uses it as a client to bind network services to, or as owner/administrator of a device or a service.
- **Group**: Providing grouping constructs for users as well as devices.
- **Organization**: Business entity to which devices and services belong.
- **Application**: X.500 defines the ApplicationProcess and ApplicationEntity classes. DEN adds information so that these classes can be associated with network elements and services
- **Alias, DSA**: Necessary entities for proper directory operation

3.2.2. Overview of Base Classes derived from CIM

- **Product, FRU, etc**: A collection of classes that represents a product and replaceable parts of a product.

- **ManagedSystemElement**: Base class for any system or system component that should be managed.
- **Configuration**: Modeling the configuration and provisioning of network elements and services.
- **Service**: Definition, management and delivery of network services.
- **Software**: General notation of software.
- **System**: Realizing the concept of a logical network element.
- **Location**: Specifying the address and location of a physical element.
- **Check** and **Action**: Used by DEN to augment notions of configuration and reboot of network devices.
- **Application**: CIM adds the concept that an application is used to support a particular business function.

3.2.3. New DEN Classes

Enhanced and extended concepts defined by DEN:

- **NetworkService**: Root of the network service hierarchy.
- **NetworkProtocol**: Root of all network protocol classes.
- **Enhancements to PhysicalPackage and Card**: Extensions and enhancements to include the functionality required by network devices.
- **NetworkElement**: Logical aspects of a network element.

New concepts defined by DEN:

- **Policy**: Rule instructing a network node on how to manage requests for network resources.
- **Profile**: Template of attributes and behaviors that describe an object or a set of objects.
- **NetworkMedia**: Associating the particular media of a given interface with services that are running on it.
- **LinkedContainer**: Container class implementing a forward link.

3.3. Important Classes

Among the DEN base classes there are a few classes that are especially interesting.

- ManagedSystemElement
- Profile
- Policy

Below I will describe these classes more specifically. The class Policy will be described in QoS policies, chapter 5.6.

3.3.1. ManagedSystemElement

The ManagedSystemElement class is the CIM base class for **PhysicalElement** and **LogicalElement** classes. It provides the fundamental notion of manageability of a system or system component.

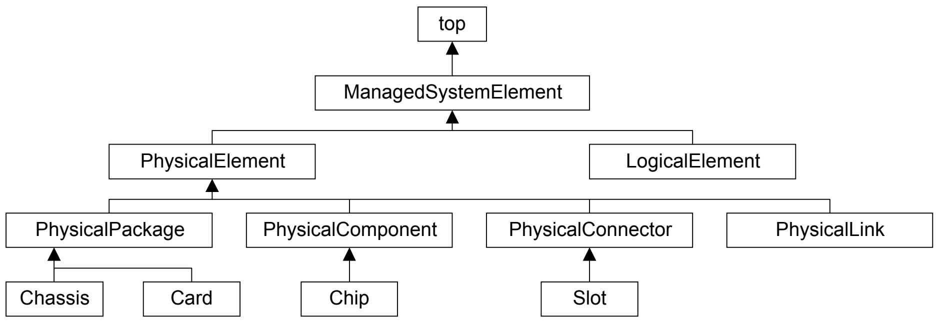

Figure 3.2 PhysicalElement Class Hierarchy

PhysicalElement (Figure 3.2):

- **PhysicalPackage**: represents physical elements that contain or host other physical elements. Subclasses are:
  Chassis: represents a PhysicalElement that can enclose other PhysicalElements and provides definable functionality as an aggregate, such as a desktop or a network device.
  Card: represents a type of physical container that is used to provide specific functionality (such as added memory) or a hosting device that can accept additional PhysicalElements.
- **PhysicalComponent**: represents any low-level physical component within a PhysicalPackage. Subclass:
  Chip: represents any type of integrated circuit hardware.
- **PhysicalConnector**: represents any PhysicalElement that is used to connect to other PhysicalElements. Subclass:
  Slot: represents ability to insert a card or adapter board.
- **PhysicalLink**: represents the cabling of PhysicalElements.

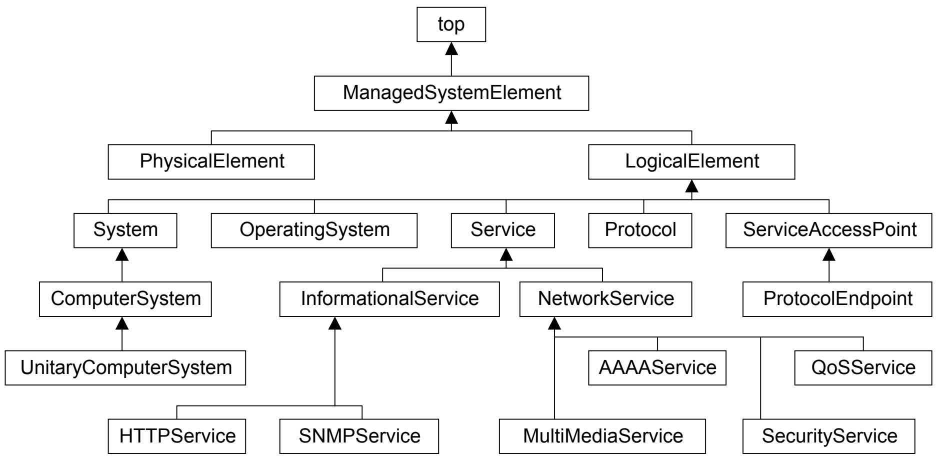

Figure 3.3 LogicalElement Class hierarchy

LogicalElement (Figure 3.3):

- **System**: represents a LogicalElement that aggregates a set of ManagedSystemElements. The aggregation operates as a functional whole. Subclass:
  ComputerSystem: represents a collection of components that have computing capabilities and serves to aggregate file systems, memory, and other elements needed for computation. Subclass:
    UnitaryComputerSystem: represents a single processing node computer system. This is the superclass for one type of network device.
- **OperatingSystem**: represents software and/or firmware that implements and manages the general processing logic performed.
- **Service**: contains the information necessary to configure represent, and manage the functionality provided by a LogicalDevice or SoftwareFeature. Subclasses:
  InformationalService: a base class for generalized services, such as HTTP and SNMP. Note the existence of subclasses, such as **HTTPService** and **SNMPService**.
  NetworkService: an abstract base class for objects that provide network services of a non-informational nature to clients. Non-informational means that it is comprised of a set of attributes and behaviours which describe a function or a set of functions that can be invoked to reside on the network. This enables application developers to match services to users, groups, and other. Note the existence of subclasses, such as **MultiMediaService**, **AAAAService**, **SecurityService** and **QoSService**.
- **Protocol**: represents the general characteristics of a given protocol.
- **ServiceAccessPoint**: represents the ability to manage the access of a service. Subclass:
  ProtocolEndpoint: represents a communication point from which information may be sent or received.

3.3.2. Profile

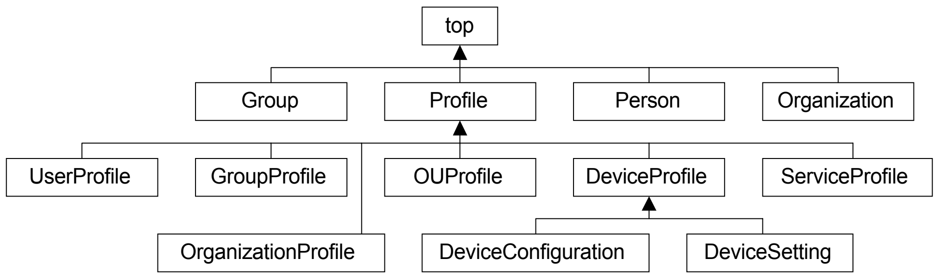

Figure 3.4 Profile Class Hierarchy

Profile (Figure 3.4):

- **UserProfile**: an abstract class for grouping together different profiles that effect individual users. For example, a UserProfile could define how a user connects to the network and what services are available to that user once the user is connected.
- **GroupProfile**: an abstract class for grouping together different profiles that effect groups of users.
- **Organizationprofile**: an abstract class for grouping together different profiles that effects an entire organization.
- **OUProfile**: an abstract class for grouping together different profiles that effects users in an OrganizationalUnit.
- **DeviceProfile**: an abstract class for grouping together different profiles that can be applied to a network device. It has two subclasses:
  DeviceConfiguration: represents the configuration of a network element as directed by a Profile.
  DeviceSetting: represents configuration-related and operational parameters for one or more network devices.
- **ServiceProfile**: an abstract class for grouping together different profiles that effect how a particular service is provided to users or groups of users.

3.4. Benefits of DEN

Directory enabled networks support a new way of performing network management. The two key areas are **scalable device configuration management** and **policy powered networking**. Policy powered networking is more complex and is described more specifically in chapter 5. The benefit with DEN is obvious as the size of an organization's network expands. When the network manager would like to add a router to the network he does not need to configure it separately. Instead the new router simply examines the directory for its pre-stored configuration parameters to go online. Device configuration management of today requires the administrator to make a Telnet connection to each device and perform a box-by-box configuration, a very expensive operation. With DEN device configuration management the network administrator only has to store configuration parameters for each network node in a common directory service (Figure 3.5). Other problems with currently existing directory services are that there are too many different types. Also much of the information is duplicated across multiple directory services (e.g. an employee's name, telephone number and email).

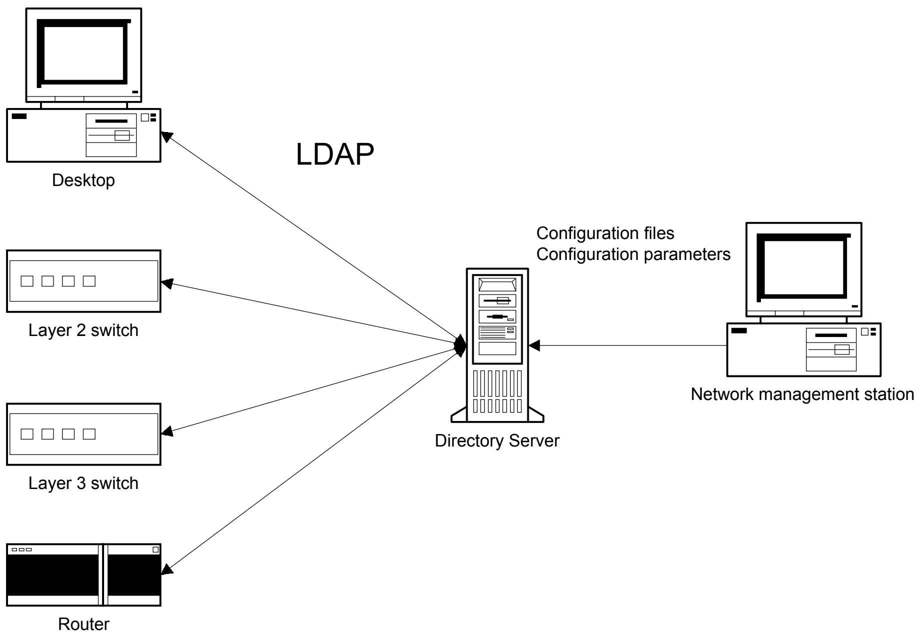

Figure 3.5 Centralized Device Configuration Management

Each of these individual directory services has its own specific access protocol, storage formats and naming conventions (e.g. a user's name can be different in each and every directory). These differences make it difficult for one directory to communicate and share information with another. The ultimate goal of a directory enabled network is to establish a single directory structure that is accessible to the entire network.

