第 8 部

ネットワーク管理

概要

インテリジェント ネットワーク管理

ネットワーク管理システムにおけるエキスパートシステムと知識ベース利用の試みは，管理システムそれ自体と同じくらい古くからあった．ここでは，知識獲得のためのボトムアップアプローチを提案する．知識獲得のためのスタートポイントは，システムの管理オブジェクト (MO: Managed Object) の集合である．ネットワーク管理システムにおいて，MO は管理の目的によって，そして管理のために観測されるように資源を表現する通信資源の抽象的な見方である．MO の集合は，管理情報ベース (MIB) を形成しており，有限である．これらの MO のおのおのに対応する知識は，管理情報知識ベース (MIKB) を形成している．MIKB のモデルを提案し，その設計と詳細なインプリメント方法について議論する．

ディレクトリにおけるネットワークの図形化

通信ネットワークについての基盤情報およびサービス関連情報が，あらゆる場所から，いつでもリーズナブルに効率的な方法で，しかも正確にアクセスされうるような枠組が必要とされている．ここでは，通信ネットワークにおいて，すべての関連する詳細事項や記述を X.500 ディレクトリ内に表現するモデルを提案する．この目的のために使われるオブジェクトとそれらの属性のスキーマを開発中である．そのモデルに対し，物理的オブジェクトと物理オブジェクトのいくつかの論理的抽象化を考えている．このネットワークマップ概念に基づくいくつかのアプリケーションについて議論する．

ConMan プロジェクト

本プロジェクトの目的は，X.500 ディレクトリの中の有効なネットワーク情報を使用して構成管理機能を提供することである．構成情報は，ディレクトリから集められ，グラフィカルに表示される．また，グラフィカルエディタによって，ユーザや管理者が表示に手を加えることも可能である．それゆえに，このように生成された情報が最適化，サービス管理，モニタリング，管理等の目的のために使われるようなネットワークのいろいろな局面に関連するコンピューテーションや意志決定が可能である．

SoftPages プロジェクト

本プロジェクトは，混雑した海外とのリンクや国内バックボーンでの FTP トラッフィックを低減したいという要望から始まった．基本的な考え方は，ユーザサイトに最も近く，コストが低く済むファイルサーバに問い合わせることである．この目的のため，ユーザホストと可能なサーバ間の距離とコストを評価すること，さらにサーバの中身を検索する必要がある．SoftPages は，ネットワーク構成に関するネットワーク基盤情報，すなわちサーバの位置やその中身，リンクやその通信費用等このドキュメントに書かれた情報を使用する．SoftPages は，このスキーム中のアイデアのためのテストベッドを持つ．

NIC サービス

ユーザに情報サービスを提供するために NIC によって運用されている集中型データベースサーバが，拡大しているインタネットのニーズや規模にもはや応えられないことは明らかである．我々は，X.500 ディレクトリがこのデータベースとして働く能力をもっていると信じている．人間と組織に関する情報を持つグローバルディレクトリは，既に運用中であるが，ネットワークそれ自体についての情報と組合わせることにより，データは新しい特質を達成するだろう．例えば，ある IP ネットワークアドレスを調べることにより，そのネットワークの記述，ネットワークを保持している組織の十分詳細な記述，そのネットワークに接する人間についてのすべてのデータ，隣接ネットワークへのリンクを含む外部ゲートウェイ等を知ることができる．NIC サービスに基づくディレクトリの実現に関する問題について議論する．

SoftPages プロジェクト

SoftPages の基本的な考え方は，技術的，ポリティカルなネットワークパラメータに気を付けながらユーザの観点からファイルサーバを評価することである．比較を簡単化するため，各ファイルサーバ (すなわちユーザのサイトとファイルサーバのサイト間のコネクション) は関連するコスト指数を付加される．低いコスト指数を持つコネクションはファイル転送に都合がよい．このコスト指数の計算方法等について説明する．

ネットワーク管理のためのディレクトリミニパイロット

前の章に書かれた提案とアイデアは，X.500 ディレクトリに関連している．我々は提案の適用性を証明するためにディレクトリの実験を行なった．X.500 ディレクトリは，データベースを保持する Directory Syetem Agents (DSAs) と，DSA に接続してデータベースに実際に問い合わせる Directory User Agents (DUAs) から構成される．我々は，予め定義されたオブジェクトクラスや属性タイプの定義の集合を拡張したため，DUA と DSA に変更を行なった．

第 1 章

The Management Information Knowledge Base

The attempt to use expert systems and knowledge bases in Network Management systems has been as old as the systems themselves. Generally the approach of knowledge acquisition has been top-down or random. In the top-down approach a specific problem or issue of network management is divided in to sub problems/issues till knowledge specific to the system can be acquired. In the random approach whatever knowledge is available is acquired and used wherever/whenever applicable. Naturally, both approaches suffer from lack of a complete coverage of the problem-space. In this work we have proposed a bottom up approach in which the starting point for knowledge acquisition is the set of *Managed Objects (MO)* of the system. In a network management system an MO is an abstracted view of a communication resource that represents the resources as seen by and for the purposes of management. The set of MOs form the Management Information Base (MIB) and is finite. The knowledge corresponding to each of these MOs form the Management Information Knowledge Base. parameters are finite. In the following sections we present a model of the MIKB and discuss its design and implementation details.

1.1 A Model for MIKB

The human manager's view of the network is limited to the objects defined in the MIB. MIKB is the semantic image of the MIB. In order to arrive at this image, we needed comprehensive knowledge about the MOs, which we gathered by interviews with network managers. We took each managed object (MO) in the MIB in turn, and asked the network managers what this MO represented to them. The qualitative information thus gathered, was quantified, and put in the MIKB.

We approached the task of creating the MIKB from two angles. On the one hand, we got the qualitative information from the network manager; on the other, we classified the managed objects by object type, and tried to see if the information given by the network managers could be categorised by object type. Subsequently, once we were satisfied with our classification procedure, we quantified the information.

1.1.1 Classification of MOs by object type

An examination of the MIBs for TCP/IP based internets shows that MOs are basically of the types shown below. The information content of various types of objects from the management point view is also given.

- Status : generally gives a direct indication of the state of a system; for example whether the status of an interface is 'UP', 'DOWN' or 'TESTING'

- Counters, Gauges : generally a measure of some quantity (e.g. input packets, output errors, Q-lengths, ..) that gives an indication of the performance and/or status of the system. There are generally allowable/normal values and when the values go beyond these thresholds, the situation calls for special investigation/action.For example, continued and rapid increase of traffic is cause for alarm.

- Descriptive objects, sequences, tables : Changes in these objects generally indicate some change in the network and may merit an investigation. For example a change in the routing table is indicative of the addition/deletion of some link somewhere. A deletion of a link is in turn indicative of some faulty gateway/interface.

- TimeTicks : measures the time since an event has occurred.

1.1.2 Knowledge from the network managers

In our sessions with the network managers, we analysed the MIB MO by MO, asking the network manager the following questions:

- What kind of information would your monitor for this object
- What does this object indicate to you (congestion, error prone system etc.)
- What kind of diagnosis do you make from the errors detected

The answers to the questions above, for selected examples, is given below:

- mgmt.mib.system.sysUpTime
  - Can be used to diagnose system breakdown
  - History needs to be stored
  - Max sysUpTime ”small” indicates an error-prone system
- mgmt.mib.interfaces.ifTable.ifEntry.ifOperStatus
  - Status of the interface
  - We need to know how long an interface has been down, how often
  - How critical is the functioning of this interface (gateway?)
- mgmt.mib.interfaces.ifTable.ifEntry.ifInOctets
  - Detect congestion with ifInDiscards
  - Use averages and peaks
  - Store history
  - If congestion continues for an hour, declare ”congestion state”
- mgmt.mib.interfaces.ifTable.ifEntry.ifInErrors
  - Monitor general health
  - Should be less than 0.025% of ifInUcastPkts
  - For a leased line, 1/s maybe ok, but 60/mt bad
  - Error continuing for 2-3 mt *may* be okay

From the answers to the above questions, we realised that we need to derive more information about the MOs, from its current value and history. The current value of a MO is a valuable piece of information. It tells a knowledgeable person more than just the figure, but only when read in the context of the history of the object. The frequency at which the object representing the operational status of an interface is changing gives the operator an insight into the reliability/error proneness of the interface. The same holds for counter and gauge type objects. The information that some counter has a value x does not signify much when judged in isolation.

The significance is seen by the manager by using the history of the object to estimate the time development of the MO. For example, it is crucial to know whether an error count is constantly increasing, rapidly increasing, or only transient. For a more quantitative analysis, it was clear that the *velocities* and the *accelerations* would be effective measures to capture time-development related derived information of Counter and Gauge type MOs.