3.4.1. Economy

The investment in directory servers and policy servers implies that one has to buy switches and routers with QoS features (i.e. those capable of acting as policy enforcement points), which cost more than plain "commodity" switches and routers without this functionality. In the long run, however these higher investments are profitable, because the costs of managing the network are reduced and resources can be used optimally. In fact a single stand-alone policy server may cost less than the cost of including an embedded policy server in each switch or router.

Table 3.1 Summary of the benefits

End users	Single network log-on. Personalized network services.
Service providers	Rapidly create, provision and deploy advanced networking services on a per-user basis. Centralized management of network resources.
Enterprise customers	Protect mission critical traffic. Simplify and enhance network management and provisioning.
Application developers	Easy access to advanced network services. Develop network aware applications using standard development interfaces and tools

3.5. History

In 1988 ISO/ITU-T released a specification of how information can be stored and accessed in a global directory. It is called X.500 (see chapter 2), and an updated version came 1993. X.500 proved overly complicated and ran only on high-powered Unix machines. The Internet Engineering Task Force, IETF (Appendix B) announced that they were not going to address the problem of creating schema standards, so in **May 1997**, Microsoft and Cisco formed the DEN-initiative aiming to integrate directory services and networks. In **September 1997**, the two founding partners announced that more than 20 network equipment vendors had agreed to support the initiative. **February 1998** the initiative moved from the ad hoc working group (AHWG) under the desktop management task force, DMTF (today the D stands for Distributed). The intent was to ensure that the work performed by DEN should be standardized. The DEN specification became a part of the DMTF's common information model (CIM) standard for enterprise management.

4. The Lightweight Directory Access Protocol, LDAP

The X.500 server software is complex and thus difficult to work with. Further more it is closely related to the OSI protocol stack. Because of this a more TCP/IP-centric solution was needed [6].

4.1. Benefits of LDAP

The Lightweight directory access protocol, LDAP, was developed in the early 1990s at the University of Michigan as a lean alternative to X.500's heavyweight DAP. Over time it has grown to be capable of supporting complete directory service.

Application developers needed earlier to choose a directory system to support or even just create a private directory that would only be used by their application to store users, configuration information, passwords, etc. Now they can use a standard protocol, LDAP, to access a directory in a standard manner (Figure 4.1). Currently the IETF has two working groups working with modifications and updates. The LDAP Directory Update, LDUP working group is hammering out a standard replication service that will make directory replication easier. The LDAP Extensions working group is charged with creating standard extensions to LDAP and the LDAP APIs.

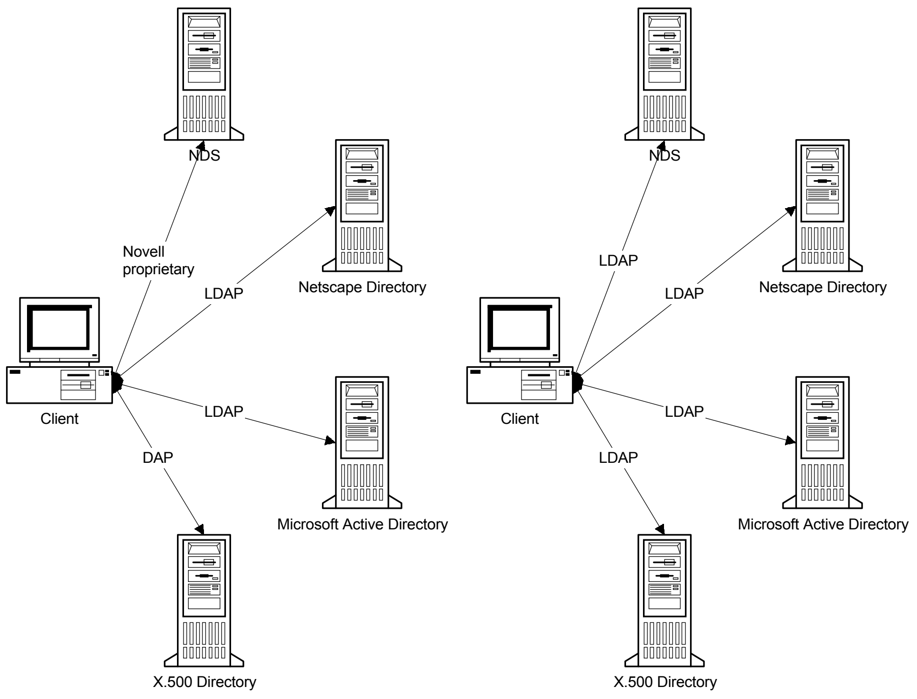

Figure 4.1 LDAP: Standard based directory access protocol

An LDAP directory is structured as a simple tree hierarchy (Figure 2.1), which conforms to the LDAP schema and naming models. LDAP performs two critical jobs. First of all, it serves as an authentication database. Second, once the identity of a user has been established it controls access to resources, applications and services. The current version is LDAPv3 [19]. LDAPv3 has a referral capability so one directory server can forward a client's query to another. For example, suppose that an LDAP client issues a search operation to Server1 (Figure 4.2) with a search base of ou=ProSoft, dc=telia, dc=se, a scope of subtree, and a filter of (:objectclass=person)(sn=Lindgren). In other words, the client wishes to find all persons within dc=telia, dc=se whose surname is Lindgren.

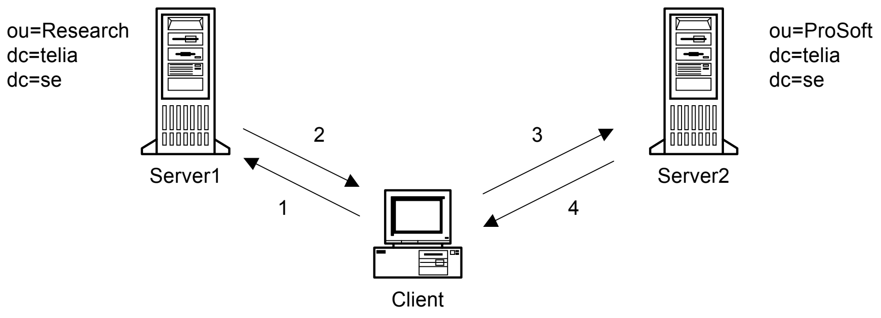

Figure 4.2 Distributed searching with referrals.

To service this search request, the distributed directory first uses name resolution to find the base object of the search. It then uses any knowledge references to refer the client to all the servers it needs to obtain the entire set of matching entries.

4.2. Specification

The current specification comprises eight features and functions for defining or performing directory-related tasks like storage and retrieval [7] [8].

- **Information model**
  Organized according to collections of attributes and values, known as entries. This model defines which data that can be stored and how that data behaves.
- **LDAP schema**
  Defines the actual data elements that can be stored in a particular server and how they relate to real world objects. Values and attributes representing countries, organizations, people, etc. are defined in the standard, but individual servers can define new schema elements as well.
- **Naming model**
  Specifies how information is organized and referenced. LDAP names are hierarchical, individual names are composed of attributes and values from the corresponding entry (Figure 2.1).
- **Security model**
  Spells out how information is secured against unauthorized access. Extensible authentication allows clients and servers to prove their identity to one another.
- **LDAP functional model**
  Determines how clients access and update information in an LDAP directory. LDAP offers nine basic functional operations: add, delete, modify, bind, unbind, search, compare, modify DN and abandon.
- **LDAP protocol**
  Defines how all of the preceding models and functions map onto TCP/IP. The protocol specifies the interaction between clients and servers and determines how LDAP requests and responses are "formed" (i.e. how the bits look on the wire).
- **Application program interface, API**
  Details how software programs access the directory, supplying a standard set of function calls and definitions [15].
- **LDAP data interchange format, LDIF**
  Provides a simple text format for representing entries and changes to those entries. LDIF and the LDAP API, along with scripting tools like Perl, make it easy to write automated tools for updating directories.

4.3. Replication requirements

As mentioned earlier the LDUP working group is developing a replication protocol [20]. The ability to distribute directory information throughout the network provides a two-fold benefit to the network:

- Increasing reliability of the directory through fault tolerance.
- Bringing directory content closer to the clients using the data.

The requirements for replication are critical to the successful deployment and acceptance of LDAP in the market place. The major objective is to provide an interoperable LDAP directory synchronization protocol which is simple, highly efficient and flexible enough to support both multi-master and master-slave replication operations to meet the needs of both the internet and enterprise environments.

To highlight the replication requirements I have chosen two examples from [20].

4.3.1. Extranet example

A company has a trading partner to whom it wishes to provide directory information. This information may be as simple as a corporate telephone directory, or as complex as an extranet work flow application. For performance reasons the company may wish to have a replica of its directory within the Partner Company, rather than simply exposed to everyone.

The requirements, which follow from this scenario, are:

- One-way replication.
- Authentication of clients.
- Common access control and access control identification.
- Secure transmission of updates.
- Selective attribute replication (fractional replication), so that only partial entries can be replicated.

4.3.2. Enterprise directory replication mesh

A corporation builds a mesh of directory servers within the enterprise utilizing LDAP servers from various vendors. Five servers are holding the same area of replication. The predetermined replication agreement(s) for the enterprise mesh are under a single management, and the security domain allows a single predetermined replication agreement to manage the five servers replication.

The requirements, which follow from this scenario, are:

- Predefined replication agreements which manage more than a single area of replication that is held on numerous servers.
- Common support of replication management knowledge across vendor implementation.
- Rescheduling and continuation of a replication cycle when one server in a replica ring is busy and/or unavailable.

4.4. Defining one’s directory

When one is defining a directory it is important to understand the requirements that will be placed on it. The directory shall for example support directory enabled applications that have certain requirements concerning the data contained in the directory.

An individual entry expressed in LDIF format consists of two parts: a distinguished name and a list of attribute values (Listing 4.1). The DN, which must be the first line of the entry, is composed of a string.

Listing 4.1 A typical LDIF file.

```
dn: uid=arli, ou=people, dc=Telia, dc=se
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Arne Lindgren
givenname: Arne
sn: Lindgren
uid:arli
mail: Arne.S.Lindgren@telia.se
telephoneNumber: +46 (0)8 713 15 77
```

In Listing 4.1 you can see that the organization unit (ou) people, inherent functionality from top, person and organizationalPerson. There exists user-friendly software, so that one does not have to make LDIF files for every entry in the directory.

In Figure 3.1 you can see the functional structure of the DEN base classes: top-person. This is a proposed standard that can not be changed, but underneath this you are free to put in own classes. When designing a directory tree structure, one important rule is to keep it flat to avoid complexity.

5. Directory Support for Policy-Powered Networking

Policy-powered networking is much more complex than device configuration management because directories are merely databases, and are not designed to gather information from a number of different sources before making a policy decision [17]. A **policy** is a rule that instructs a network node on how to manage requests for network resources.