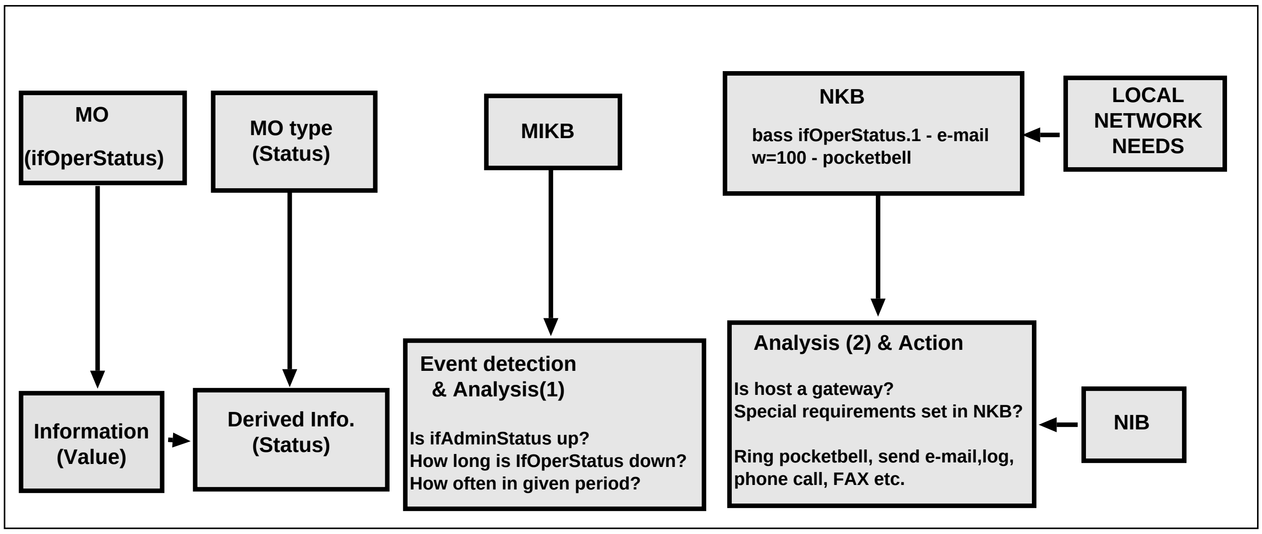


図 1.1: Flow of information (MIKB, NKB and NIB)

The history of MOs are described in terms of averages, peaks, deviations etc. depending on the object type, on a periodic basis (hourly/daily/weekly ..) and are retained in the Network Information Base (NIB), described in detail in later.

The data in NIB are indications of the patterns, if any, that the MOs are having in space and time and provide valuable indicators in determining the significance (normal/abnormal) of the current states of MOs. The details of MIKB and NKB are described in later.

In the model developed so far, MOs have been treated individually. However the inter-dependency of the MOs does appear several times in the criteria for deciding the abnormality of an object. For example, the time threshold beyond which if the OutQLen persists (continues to be non zero) the condition will be labelled abnormal, is dependent on the speed of the interface as shown in the example. Also, the Operational Status of an interface being down will be considered abnormal only if the administrative status of the interface is UP.

1.2 MIKB Design and Operation

From the discussion above, we can see that we need to put all this information in an accessible form, and also design the MIKB in such a way that we can detect *possible* abnormalities easily, and confirm/dismiss the abnormality by a detailed analysis with the help of the values stored, and further, diagnose the abnormality on the basis of the history and other details.

1.2.1 MIKB prototyping

The MIKB template, given in the next section, was arrived at after a few trials and errors. We started the testing by having the prototype, which had *all* the conditions and analysis defined together. While working with the prototype, we found that it is absolutely necessary to have the abnormalities defined as easy to detect events (EVENTS in the template below), so that detection of *possible* abnormalities is easy. The detailed analysis (ANALYSIS) follows this detection.

1.2.2 MIKB template

The template of the MIKB is as given below.

```
<object> OBJECT-TYPE
        OBTYPE  <timeticks, dependent, status, threshold, table>
        ATTRBT  value, stval, ltval, vel, stvel, ltvel,
                pkval, avval, pkvel, avvel, count, time
        EVENTS  {
                 (attr <!/</=/> n)
                  .....
                }
       ANALYSIS {
                (MO.attr <!/</=/> n) .....  w=wt,
                    .........................
                }
         ::= { parent entrynumber }
```

Based on the object type of MO defined in the MIB, the OBTYPE above is decided. The ATTRBT field gives the values for this MO which need to be monitored. The EVENTS are easy to detect conditions, which *may* indicate an abnormality. The ANALYSIS is the actual intelligence, a set of conditions which need to be evaluated to confirm or reject the abnormality. If the abnormality is confirmed, the given weight (w=wt), is set, and subsequent actions are performed from NKB, either based on this weight, or on the host and MO where the abnormality has been detected.

1.2.3 MIKB example

Given below is the template described above, applied to the managed object, ifOperStatus, which indicates the operational status of an interface. The value UP(1) indicates that the interface is operational (hence no need for any further processing), the value DOWN(2) indicates that the interface is down, and the value TESTING(3) indicates that the interface is being tested. In the following example, we have shown that ifOperStatus is dependent on ifAdminStatus, which is the administrative status of the device. This is the MO which indicates what operational status the interface *should* have. For example, if the administrative status is DOWN or TESTING, the operational status is of no concern.

```
ifOperStatus OBJECT-TYPE
  OBTYPE  status
  ATTRBT  value, count, time
  EVENTS  {
          (value = DOWN)
          (value = TESTING)
          }
 ANALYSIS {
          (ifAdminStatus.value=UP) (value=DOWN) (count>10) (time<7d) w=100 count=0 time=0,
          (ifAdminStatus.value=UP) (value=DOWN) (count>3)  (time<7d) w=20,
          (ifAdminStatus.value=UP) (value=TESTING) (count>10) (time<7d) w=70 count=0 time=0,
          (ifAdminStatus.value=UP) (value=TESTING) (count>3)  (time<7d) w=10,
          }
  ::= { ifEntry 8 }
```

Though the values have been given in the MIKB itself, we realise that these values may not hold good under all conditions, for all nodes and networks. Hence, we have provided the facility to set user selected thresholds; the user has the option of gathering statistics regarding *this* particular network for a period of time, and tuning the MIKB to reflect the appropriate values. This is covered in more detail in the section on implementation.

1.3 Implementation

The implementation details are shown in Fig. 5. There are three major stages in the flow of events. The first stage is 'tailoring' the MIKB to suit the local conditions; the second is the monitoring of selected MOs. The third is the analysis of events detected, and subsequent action triggering in case the event is confirmed.

The values of the MOs for a given period (preferably an uneventful period), are already stored in the database. The user consults these values, and the template given in the MIKB, to create the files which will subsequently control the flow of the system. The poller periodically updates the current status of the MOs under observation.

Detection of events specified in the MIKB triggers the detailed analysis. Detailed analysis is done based on the history and current value of the MO, and values of other MOs in the case of interdependent objects. The history of the object is maintained in the Network Information Base (NIB). Once the situation is evaluated completely, if the event detected is confirmed, the weight given is set, and the specified action in the Network Knowledge Base (NKB) for this weight is performed. This flow is illustrated in Fig. 5.

1.3.1 Tailoring

This consists of creating the three files, poller.cf, MOevent.cf and MOanalysis.cf, and this involves choosing the thresholds and values for selected MOs for event detection, with the help of the MIKB and values of the MOs for a given period under normal conditions (Fig. 4). The MIB, MIKB and the database containing MO values for an uneventful period, are consulted. The user selects the MO from the MIB, and the 'intelligence' regarding this MO in the MIKB is displayed, along with the appropriate values for a 'normal' week. The user has the option of modifying the values to suit the system and the current environment. As the MOs are identified by host, the user also has the option of setting different values depending on the host. For example, the status of the interface which is the gateway, is more critical than the status of an inteface to an end work station. Or again, depending on what the traffic is normally, we need to set the thresholds and so on.

As we see, the MIKB is used only during the initial setup (and whenever the user wants to change this). The user consults the MIKB, modifies the values if need be, and sets up a table, which is tailor-made to the system. Subsequently, the events and actions are evaluated from the table, not the MIKB.

1.3.2 Monitoring

The poller polls the set of MOs at periodic intervals, and creates a file containing the MOs with their current status. The monitor programs consults this file and the MOevent.cf, which contains the events associated with these MOs, and determines if any events have occurred. If an event is detected, the third phase, analysis and action, is triggered off.

1.3.3 Analysis & Action

Detection of an event triggers this phase. The details of MO intelligence are implemented in this phase; for example, once we have detected that the operational status of an interface is down, we come here to check if the administrative status is up, and also how many times in a given period of time this operational status has gone down; or how long it has been down. In order to do this, we may need to consult the history of the object, stored in the NIB. The analysis phase also determines the seriousness of a situation, and depending on this, the action is chosen from the NKB file and performed.

1.3.4 Network Information Base

Network Information Base contains the history of the MOs. This information is consulted for detailed diagnosis. As we can see from the example given above, we need to know the past behaviour of an MO before we can decide on the seriousness of the current situation. In the NIB all events and short/medium/long term statistical charactersistics of the MOs are recorded. This allows the Network Manager application to make intelligent decisions, by looking at the current status of an MO in the perspective of the history of the MO and that of the network. The NIB is maintained in a DB form and is accessible for offline diagnosis and maintenance purposes too. The details stored in the NIB depend on the attributes defined in the MIKB for this MO.