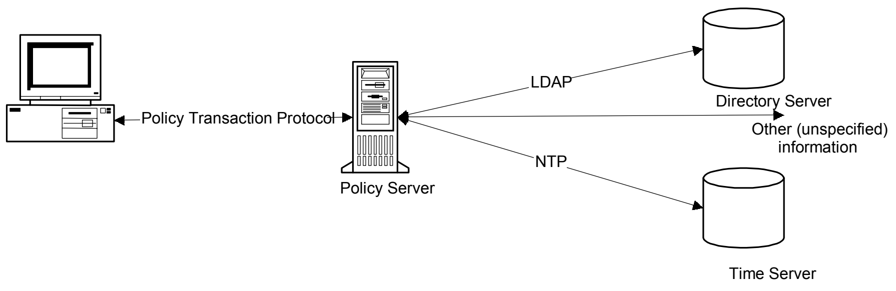

Figure 5.1 Policy-Powered Networking

5.1. Roles in a Policy-Powered Network, PPN

The four roles required in a policy-powered networking decision include (Figure 5.2) [16].

- **Information storage**
  Information storage is concerned with the back-end data stores, the replication of data throughout the network, and the schema of the directory information.
- **Policy Requester**
  A policy request is triggered when a policy client observes a demand to access network resources. The trigger could be the launch of an application, a dial-in to a remote access server port, traffic across an Ethernet interface, or the start of a routed flow.
- **Policy Interpreter**
  Policy interpretation is the role performed by the network device that manages state and evaluates the particulars of the resource request. If the enforcing device performs the "police" function in the network, the interpreter is the judge that weighs the rule of law and the specific circumstances of the case.
- **Policy Enforcer**
  Enforcement is the role performed by a network node to ensure that the policy lease is realized. Enforcement of policy decisions is

carried out by the specific hardware/software features residing in the policy client, such as packet filtering, bandwidth reservation, traffic prioritization, multiple forwarding queues, etc.

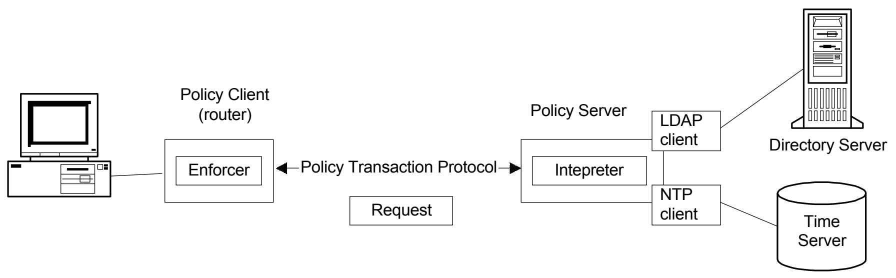

Figure 5.2 Basic model for policy-powered networking

For effective management to take place, there are a number of requirements relevant to the current discussion which can not be ignored. Below are some of the areas which any policy management paradigm must address to successfully integrate into a currently deployed Internet management environment.

- **Multiple management stations**
  In modern networks service and performance interruptions must be kept to a minimum. This applies to the management components are often on a hot stand by and in full sync with their primary counterparts to avoid extra delay for connection re-establishment in case of failure.
- **Multiple users in multiple roles**
  From the previous item, it should be clear that multiple users must have access to a single system - often concurrently. What is also true is that not all management personnel are given the same privileges with respect to access to information.
- **Connection-oriented and connection-less Management**
  Connection-less polling for short term information is far less costly than maintaining many (potentially thousands) of connections for the collection of this type of data. Connection oriented management systems have always been problematic with regard to scale.
- **Hierarchically managed networks/overlapping domains**
  Hierarchy is not always exactly straight up and down. It is often the case that policy does not have clean boundaries and systems must to some degree have multiple masters.
- **Policy and instance specific information - required integration**
  The idea of saving data transfer to the policy enforcement point (PEP) by avoiding the granularity of instance level data is a helpful concept. There is often some configuration information that is required on a per instance basis. A good example would be the IP address of an interface. It is essential that these two types of

information, general policy rules/roles and specific instance related information be well connected. Otherwise detecting and correcting configuration errors will be more difficult.

- **Compatibility with simple network management protocol (SNMP) management data**
  Similarly, data about the utilization of various resources in the network is necessarily collected via SNMP based mechanisms. It will be impossible to make good policy decisions without this information. The conversion of this (SNMP) data to another format could be costly.
- **Multiple configurations and switching configurations**
  Configuration management systems in general have to support the notion of multiple configurations. There are usually at least two configurations stored in networking devices: The currently active configuration and a configuration, which is stored in stable storage and reloaded upon re-initialization.
- **Evolution of configuration management data**
  Configuration management data models are not fixed for all time and are subject to evolution like any other management data model. It is therefore necessary to anticipate changes in the configuration data model and to provide mechanisms that can deal with changes effectively without causing interoperability problems or having to replace/update large amounts of fielded networking devices.

5.2. Policy Transaction Protocols

A number of working groups in IETF needed support for the configuration of policy in the network. SNMP was evaluated by some working groups and found lacking in a number of areas. As a result, new work was undertaken which led to Common Open Policy Service protocol (COPS) and DIAMETER [2] among others. These solutions focused on solving the local problems of policy management. A local solution like COPS can, if properly specified and implemented, be more efficient than general ones. But some mean that the advantage of effective management through the use of a single integrated management framework (e.g. an improved SNMP) far outweigh the gains of locally optimized solutions.

After following the work done by the IETF, I must come to the conclusion that COPS or perhaps an enhanced SNMP [14] will become the future standard policy-transaction-protocol. Below are descriptions of these two policy-transaction-protocols (i.e. the protocol that carries the policy request/response between the policy client and the policy server).

5.2.1. Simple Network Management Protocol, SNMP

SNMP is a simple request-reply protocol between an SNMP manager and an SNMP agent [4]. The Management Information Base (MIB) defines the variables that are maintained by the agent for the manager to query or set. SNMP defines only five types of messages that are exchanged between the manager and agent.

- **get-request** **manager→agent**
  Fetches the value of one or more variables.
- **get-next-request** **manager→agent**
  Fetches the next variable after one or more specified variables.
- **set-request** **manager→agent**
  Sets the value of one or more variables.
- **get-response** **agent→manager**
  Returns the value of one or more variables. This is the message returned by the agent to the manager in response to the get-request, get-next-request and set-request operators.
- **trap** **agent→manager**
  A notification to the manager when something happens on the agent.

SNMP uses UDP as transport protocol, the manager sends its requests to UDP port 161 and the agent sends traps to UDP port 162. By using two different port numbers, a single system can easily run both a manager and an agent. SNMP's typical use of UDP is an appropriate one because research and field experience have shown that UDP, with an appropriate application layer retransmission strategy is more robust, and provides a lower mean time to conduct a transaction, than TCP does with its one-size-fits-all retransmission algorithm.

5.2.2. Common Open Policy Service (COPS) protocol

The COPS protocol is an Internet-Draft specifying an admission control protocol that is currently under development by the RSVP admission policy (RAP) working group [1]. COPS is a simple query-and-response protocol (Figure 5.3) for exchanging policy information between a policy server (policy decision point, PDP) and its clients (policy enforcement points, PEPs), which are in a sense the "judge" and the "policeman". The PEP is an enforcer, and the PDP makes decisions based on the policies it retrieves from the policy repository (e.g. LDAP-directory) and perhaps other locations such as authentication servers.

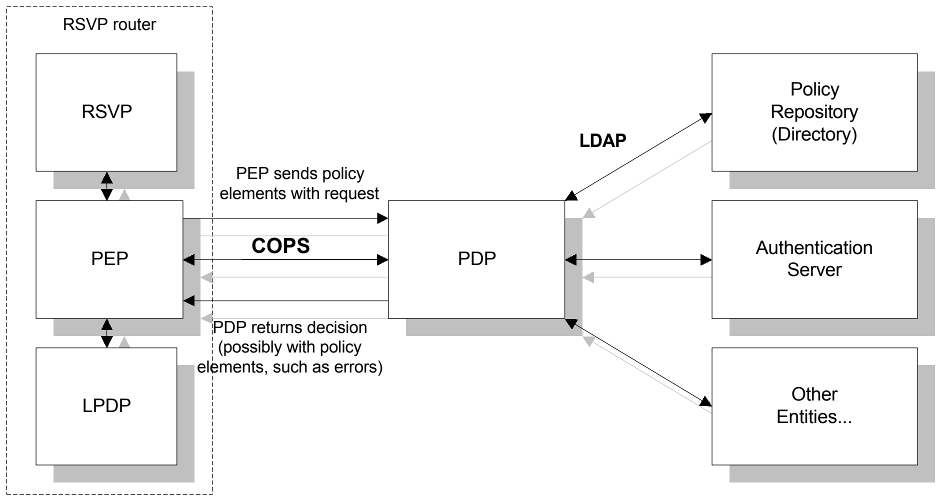
RSVP router
RSVP
PEP
LPDP
PEP sends policy elements with request
COPS
PDP returns decision (possibly with policy elements, such as errors)
PDP
LDAP
Policy Repository (Directory)
Authentication Server
Other Entities...

Figure 5.3 A Policy Framework utilizing COPS

The PEP may also have the capability to make a local policy decision via its Local Policy Decision Point (LPDP), however, the PDP remains the authoritative decision point all times. This means that the relevant local decision information must be relayed to the PDP. That is, the PDP must be granted access to all relevant information to make a final decision. To facilitate this functionality, the PEP must send its local decision information to the remote PDP via an LPDP decision object. The PEP must then abide by the PDP's decision, as it is absolute. As I mentioned above, SNMP uses MIBs for defining the variables that are maintained by the agent for the manager to query or set. COPS uses PIBs (Policy Information Base), which are intentionally similar to MIBs.

It is possible for a single PEP to have open connections to multiple PDPs. This is the case when there are physically different PDPs supporting different client types.

The following ten messages are the basic exchange between a PEP and a remote PDP.

- **Request (REQ)** **PEP→PDP**
  The PEP establishes a request state client handle for which the remote PDP may maintain state. The remote PDP then uses this handle to refer to the exchanged information and decisions communicated over the TCP connection to a particular PEP for a given client-type.
- **Decision (DEC)** **PDP→PEP**
  The PDP responds to the REQ with a DEC message that includes the associated client handle and one or more decision objects. If there was a protocol error an error object is returned instead. All requests must have a corresponding decision.

- **Report State (RPT) PEP→PDP**
  The RPT message is used by the PEP to communicate to the PDP its success or failure in carrying out the PDP's decision, or to report an accounting related change in state.
- **Delete Request State (DRQ) PEP→PDP**
  When sent from the PEP this message indicates to the remote PDP that the state identified by the client handle is no longer available/relevant.
- **Synchronize State Req (SSQ) PDP→PEP**
  This message indicates that the remote PDP wishes the client to re-send its state. If the PEP does not recognize the requested handle, it must immediately send a DRQ message.
- **Client Open (OPN) PEP→PDP**
  The Client-Open message can be used by the PEP to specify to the PDP which client types the PEP can support, the last PDP to which the PEP connected for the given client-type, and/or client specific feature negotiation. A Client-Open message can be sent to the PDP at any time and multiple Client-Open messages for the same client-type are allowed (in case of global state changes).
- **Client-Accept (CAT) PDP→PEP**
  The Client-Accept message is used to positively respond to the Client-Open message. This message will return to the PEP a timer object indicating the maximum time interval between keep-alive messages. Optionally, a timer specifying the minimum allowed interval between accounting report messages may be included when applicable.
- **Client-Close (CC) PEP→PDP, PDP→PEP**
  The Client-Close message can be issued by either the PDP or PEP to notify the other that a particular type of client is no longer being supported.
- **Keep-Alive (KA) PEP→PDP, PDP→PEP**
  The keep-alive message must be transmitted by the PEP within the period defined by the minimum of all KA Timer values specified in all received CAT messages for the connection. A KA message must be generated randomly between 1/4 and 3/4 of this minimum KA timer interval. When the PDP receives a keep-alive message from a PEP, it must echo a keep-alive back to the PEP. This message provides validation for each side that the connection is still functioning even when there is no other messaging.
- **Synchronize State Complete (SSC) PEP→PDP**
  The Synchronize State Complete is sent by the PEP to the PDP after the PDP sends a synchronize state request to the PEP and the PEP has finished synchronization. This way the PDP will know when all the old client state has been successfully re-requested and, thus, the PEP and PDP are completely synchronized.

COPS policy services can also be supported in Resource ReSerVation Protocol (RSVP) environments [1]. When a RSVP message arrives at the router (or a RSVP related event requires a policy decision), the RSVP module is expected to hand off the request (corresponding to the event or message) to its PEP module. The PEP will use the PDP (and LPDP) to obtain the policy decision and communicate it back to the RSVP module. That means that the PEP and PDP share RSVP state, and the PDP is assumed to implement the same RSVP functional specification as the PEP. In he case where a PDP detects the absence of objects required by RSVP it should return an <Error> in the Decision message indicating "Mandatory client-specific info missing". If, on the other hand, the PDP detects the absence of optional RSVP objects that are needed to approve the Request against current policies, the PDP should return a negative Decision.

COPS uses TCP as its transport protocol for reliable exchange of messages between policy clients and a server.

5.3. A comparison of COPS and SNMP

The use of COPS as the protocol by which a router (the Policy Enforcement Point, or PEP) queries a central Policy Server (the Policy Decision Point, or PDP) on receipt of RSVP messages, has been successfully tested in interoperability tests involving multiple vendors. The specifications are currently under review by the IESG for progression to proposed Internet standard status. However before it reaches this status, it has been requested that a comparison be undertaken as to why COPS is better suited to this than a (modified is necessary) version of SNMP [13]. Here it is important to understand that SNMP are being used today and nothing that could be replaced by COPS. COPS can never take SNMP's place but could be used as a complement for Policy services.

An application, which configures a device using SNMP, can never be sure that the configuration it set several minutes/hours/days ago is still in effect, because it is possible that some other management application (or human) might have modified the configuration more recently. So, a prudent management application will periodically recheck that the configuration is unchanged. This need for re-checking is not specific to SNMP, and is similarly required when configuration is done via CLI. In contrast, exclusive access by a single PDP avoids the uncertainty and consequent need for re-checking that the device's configuration has not been tampered with.

Many of the SNMP protocol's design choices were based on the need for its use in debugging network problems. In contrast, the design choices for the COPS protocol have always assumed that the network is up and running. Thus SNMP runs over UDP, whereas COPS runs over TCP. The use of TCP allows COPS to enjoy the use of large and therefore more efficient messages (up to 64KB for each COPS object). In contrast, SNMP over UDP requires an implementation to support a minimum of 484 bytes as its largest message size. Both protocols are "transaction" protocols, meaning that all policies contained in one message must be successfully installed, or none of them are. This gives COPS an advantage, while it allows much larger messages implementing much more policy to be created, deleted and/or modified in one transaction, than would be possible with SNMP.

Another consequence of SNMP's small message size is the potential that a complete row of a MIB table might not fit into one message. COPS's use of larger messages ensures that a whole row will fit into a single message, this makes it no longer necessary to include the name of each and every columnar object instance being accessed. Instead only the name of the row needs to be specified. In other words a COPS message includes an OID naming the row, plus a vector of values, one for each value in the row. In contrast a SNMP message contains a list, consisting of the name and value of each referenced columnar object in the row. For example, if a row consisted of 10 integers, with OIDs of length 15 bytes and integer values of 10 integers, it would use up 210 bytes of an SNMP message, but only 57 bytes in COPS.

For the purpose of obtaining policy, it is better for the device to locate a PDP, rather than for a PDP to try to find devices that need their policies. In contrast, SNMP is designed such that multiple stations can monitor a device as and when they need to, and hence it is each management station that must initiate SNMP communication. It has been suggested to get initiation by the PEP using SNMP by having devices (e.g., a router) contain a command generator. With COPS, initiation by the PEP is achieved without the PEP needing to issue any search queries. Instead, on establishment of the TCP connection, the PEP sends an indication of its capabilities and status to the PDP. It then asks the PDP to send it the relevant policies.

COPS and PIBs could never replace the use of SNMP and MIBs, which is still needed for monitoring and setting local configuration. One might ask, why not SNMP could be modified to have the same functionalities as are described for COPS? It could. However it would no longer be the same protocol, and this new protocol would need to be an addition for the existing protocol which currently provides the network monitoring functions of the Internet.

5.4. Redundancy and High Availability

The policy transaction process needs to offer redundancy and support for a failover capability. Typically, client/server protocols provide resilience in either a server-side or a client-side manner.

- **Server-side redundancy**
  A standby or backup server is waiting to take over if the primary server fails.
- **Client-side redundancy**
  Both primary and secondary servers are defined at the client platform. This requires client intelligence to automatically transition to the backup server if the primary server fails. Directories are generally client-side redundant.

If a policy server is unreachable, the enforcing device must behave in a manner that does not negatively impact network operation or performance. If the policy request consumes a significant amount of bandwidth (e.g. if a video-conferencing application consumes 60 % of a WAN link) the default behaviour would probably be to deny the request. Anyway, the policy system must be designed to function so that it never negatively impacts traffic by forcing it below the "best-effort" level that would normally exist if the system were not in place.

5.5. Types of Policy-Enforcement Devices

The model establishes a 2x2 matrix (Figure 5.4) consisting of active versus passive devices, and policy versus legacy devices. An active device is one that can issue queries to a policy server, while a passive device can not. A policy device is defined as one that communicate via a policy transaction protocol such as COPS while a legacy device is one that only uses its legacy protocol (e.g. SNMP or RADIUS) and is not updated to a native policy transaction protocol.

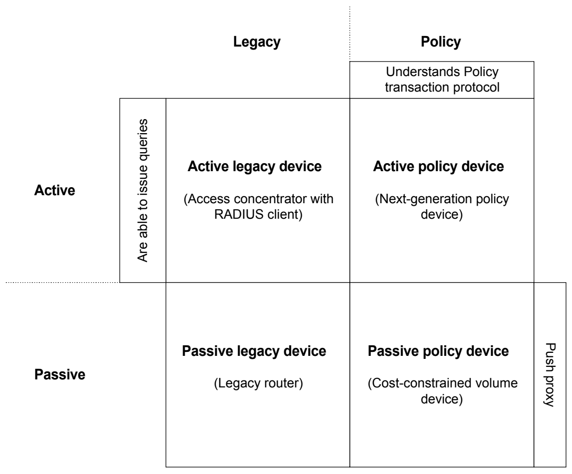
Legacy
Policy
Understands Policy transaction protocol
Active
Are able to issue queries
Active legacy device
(Access concentrator with RADIUS client)
Active policy device
(Next-generation policy device)
Passive
Passive legacy device
(Legacy router)
Passive policy device
(Cost-constrained volume device)
Push proxy

Figure 5.4 Four classes of Policy-Enforcement Devices

As seen in Figure 5.4, passive devices do not query a policy server. These devices can only be configured to enforce policy via a device-specific method (SNMP, FTP, etc.). Policy proxies facilitate the deployment of policy-powered networks by pushing the policies out to the device. They can not issue queries either. Active policy devices and passive policy devices are the only devices that have the possibility to understand a policy transaction protocol (e.g. COPS).

5.6. QoS policies

A LDAP schema that consists of a total of thirteen classes, comprised of two class hierarchies; structural classes and relationship classes, is defined in [18]. The structural classes represent policy information and control of policies. They comprise six general classes and two vendor-specific classes, all of which correspond to similarly named classes in the DMTF Common Information Model, CIM: **policy, policyGroup, policyRule, policyCondition, policyTimePeriodCondition, policyAction, vendorPolicyCondition,** and **vendorPolicyAction** (Figure 5.5).

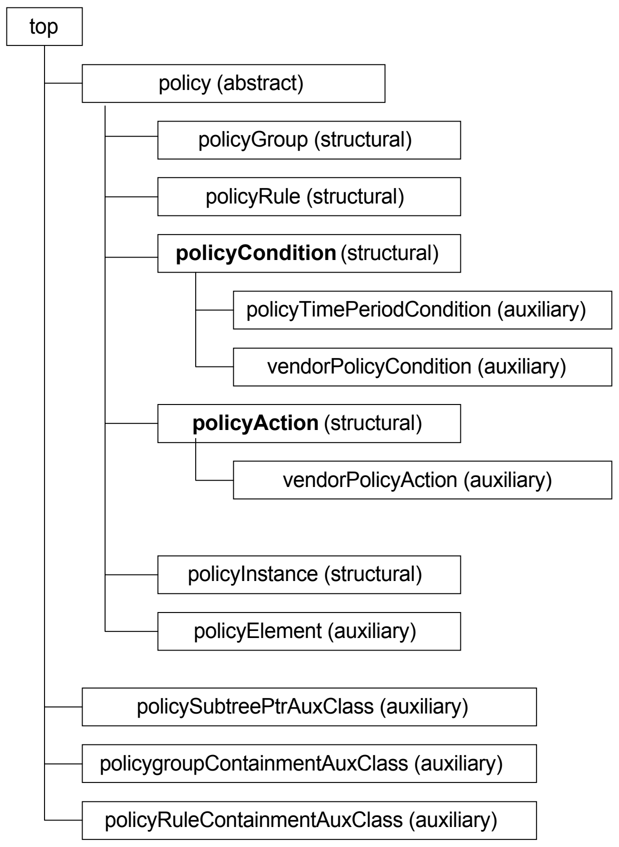
top
policy (abstract)
policyGroup (structural)
policyRule (structural)
policyCondition (structural)
policyTimePeriodCondition (auxiliary)
vendorPolicyCondition (auxiliary)
policyAction (structural)
vendorPolicyAction (auxiliary)
policyInstance (structural)
policyElement (auxiliary)
policySubtreePtrAuxClass (auxiliary)
policygroupContainmentAuxClass (auxiliary)
policyRuleContainmentAuxClass (auxiliary)

Figure 5.5 Class hierarchy in core policy schema

Notice that **policyCondition** and **policyAction** are in bold in Figure 5.5 to highlight the fact that they are extended by subclasses and auxiliary classes.