1.3.5 Network Knowledge Base

The focus of the network management knowledge base so far has been the MIB, i.e. we have been concentrating on that part of the knowledge which is not specific to a network. However, in actual management several judgments, decisions and actions are dependent on the local environment and configuration. Thus it may so happen that the MIKB is indicating the likelihood of an abnormality but the NKB on the basis of some local knowledge, overrides it. Also, the importance of an event may be further emphasized in the NKB. For example, all interfaces are important and should be operationally UP but some interfaces, on specific hosts, may be more important than the others. Actions are generally logging messages, sending e-mail, fax messages and in the most critical cases ringing pocket-bells, etc. These of course very much depend on the local circumstances and are coded in the NKB.

The NKB is the interface through which the local administrator can tailor the management system to suit their specific requirements. Along with setting down what action is to be performed for the different weights set, this also contains information on whether there are special requirements for specific hosts and MOs.

The KB is formed by the union of the rules in the MIKB and the NKB, with the rules in the NKB deciding the actual actions. The NIB gives the information regarding the history of the object. Hence the total KB may be said to contain of the MIKB, NKB and the NIB.

第 2 章

Network Maps - Concept, Realization & Applications

2.1 Configuration Information for Network Management

The rapid and widespread use of computer networking has highlighted the importance of holding and servicing information about the networking infrastructure itself. The growing and active interest in network management [46], which has concentrated mainly in the areas of fault and performance management on a local scale, is severely constrained by the lack of any organized pool of information about the network infrastructure itself. Some attempts have been made, on a piecemeal basis, to provide a larger view of some particular aspect of the network (WHOIS, DNS, .. in the case of the Internet; [47], [48]). But to date, little or no effort has been made in setting up the infrastructural framework, for such an information pool. In this work we explore the possibility of setting up a framework to hold and serve the infrastructural information of a network.

2.1.1 Infrastructural information requirements

Network operation and management requires information about the structure of the network, the nodes, links and their properties & functions. Further, with current networks extending literally beyond bounds, the scope of the information covers networks beyond the span of local domain of authority or administration. When the Network was relatively small and simple the map was already known to the knowledgable network administrator. Based on this knowledge the course of the packets to different destinations would be charted. But presently the size of the Network is already beyond such usages, with the current growth being near explosive. This is giving rise to the urgent necessity of having infrastructural and service related information made accessible from all places and at all times in a reasonably efficient manner and with reasonable accuracy. In the rest of this work a *network* is the media for transmitting information. *Network* *elements* are equipment with one or more *network interfaces* whereby it is possible to exchange information with the network. *Network elements* with multiple interfaces e.g. gateways/routers/bridges/repeaters... may be used to connect networks. Network related information, referred to as 'network map' in the rest of this paper, should

1. Show the interconnection between the various network elements. This will basically represent the Network as a graph where vertices represent objects like gateways/workstations/ subnetworks and edges indicate the connections.
2. Show properties and functions of the various network elements and the interconnections. Attributes of vertices will represent various properties of the objects e.g. speed, charge, protocol, OS, etc. Functions include services offered by a network element.
3. Contain various name and address information of the networks and network elements
4. Contain information about various administrative and management details related to the networks and network elements.
5. Contain the policy related information, part of which may be private while the other part may be made public.

Using this map the following services may be provided

1. Configuration management:
   - Display the physical configuration of a network, i.e. nodes and their physical interconnections
   - Display the logical configuration of a network, i.e. nodes and their logical interconnections.
2. Route management:
   - Find alternate routes by referring to the physical and logical configurations.
   - Generate routing tables considering local policy and policy of transit domains
   - Check routing tables for routing loops, non-optimality, incorrect paths, etc.
3. Fault management: In case of network failures alternatives may be found and used to bypass the problem node or link.
4. Service management: Locate various services and servers in the Network.
5. Optimization: The information available can be used to carry out various optimizations, for example cost, traffic, response-time, etc.
6. Provide mappings between the various names and addresses of elements
7. Depict administrative/autonomous domains.
8. Network Administration and Management: References to people responsible for administering and technically maintaining a network will be useful.