5.6.1. Policy example

This section will provide an example of the canonical use of policies, policy rules, policy conditions and policy actions (from Section 6 in [17]).

Assume that following business rules is to be implemented as a **policy**:

> *Provide the jitterFreeMPEG2 video service for authorized users between authorized points, but only at agreed-upon times*

This **rule** can loosely be translated as:

IF *user* IN *ApprovedUsers*
AND *service* IN *VideoServices*
AND *source* IN *VideoSources*
AND *destination* IN *VideoDestinations*

```
        AND time IN ApprovedTimePeriods
    THEN provide JitterFreeMPEG2
```

The policy **condition** is loosely translated as:

```
    IF the user is a member of an approved group (ApprovedUsers)
        that are authorized to have this service
    AND the service requested is one supported (VideoServicesgroup)
    AND the source of the request is approved (in the VideoSources
        group or has been authenticated)
    AND the destination is approved (in the VideoDestinations group
        or has been authenticated)
    AND the time requested is OK (in ApprovedTimePeriods)
```

Here, the policy condition types are: user, service, source, destination, and time and the policy condition elements are: ApprovedUsers, VideoServices, VideoSources, VideoDestinations, andApprovedTimePeriods, which are all instances of pre-defined groups of objects.

The policy **action** is:

```
    IF the conditions are satisfied
    THEN provide the user with video having a QoS defined by the
    JitterFreeMPEG2 service
```

Note that this policy could require "sub-polices" in order for it to be implemented. For example RSVP requests might be used to precondition the path between VideoSources and VideoDestinations.

6. Summary theory

There is no established standard to describe the structure of the directory database (i.e. how network objects and their attributes are defined and represented). A common directory schema is needed if multiple vendor applications are to share the same directory information. The good news is that most network equipment vendors have announced their support for the forthcoming DEN standard, now being developed by the DMTF. There is a standard for access protocol LDAP that is being developed by IETF and the current version is LDAPv3. Some analysts speculate that LDAP client software will soon become a standard feature of Internet browsers and the standard protocol for accessing directory information over the Internet.

Most large enterprises already have a diverse collection of directories embedded in other systems, including email servers, firewalls and authentication servers. The recommendation to these organizations is to begin developing a strategy for eventual migration of this directory information to a common LDAP-based directory. Establishing a single common directory should be easier than maintaining the multiple directories that already exist today. By exploiting the emerging DEN schema standard in such a directory, the enterprise will be well positioned to add policy information when it is ready for policy-based networking (Figure 6.1).

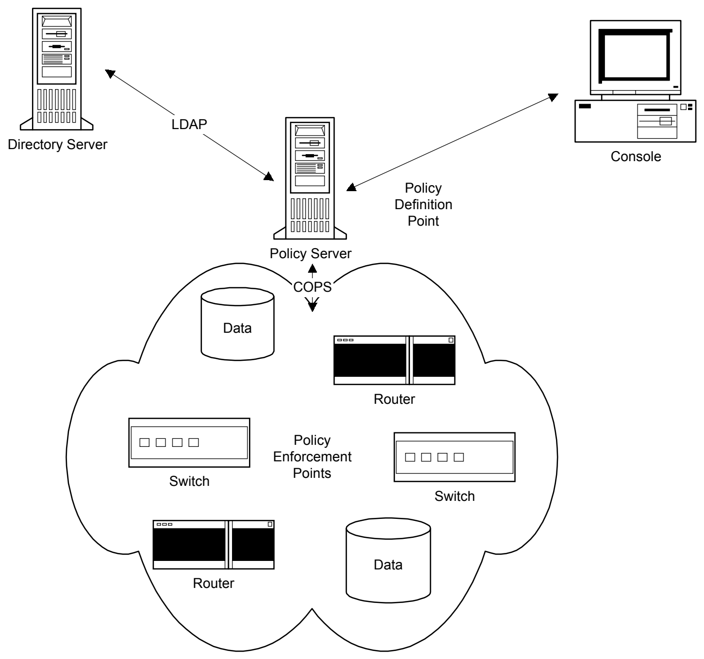

Figure 6.1 Policy Based Networking

Now when we know the theory behind the directory enabled networks initiative, it is time to define the problem.

7. Definition of the problem

Knowing the theory behind the DEN-initiative, it is now time to define Telia’s problems. In Sweden the past couple of months a discussion around broadband technology have been in focus in the media. One problem for Telia as an Internet Network Provider (INP) is that the network architecture today can not handle the requirements that new services will require. The new requirements on the network structure are:

- Flexibility
- Users should be able to use several Internet Service Providers, ISPs
- One user should be able to access the net in different roles
- Make new services possible (e.g. pay-per-view)

I am going to present a proposal how Telia could redesign their network based on the DEN-initiative. In order to do that it is necessary first to give a brief description of Telia's network (Telianet).

8. Telianet

In this chapter I will describe the logical structure of Telia's network called Telianet. Figure 8.1 shows a schematic sketch of Telianet.

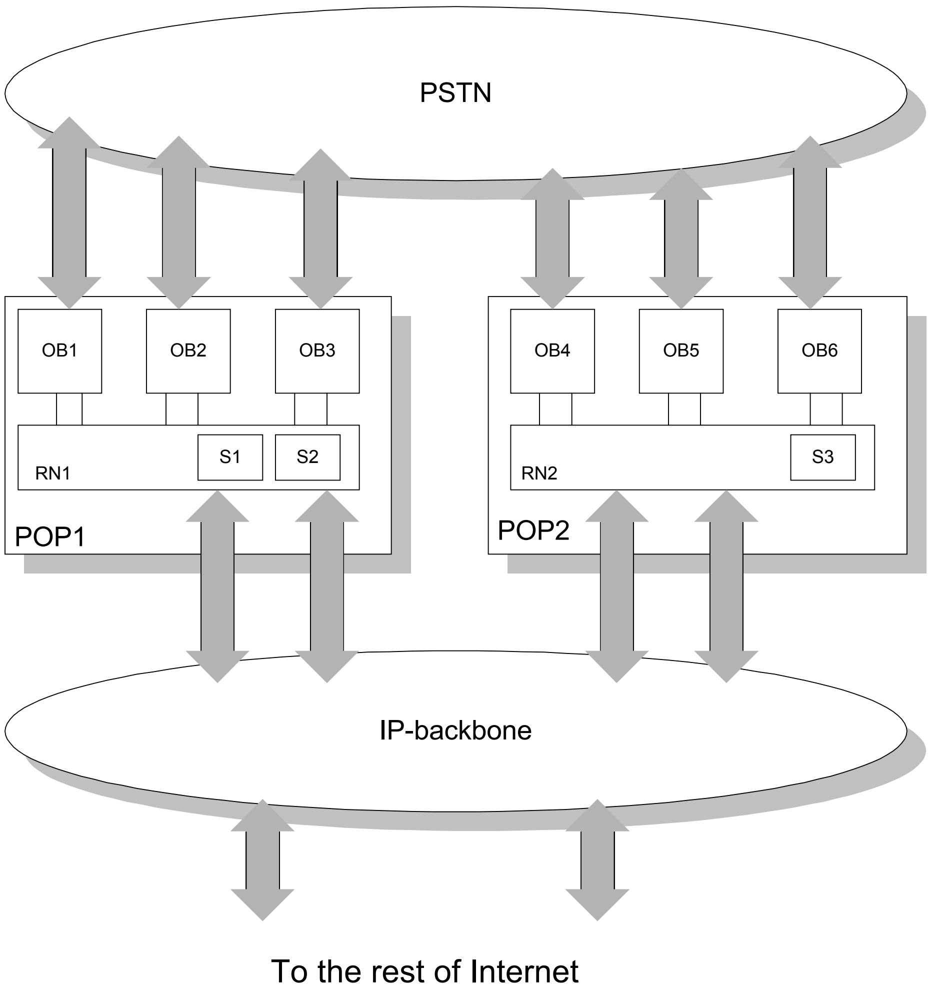

Figure 8.1 Logical structure of Telia's network, Telianet
OB = Ollbox RN = Regional Network
POP = Point of Presence S = Server
PSTN = Public Switched Telecommunication Network

The logical structure in Figure 8.1 can be described as follows; You connect to the Ollbox of the that is nearest to you. Ollbox is the name of the hardware used for remote access. This Ollbox and a couple of others are connected to a regional network, and all of these form something called a Point Of Presence, POP. The POP is your connection to the IP-backbone net. Telianet consists of around 70 ollboxes and 6-8 POPs.

When a client wants to access the Telianet in any purpose (e.g. browsing the Internet) he connects to an access server near him. Figure 8.2 shows how the access server with help of a RADIUS-server runs the authentication routine. The client dials a 020- number and depending on from where he calls he ends up in a certain access server. His user login data is just stored in one database (i.e. the client must call from his own telephone number to be able to access).

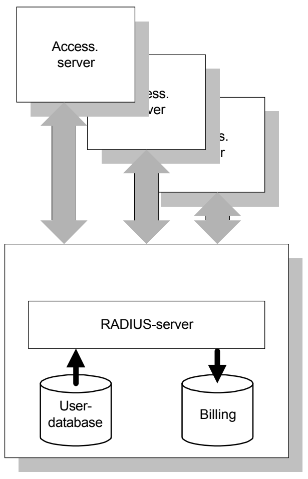

Figure 8.2 The authentication routine with help of a RADIUS-server

This restricts the possibilities of a flexible net (i.e. a person can not access the Telianet from a different location than his normal).

The net is built with a star configuration. It uses four different levels of routers. Core-routers form the highest level, the next level consists of distribution-routers, after them access-routers and the lowest level customer-access routers. A router on level (N) belongs to a couple of the routers on level (N-1).

Telianet Configuration Management (TCM) and Telianet Internet Management (TIM) are the two systems that monitor the net. TCM stores two types of data: system-, and production data. System data is user specific data (e.g. to which services does the user subscribe) and how the network shall act for a certain type of service is stored in the production data. TIM manages the network by collecting and storing data with help of SNMP-managers and detects failures with help of SNMP-traps.

POP3-servers are located around the country and you get a home-location address where your emails are directed to, but even if you move to another part of the country you would still be able to use your old POP3-server.

9. New services

Internet has grown from a playground for technicians to a natural part of life. To have access to the Internet is today as natural as having a TV. But the users are not satisfied with being able to find information and to look at pictures. People want more.
Three new services possible in the future are:

- Internet roaming
- Pay-per-view
- ISP-selection

Below I will describe each service, and the demands it requires of the network.

9.1. Internet Roaming

This section describes how the LDAP protocol can be used to transfer the Authentication Authorisation Accounting (AAA) information between AAA servers of remote ISP (R-ISP) and home ISP (H-ISP). This is very useful when one is travelling to a country with one's portable PC where the H-ISP has no Point of Presence (POP), but where another ISP (R-ISP) has a roaming agreement whith the H-ISP.

As shown in Figure 9.1, when the roaming user from H-ISP authenticates himself against the R-ISP's Access server, the LDAP client checks whether a profile for the roaming user is contained in R-ISP's LDAP server. The roaming user needs to authenticate as userID@H-ISP. In this way, the R-ISP's LDAP server understands that the roaming user belongs to a foreign ISP. By means of the LDAP "intelligent referral mechanism" it returns to the LDAP client the referral to the LDAP server where the roaming user's profile is most likely to be located.

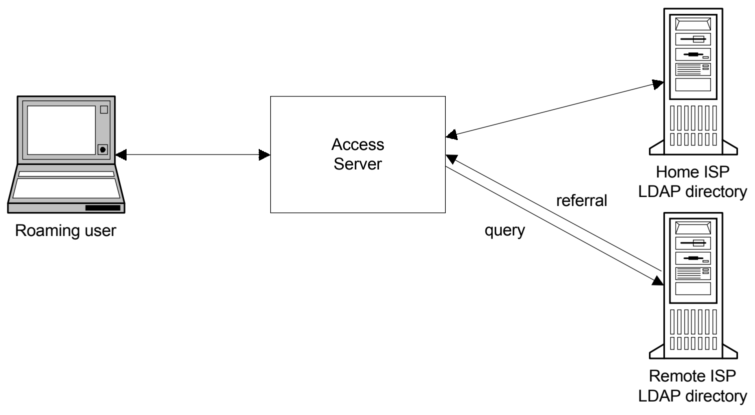

Figure 9.1 The roaming solution based on LDAP

The information in an LDAP referral is in the format of an LDAP Uniform Resource Locator (URL). A referral gives the following information:

- The host name of the server to contact.
- The port number of the server.
- The base DN (if processing a search operation) or target DN. This part is optional, if absent, the client should use the same base DN it used when performing the operation that resulted in the referral.

For example, if a client searches the subtree dc=telia, dc=se for all entries with a surname Lindgren (as in Figure 4.2), the referral would be returned as the following LDAP URL:

```
Ldap://server1.telia.se:389/ou=ProSoft, dc=Telia dc=se
```

9.2. ISP selection

A person must, regarding to the Post & Telestyrelsen (PTS) regulation SFS 1997:399, be able to choose which Internet Service Provider (ISP) he or she wants (Figure 9.2). This means that even if Telia owns the net, which connects the user to the internet, a user should be able to use services from a different ISP.

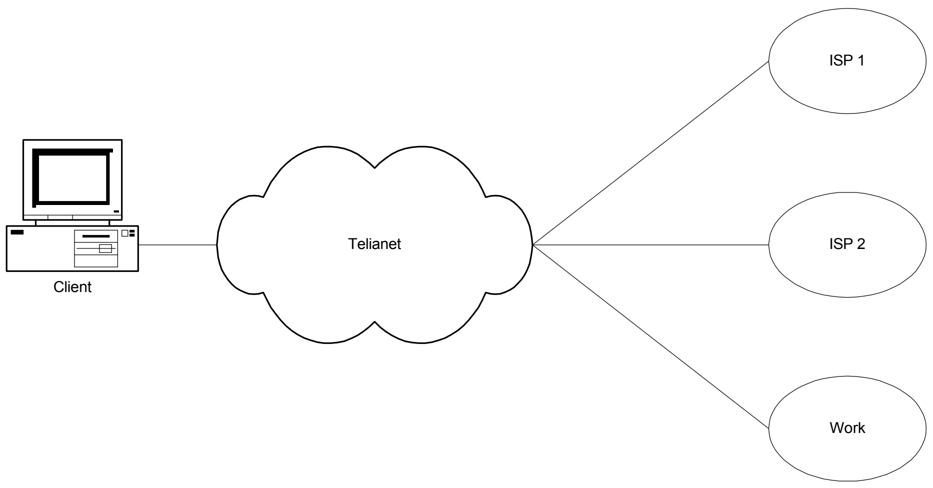

Figure 9.2 A client must be able to choose between different Internet service providers (ISPs)

To be able to support this it is important to build Telianet so, that it does not mean any security risks and that Telia is able to make a profit on it.

Using the DEN-initiative the solution is quite similar to the solution for Internet Roaming (based on that the other ISPs also are using LDAP-servers). One gets a referral to the ISP that he or she wants to use. A difference here is that the ISP can subscribe to the directory information so that not every ISP has to build own user directory. For example: user1 wants to connect to ISP1. He makes a LDAP query to his home directory and gets a referral to the right ISP's directory. When he is connected, he can order the services and the ISP gets the user ID from the user’s home directory. This could also be done so that the ISP updates its own directory automatically from a central LDAP-directory.

9.3. Pay-Per-View

Pay-Per-View is a service where a client via a user application is able to order a movie at a certain time. Figure 9.3 shows how this is implemented with an LDAP-directory and a Policy-server.

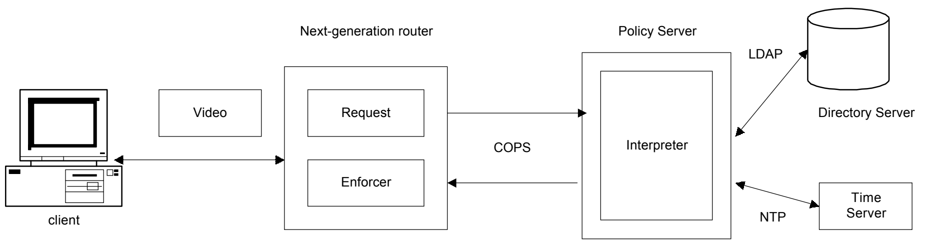

Figure 9.3 Pay-Per-View

The client decides which movie he wants to watch and sends a request to the nearest router, which forwards it to the policy server. The policy server checks the user-ID in the directory server to see that the user subscribes to this service. The policy server checks the time through a timeserver and when it is time to show the movie, and it looks up the user's IP-address in the directory server. Figure 9.4 shows what simple directory structure can look like in this case.

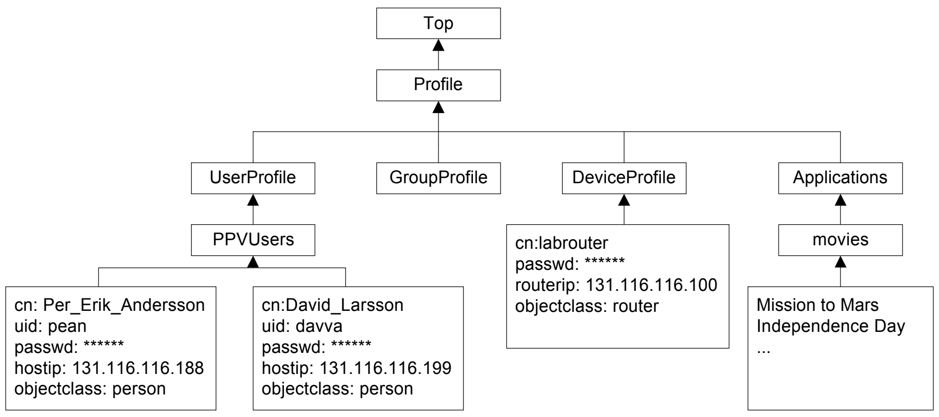

Figure 9.4 Simple directory tree structure for a pay-per-view service

This directory tree structure is the result of a Master Thesis by David Larsson [12].

10. Telianet based on the DEN initiative

As the Internet continues to grow, users want to use more complex services that require more bandwidth. In the DEN-initiative a policy structure with an enforcement-point (PEP) and a decision-point (PDP) is proposed (see chapter 5). What this structure looks like in a large network is shown in Figure 10.1.

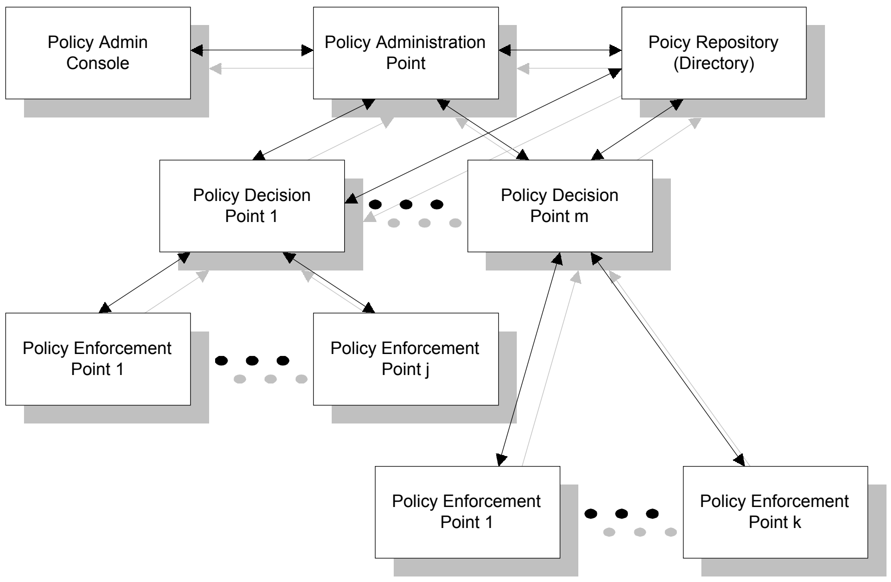
Policy Admin Console
Policy Administration Point
Poicy Repository (Directory)
Policy Decision Point 1
Policy Decision Point m
Policy Enforcement Point 1
Policy Enforcement Point j
Policy Enforcement Point 1
Policy Enforcement Point k

Figure 10.1 DEN Policy architecture

As seen in Figure 10.1, the DEN Policy architecture is designed so that several Policy Decision Points are connected to a Policy Administration Point and also to a Directory server. Each PDP has in turn several Policy Enforcement Points connected to it. A PEP can also be connected to more than one PDP if needed.

I have illustrated this in a logical sketch over Telianet based on the DEN-initiative in Figure 10.2. In this picture you can see that instead of a RADIUS-server the OLLboxes are using a LDAP-server. Furthermore the authentication routine (AAA) is moved out of the OLLbox into an LDAP-directory. This gives a flexible net, and the possibility for a user to connect to any OLLbox, not just his usual, and still be authorised. For example, user1.1 under OLLbox 1 can connect to OLLbox 3. OLLbox 3 will then try to find user1.1 in LDAP-directory 2. While the userprofile is not stored there, OLLbox 3 instead gets a referral to LDAP-directory 1 where user1.1's profile is stored.