2.1.2 Nature of the network map

Implementing and maintaining a detailed map of the network poses a serious problem. The scope of the map is global and the network itself is expanding. Some of the problems that are peculiar to the network map are listed below:

- The network configuration is quasi-static. Nodes, links and networks are being added, updated and deleted someplace or the other.
- The network is huge and geographically distributed.
- The network spans several political and administrative areas. The related information is also controlled and maintained in a distributed fashion.

In short, global network configuration information is unwieldy and growing continuously. It is impossible to service such information in a centralized fashion. So, a distributed database system is necessary. In this context, the X.500 Directory system [49] is an appropriate candidate on which the network map be implemented. The X.500 Directory is intended to be a very large and highly distributed database. It is structured hierarchically with entries arranged in the form of a tree in which each object corresponds to a node or an entry. Information is stored about an object as a set of attributes.

2.1.3 Hierarchical model of a network

For representing networks in the Directory we use a hierarchical model.

A network is the media for transmitting information with zero or more network elements each having at least one network interface on the media. The media may be a line (physical circuit/virtual circuit), a coaxial-cable, or a collection of interconnected networks.

The model allows hierarchy of subnetworks. Network elements with multiple interfaces may act as *external gateways* to the attached network and to networks higher up in the hierarchy. Thus, a gateway may be the external gateway of several networks which are either interconnected or have a hierarchical relationship.

2.1.4 Network Maps

A network may be *simple* (consisting of zero or more network elements) or *composite* (consisting of several sub-networks). Examples of simple networks are Ethernets, Optical fiber/copper cables, free space, ...

Using the above model it is straight forward to draw the topological graph of the network where the vertices represent the components of the network and edges indicate

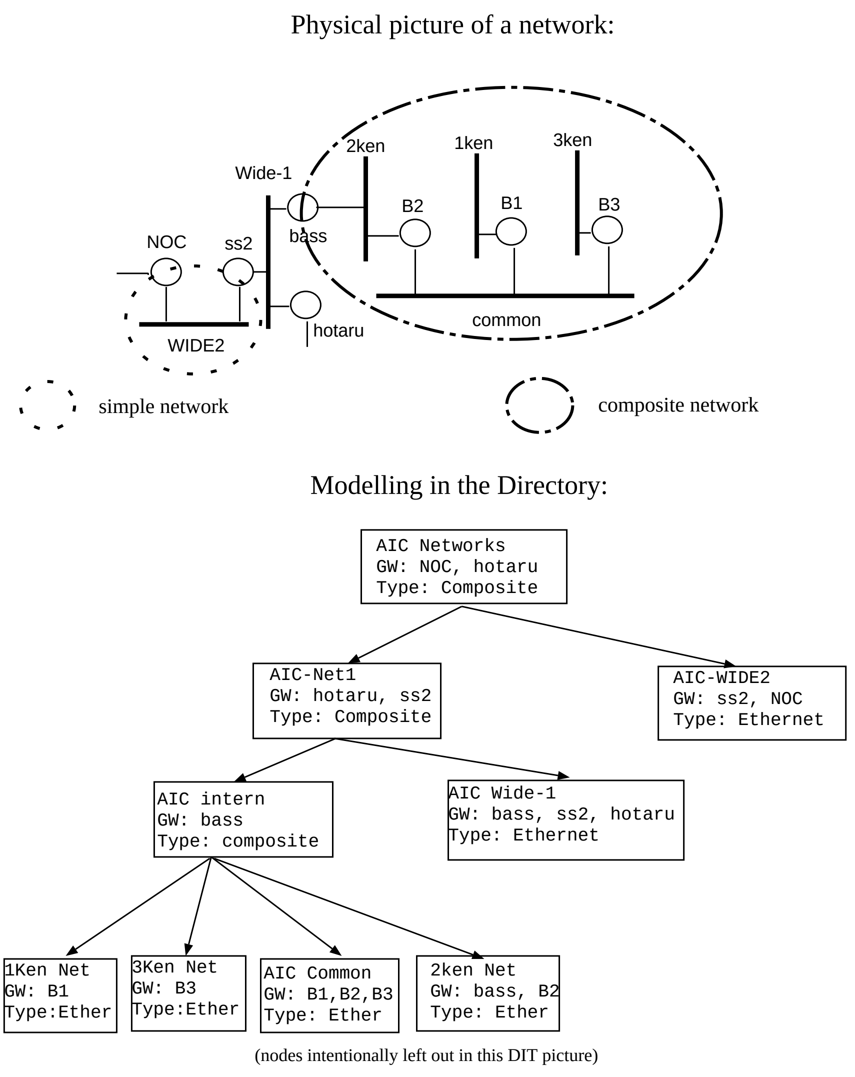


図 2.1: Simple and composite networks and their mapping to the DIT.

the connections (fig. 2.1). For visual representation the graph may be translated to a more "physical" illustration.