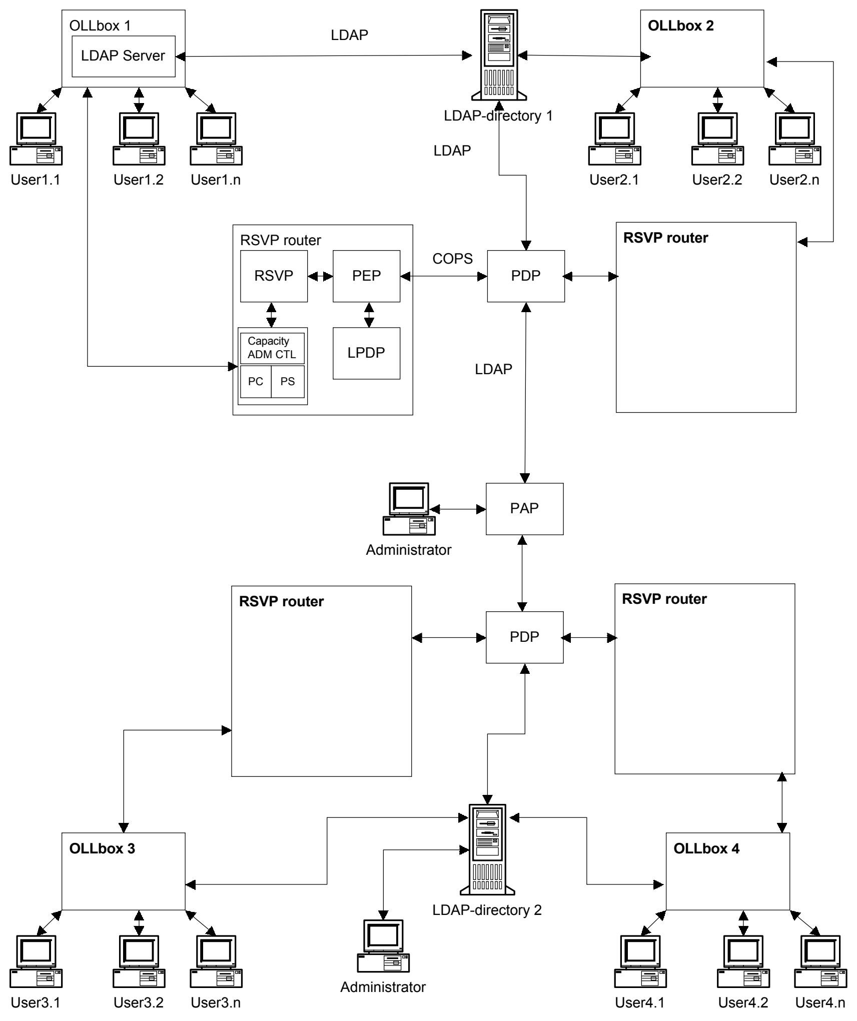

Figure 10.2 Logical structure of Telianet based on the DEN-initiative

It's important to understand when looking at Figure 10.2 that this is a very simplified sketch of a complex structure. In this case I have chosen to use Resource ReSerVation Protocol (RSVP) enabled routers to be able to make resource reservations.

What happens if user1.1 wants to use a pay-per-view service? Below is a listing of the steps:

1. User1.1 connects to OllBox1.
2. The LDAP-server in OllBox1 sends a request to LDAP-directory 1. If user1.1 exists in this directory and password and username are correct he becomes authenticated.
3. User1.1 access an ISP's web-server for pay-per-view services.
4. The web-server sends a query to LDAP-directory 1 to see if user1.1 subscribes to this service.
5. User1.1 can now reserve the movie he wants to watch.

6. The web server makes the reservation to the PDP, which forwards it to the PEP and finally it is put in the LPDP.
7. When it is time for the film to start the LPDP alarms the PEP. The PEP sends a request to the PDP for permission to open up the flow.
8. The PDP looks up the userIP-address in the LDAP-directory 1, and sends the decision to the PEP.
9. The PEP enforces the decision.

In chapter 5.6.1 it is described what the policy rules look like, and below in chapter 10.1.1 I describe how the core schema could look like.

10.1. Proposal of a LDAP-directory structure

Designing a directory namespace is one of the most important tasks when designing a directory service. A directory namespace provides the basic means for referencing information in a directory, but it has many other benefits as well. A properly designed namespace can lead to:
- Easier data maintenance.
- More flexibility in setting access control and replication policies.
- The ability to satisfy a wider variety of directory enabled applications.
- More natural navigation through the directory.

In Telia's case the directory will contain different people, with specific needs subscribing to various services. A person shall also be able to use varying profiles depending on which role he plays at the time he logs in. For example he should be able to work at home and then login as name@work.se. He should now be able to access his working profile and use all the services that his employer subscribes to. After working hour he logs in with name@isp.se, and has access to his own personal profile and the services he subscribes.

The Swedish municipalities is something that can be used in a directory. For example a person who lives in Stockholm gets the email address name@stockholm.telia.se. Then if a user, registered in Stockholm, logs onto a server in Gothenburg he will get a referral to the directory in Stockholm. This structure does not imply that every municipality needs to have a directory server, in fact small municipalities are able to share one directory server.

10.1.1. Core schema encoding example

In this section I will illustrate one way in which the policy in section 5.6.1 could be encoded. This is by no means the only way that this policy could be encoded, and only serves to illustrate the core schema action. I have also chosen to define a single policy rather than a policy group in which case I would need to define multiple policy rules, hence I do not reference the **policyGroup** class here. The classes I refer to, and their class hierarchy is shown in Figure 10.3.

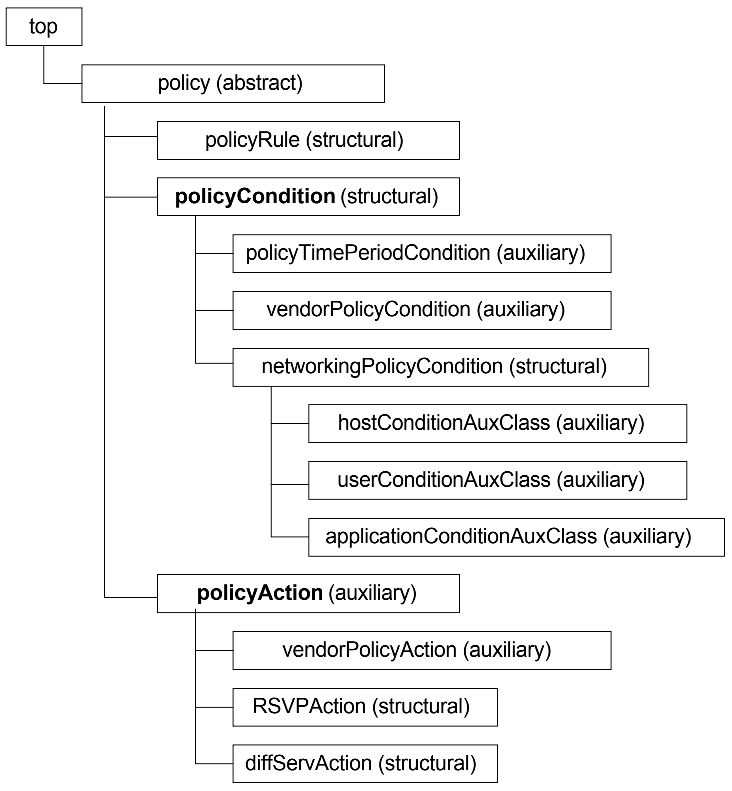
top
policy (abstract)
policyRule (structural)
policyCondition (structural)
policyTimePeriodCondition (auxiliary)
vendorPolicyCondition (auxiliary)
networkingPolicyCondition (structural)
hostConditionAuxClass (auxiliary)
userConditionAuxClass (auxiliary)
applicationConditionAuxClass (auxiliary)
policyAction (auxiliary)
vendorPolicyAction (auxiliary)
RSVPAction (structural)
diffServAction (structural)

Figure 10.3 Class hierarchy in core policy schema

Class: **policy** (type: abstract), Attributes:

- **CreationClassName** (required): "MPEG2Policy"
- **CN** (commonName, optional): "JitterFreeMPEG2Policy"
- **Caption** (optional): "Exclusive high quality video service"
- **Description** (optional): "Provide jitter free MPEG2 service within prescribed service parameters for users, during fixed time period, using approved source and destination addresses."
- **PolicyKeywords** (optional): "POLICY", "USAGE", "mpeg2", "network-video", "jitter-free"

Class: **policyRule** (type: structural), Attributes:

- **PolicyRuleName** (required): "JitterFreeMPEG2Rule"
- **Enabled** (optional): "enabled" (1) or "enabled for debug" (3)
- **ConditionListType** (optional): <use default value:DNF>

- **PolicyRuleConditionList** (optional): ordered list of policyAction DNs
- **PolicyRuleValidityPeriodlist** (optional): unordered list of policyTimePeriodCondition DNs
- **RuleUsage** (optional): boolean for rule evaluation
- **SequencedActions** (optional): <not used in this example>

Class: **policyCondition** (type: structural), Attributes:

- PolicyConditionName (required): "JitterFreeMPEG2Condition"

The policyCondition class is extended to the subclass **networkingPolicyCondition**, in which five auxiliary classes are defined.

Subclass: **policyTimePeriodCondition** (type: auxiliary), Attributes:

- **TimePeriod** (optional): "19991201070000:20000331190000" ("yyyymmddhhmmss" format, from 07.00 1 Dec. 1999 to 19.00 31 March 2000)
- **MonthOfYearMask** (optional): "111000000001" (mask indicates valid months: December-March)
- **DayOfMonthMask** (optional): (default is all days of month)
- **DayOfWeekMask** (optional): "1111100" (mask indicates valid days: Monday-Friday)
- **TimeOfDayMask** (optional): "070000-190000" ("hhmmss" format, from 07.00-19.00)
- **ApplicableTimeZone** (optional): "+0100"
  ('+' indicates east of Greenwich Mean Time, "hhmm" format)

Class: **policyAction** (type: structural), Attributes:

- **PolicyActionName** (required): "JitterFreeMPEG2"
  (This is the "user-friendly" common name for this policy action)

There exists two vendor specific classes **vendorPolicyCondition** and **vendorPolicyAction** that will not be discussed further.

Class: **hostConditionAuxClass** (type: auxiliary), Attributes:

- **SourceIPAddressRange** (optional): "10.0.0.1-10.0.0.5"
- **DestinationIPAddressRange** (optional): "10.1.0.2-10.254.254.254"
- **sourceHostID** (optional): <not used in this example>
- **destinationHostID** (optional): <not used in this example>

Class: **userConditionAuxClass** (type: auxiliary), Attributes:

- **senderID** (optional): <not used in this example>
- **receiverID** (optional): "Key-ID:666666666"

Class: **applicationConditionAuxClass** (type: auxiliary), Attributes:

- **sourcePortRange** (optional): <not used in this example>
- **destinationPortRange** (optional): "48879:57005"
  (pre-arranged destination port range in decimal values)
- **protocolNumberRange** (optional): "17"
- **receiveTOSByteCheck** (optional): <not used in this example>

Class: **RSVPAction** (type: structural), Attributes:

- **RSVPFlowServiceType** (required): ControlledLoad
- **RSVPPermission** (required): Accept
- **RSVPActionName** (optional): RSVPLowJitterDTV
- **RSVPMaxRatePerFlow** (optional): 4000(kbps)
- **RSVPMaxPeakRatePerFlow** (optional): 8000(kbps)
- **RSVPMaxTokenBucketPerFlow** (optional): 64(kb)
- **RSVPMinDelay** (optional): 2000(ms)
- **RSVPMaxFlowDuration** (optional): 180(s)
- **RSVPUserAuthPolicy** (optional): Public-Key
- **RSVPResourceGroupRef** (optional): <not used in this example>
- **DiffServActionRef** (optional): DN of diffServAction

Class: **diffServAction** (type: structural), Attributes:

- **diffServPermission** (required): Allow
- **diffServActionName** (required): DSLowJitterDTV (common name)
- **diffServInProfileRate** (optional): 4000 (kbps)
- **diffServInProfilePeakRate** (optional): 8000(kbps)
- **diffServInProfiletokenBucket** (optional): 64(kb)
- **diffServInProfileTransmittedTOSByte** (optional):
  "11100000:11000000"
- **diffServOutprofileTransmittedTOSByte** (optional):
  "00000000:00000000
- **diffServResourceGroupRef** (optional): <not used in this example>

In this example I have chosen to use both the RSVPAction and diffServAction classes since there could be both RSVP-enabled and Differentiated Services enabled routers in the net.