Just as there are several maps of the same geographical domain (political, natural...) one can envisage several views of the same network and its components. A view (called "image" in the remainder) could pertain to a particular protocol suite (IP/OSI/...), an administrative domain or purpose. Using images, several abstractions of the same object are possible.

2.2 Representation of Network Maps in the X.500 directory

The issues that needed to be taken care of while designing the schemas are as follows:

1. Scalability
2. Distribution of control & maintenance
3. Preservation of administrative boundaries/controls.
4. Simple representation should closely resemble the real world situation
5. Minimize data duplication
6. Bootstrapping strategies
7. Security : Though no special emphasis has been placed
8. Bootstrapping strategies
9. Security : Though no special emphasis has been placed in this work we believe the X.500 access control policies will provide reasonable control.

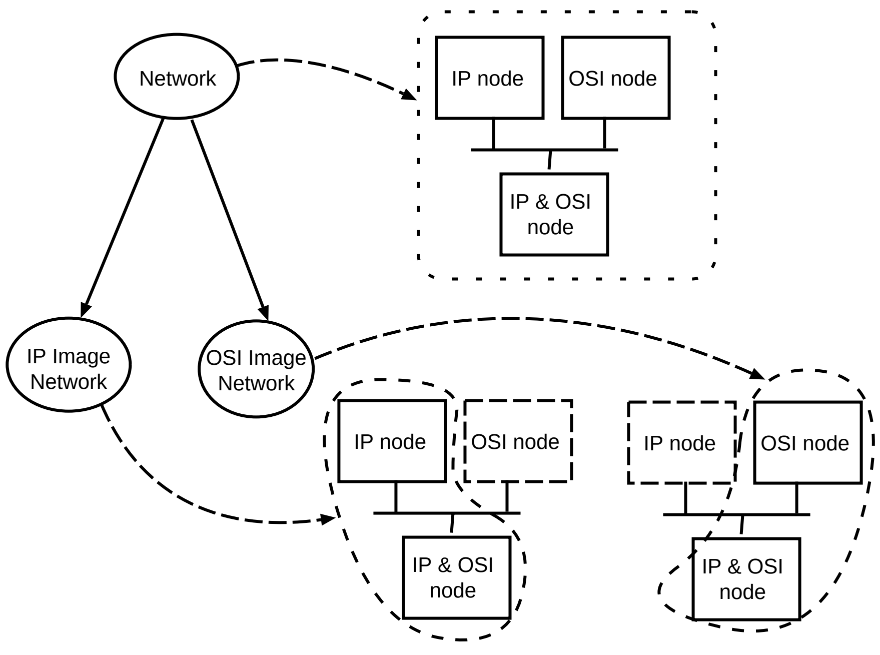
Issues like caching policies/ replication etc are not addressed- and are expected to be locally resolved.

2.2.1 Directory object classes for networks

A physical network comprises of wires and machines. The physical map of the network will show the interconnections of these nodes by these circuits. For each physical network element, one or more images may exist. Similarly, an image may be attached to one or more physical objects. The types of images can grow along with the requirements. Relationship between elements (physical or logical) are expressed by attributes or the position in the Directory tree. To represent the various images of networks and its components along with the real-image relationship among the various objects we introduce the following classes of objects:

- Communication Object Class (CO): All objects defined furtheron in this document belong to this class. Common attributes for all communication objects are defined in subsection 2.2.2.
- Physical Communication Object Class (PCO): A subclass of CO-class, this class defines common properties for all objects representing physical communication objects.
- Image Communication Object Class (ICO): A subclass of CO-class, this class defines common properties for all objects representing images of communication objects.

The above classes sort communication objects into physical or image object. As is implied in the nomenclature a physical object will have several attributes describing it physical properties - location, weight, size, ... etc. An image object will have an *Image-of* attribute. The *Image-of* attribute will point to a physical object or to another image object.



FUNCTIONAL IMAGES OF A NETWORK

図 2.2: Several logical views of the same physical network

Using this scheme it is possible to represent the case of several logical network systems (running different protocol stacks - IP, XNS, SNA, OSI, ...) which coexist on the same physical network. Information related to different types of networks, no matter what the underlying communication protocol is, will reside in the Directory in harmony. Also, their interrelation will be represented and accessed in a fashion independent of the source/destination network, namely, using the OSI X.500 protocol.