11. Conclusions

Since 1997 when Microsoft and Cisco announced the DEN initiative the development has been slow. Most vendors agree that a global standard directory schema (e.g. the DEN-initiative) is almost a must. What slows the standardisation down is the vendors who have to agree on how the schema shall look. DMTF is working with the DEN schema and IETF is trying to agree on a Policy-Transaction-Protocol standard. LDAP-servers and the protocol LDAP are gaining respect and shares of the market. What is really missing is a standard Policy-Transaction-Protocol. When there is a standard the vendors will be able to build new "intelligent" routers and switches. Then it is possible at first to use the advantages of managing a net based on the DEN-initiative.

11.1. DEN in Telia

Telia can start with installing LDAP-servers in the network, and use the protocol for queries and authentication, among other things. If Telia also starts to build up a directory according to the guidelines from the DEN-initiative (which takes time) they will be well placed when the standard comes. Telia should also become a full member of the DMTF, and in that way be able to follow and influence the standardisation work. Solutions on the market, as for example Novell Directory Services (NDS), need to be tested and evaluated for best result, when put in a large network.

11.2. Policy-Powered Networking

As the Internet grows, the need for a management system grows too. To manage a system one has to have certain rules (i.e. policies). Internet Service Providers must be able to decide which users get what information. That is why a working Policy-Powered-Network is important and would make several new services possible.

The intent of the Policy Working Group of the IETF fits nicely with the work underway in the networks working group of the Distributed Management Task Force, DMTF. The discussion in IETF today is about which protocol should serve as a policy transaction protocol, COPS or SNMP. Some are trying to make a modified version of the SNMP standard and others are pushing for COPS. Perhaps the best way would be to integrate COPS in SNMP and launch SNMP/COPS as a new protocol. My personal opinions is that SNMP shall be used for the same purposes that it is done today, and then launch COPS as the policy transaction protocol.

11.3. Products

Novell Directory Services (NDS), Netscape's Directory Server and Microsoft Active Directory are the three most popular directory products using the DEN-initiative's schema. Other vendors claim interoperability with all of these products, but provide value-added capabilities with only a preferred directory product. These three directories are all built to work with LDAP with the result that all applications or clients capable of using LDAP can access directory information. For example, a phone directory can easily and automatically be updated with new or changed information with an LDAP-enabled search application. To implement policy-based networking one should deploy switches and routers that can directly understand a policy transaction protocol. These switches and routers can generate requests to policy servers so that they do not have to rely on policies being pushed from the server. These routers will be more expensive (such routers or switches are not available commercially today), but are in the long run a good investment. It will be interesting to follow the progress in this area. The question is, if giants like Cisco and others are patient enough to wait for a standard, or if they will start to market routers that understands a certain protocol, and in that way create their a own standards.

Appendix A: Acronyms and Abbreviations

API	Application Programming Interface
BER	Basic Encoding Rules
CIM	Common Information Model
COPS	Common Open Policy Service
CoS	Class of Service
DAP	Directory Access Protocol
DBMS	DataBase Management System
DEN	Directory Enabled Networks
DHCP	Dynamic Host Configuration Protocol
DIB	Directory Information Base
DISP	Directory Information Shadowing Protocol
DIT	Directory Information Tree
DMI	Desktop Management Interface
DMTF	Desktop Management Task Force
DN	Distinguished Name
DNS	Domain Name Service
DSA	Directory System Agent
DSP	Directory Service Protocol
DUA	Directory User Agent
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
ISO	International Standards Organization
ISP	Internet Service Provider
ITU-T	International Telecommunication Union-Telecommunication Standardisation Sector
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interface Format
LDP	Local Decision point
MIB	Management Information Base
MIF	Management Information Format
NDS	NetWare Directory Services
NOS	Network Operating System
NSP	Network Service Provider
NTP	Network Time Protocol
OSI	Open Systems Interconnection
PBX	Private Branch exchange
PDP	Policy Decision Point
PEP	Policy Enforcment Point
PIB	Policy Information Base
PSTN	Public Switched Telecommunication Newtwork
QoS	Quality of Service
RADIUS	Remote Authorization DialIn User Service

RAP	RSVP Admission Policy
RMON	Remote Monitoring
RSVP	Resorce Reservation Protocol
SASL	Simple Authentication and Security Layer
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transfer Protocol
ToS	Type of Service
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLAN	Virtual LAN
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol

Appendix B: Organizations

Ad Hoc Working Group, AHWG (murchiso.com/den)

The goal of the AHWG is the rapid specification of a directory services information model and schema to facilitate the interoperability of distributed applications, management tools, and network elements. The Directory-enabled Networks Specification forms the basis of an implementers agreement in support of immediate customer requirements for leveraging network services and ensuring interoperability between networking applications of different vendors.

Distributed Management Task Force, DMTF (www.dmtf.org)

DMTF originally developed a method of modelling the components that define a typical desktop system: display, mouse, RAM, processor, etc. After leveraging the success and adoption of the Desktop Management Interface (DMI) standard for desktop management, the DMTF has evolved its mission to address distributed management through the use of its Common Information Model (CIM) standard. DMTF changed its name from Desktop-MTF to Distributed-MTF. The DMTF is the industry organization that is leading the development, adoption and unification of management standards and initiatives for desktop, enterprise and Internet environments. Working with key technology vendors and affiliated standards groups, the DMTF is enabling a more integrated, cost effective and less crisis-driven approach to management through interoperable management solutions.

Internet Engineering Task Force (www.ietf.org)

The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.
The actual technical work of the IETF is done in its working groups, which are organized by topic into several areas (e.g., routing, transport, security, etc.). Much of the work is handled via mailing lists. The IETF holds meetings three times per year.
The IETF working groups are grouped into areas, and managed by Area Directors, or ADs. The ADs are members of the Internet Engineering Steering Group (IESG). Providing architectural oversight is the Internet Architecture Board, (IAB). The IAB also adjudicates appeals when someone complains that the IESG has failed. The IAB and

IESG are chartered by the Internet Society (ISOC) for these purposes. The General Area Director also serves as the chair of the IESG and of the IETF, and is an ex-officio member of the IAB.

The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols. The IANA is chartered by the Internet Society (ISOC) to act as the clearinghouse to assign and coordinate the use of numerous Internet protocol parameters.

The Open Group (www.opengroup.com/directory)

The Open Group is committed to delivering greater business efficiency by bringing together buyers and suppliers of information systems to lower the time, cost and risk associated with integrating new technology across the enterprise.

The Open Group consortium ensures that multi-vendor information technology matches the demands and needs of customers. This is achieved through the development and deployment of frameworks, polices, best practices and standards in pursuit of the group's IT DialToneTM vision - the concept of making all technology as open and accessible as using a telephone.

By leveraging the expertise of its sponsors - Compaq, Fujitsu, HP, Hitachi, IBM, NCR, Fujitsu Siemens Computers, Sun - and other members comprising leading software companies, Fortune 500 users and government departments, The Open Group is well on its way to achieving the IT DialTone vision. Embracing the maxim "we can do so much more together than alone", The Open Group is helping its members to stay ahead in the race to use technology to gain competitive advantage.

References

[1] Boyle J, Cohen R, Durham D, Herzog S, Rajan R and Sastry A, "The COPS (Common Open Policy Service) Protocol," *Request for Comments 2748, Network Working Group*, Jan. 2000.

[2] Boyle J, Cohen R, Durham D, Herzog S, Rajan R and Sastry A, "COPS usage for RSVP," *Request for Comments 2749, Network Working Group*, Jan. 2000.

[3] Calhoun P, Rubens A, Akhtar H and Guttman E, "DIAMETER Base Protocol*," <draft-calhoun-diameter-11.txt>*, Dec. 1999.

[4] Case J, Fedor M, Schoffstall M and Davin J, "Simple Network Management Protocol, SNMP," *Request For Comments 1157, Network Working Group*, <http://www.ietf.org/rfc/rfc1157.txt>, May 1990.

[5] DMTF inc., "Common Information Model, CIM, Specification," <http://www.dmtf.org>, Jun 1999.

[6] Goodman D and Robbins C, "Understanding LDAP & X.500," *The European Electronic Messaging Association (EEMA),* v. 2.0, Aug. 1997.

[7] Howes T, "LDAP:Use as directed*," Data Communications International,* vol. 28, p. 95-104, Feb. 1999.

[8] Howes, T, Smith M and Good G, "Understanding and deploying LDAP Directory Services*,*" *Macmillan Network Architecture and Development Series, ISBN 1-57870-070-1*, 1999.

[9] ITU-T, "Recommendation X.500," *Telecommunication standardization sector of ITU,* Aug. 1997.

[10] Jagadish H.V and Lakshmanan L.V.S, "Revisiting the Hierarchical Data Model," *IECE Transactions on Information and Syst*em, vol. E82-D, no.1, p. 3-12, Jan. 1999.

[11] Judd S and Strassner J, "Directory Enabled Networks - Information Model and Base Schema ver.3.0c5," <http://murchiso.com/den/specifications/directory-enabled-networks-v3-lastcall.pdf>, Oct.1998.

[12] Larsson D, “Evaluation of IP Multicast in supporting IP based digital Television,” *Exjobb på Telia ProSoft AB: RE BredBand*, Jan. 2000.

[13] McCloghrie K and Fine M, "A Comparison of Policy Provisioning Protocols," *<draft-kzm-policy-protcomp-00.txt>*, Oct. 1999.

[14] Saperia J and Schoenwaelder J, "Policy-Based Enhancements to the SNMP Framework," *<draft-schoenw-policy-snmp-00.txt>*, Sep. 1999.

[15] Schreiber SB, "Examining Microsoft's LDAP API," *Dr Dobb's Journal*, vol. 23, no. 12, p. 108-111, Dec. 1998.

[16] Stevens M, Weiss W, Mahon H, Moore B, Strassner J, Waters G, Westerinen A, and Wheeler J, "Policy Framework*," <draft-ietf-policy-framework-00.txt>*, Sep. 1999.

[17] Strassner J and Ellesson E, "Terminology for describing network policy and services," *<draft-ietf-policy-terms-00.txt>*, June 1999.

[18] Strassner J, Ellesson E, Moore B and Moats R, "Policy Framework LDAP Core Schema*," <draft-ietf-policy-core-schema-06.txt>*, Nov. 1999.

[19] Wahl M, Howes T and Kille S, "Lightweight Directory Access Protocol (v3)," *Request for Comments 2251, Network Working Group*, <http://www.ietf.org/rfc/rfc2251.txt>, Dec. 1997.

[20] Weiser R and Stokes E, "LDAP v3 Replication Requirements," *<draft-ietf-ldup-replica-req-02.txt>,* Oct. 1999.