2.2.2 Schema for Communication Object Classes

The schemas for the object classes introduced above are defined as follows:

```
CommunicationObject OBJECT CLASS
    SUBCLASS of top
    MAY CONTAIN {
            description :: CaseIgnoreStringSyntax,
                        /* can contain any information about the object,
                        however, whereever an appropiate attribute
                        exists, this should be used first to hold
                        information */
            adminContact :: distinguishedNameSyntax,
                        /* points to the person which is responsible for
                        the administration of the instance this object
                        describes;
                        This refers to the instance only in the context
                        of the concrete object class */
            technContact :: distinguishedNameSyntax,
                        /* points to the person which is responsible for
                        the technical maintanence of the instance this
                        object describes;
                        This refers to the instance only in the context
                        of the concrete object class;
                        Availability (e.g. hours of service) is not
                        covered by this attribute. */
            }


PhysicalCommunicationObject OBJECT CLASS
    SUBCLASS of CommunicationObject
    MAY CONTAIN {
            owner :: distinguishedNameSyntax,
                        /* refers to organization or person owning the
                        physical element;
                        Note that more detailed information (like lease,
                        rental, etc.) can be covered in a specific image
                        (ownerImage) of this element */
            localityName :: CaseIgnoreStringSyntax
                        /* where the object is located;
                        can be used freely to "spot" a network element,
                    e.g. state/city/street/building/floor/room/
                    desk/... */
        ICO :: distinguishedNameSyntax
                    /* points to image object the physical object
                    is related to;
                    might have several values if physical object
                    is use for several applications at the same time */
        }


ImageCommunicationObject OBJECT CLASS
    SUBCLASS of CommunicationObject
    MAY CONTAIN {
        type :: caseIgnoreStringSyntax,
                    /* expresses the view this object referes to, e.g.
                    view of provider/user/IP/OSI/...;
                    Note that this information can be covered by
                    the object class in some cases
                    (e.g. ipNetworkImage gives the IP view) */
        imageOf :: distinguishedNameSyntax,
                    /* points to physical/image object the image
                    is related to;
                    might have several values if view applies to
                    several physical objects at the same time */
        }
```

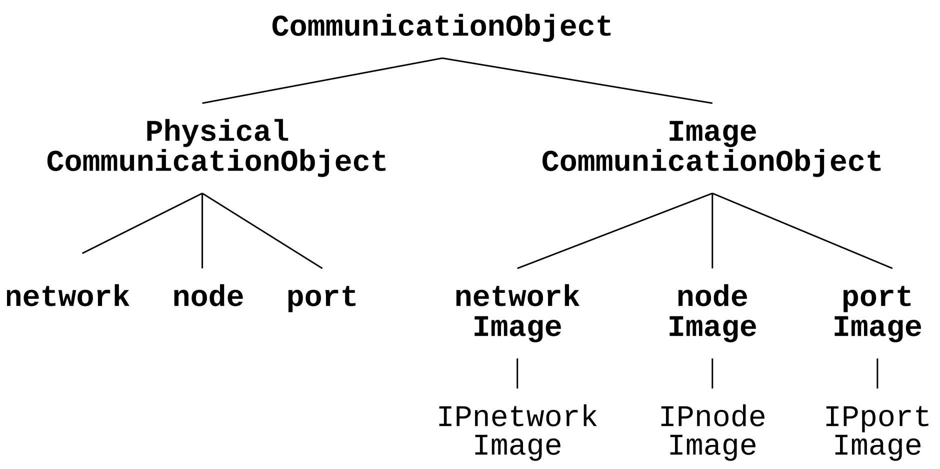


図 2.3: Class inheritance for communication objects

2.2.3 Physical elements

The following objects describe network elements without saying anything about their usage. All objects inherit properties of the PhysicalCommunicationObject class.

Network The network object supplies general descriptions which are common for a set of nodes and circuits comprising one network. This includes information about the type of circuits (medium, broadcast or point-to-point, etc.) and properties (speed etc.).

```
network OBJECT CLASS
    SUBCLASS of PhysicalCommunicationObject
    MUST CONTAIN {
            networkName :: caseIgnoreStringSyntax }
    MAY CONTAIN {
            externalGateway :: distinguishedNameSyntax,
                        /* points to one or more nodes that connect
                        this network to neighbor networks;
                        whether a node actually is used as gateway
                        for one or the other protocol, is defined
                        in a related networkImageObject */
            nwType :: caseIgnoreStringSyntax,
                        /* type of this network;
                        either "composite" (if consiting of subnetworks)
                        or type of a line:
                        bus, ring, star, mesh, point-to-point */
            media :: caseIgnoreStringSyntax,
                        /* if network is not composite,
                        describes physical media:
                        copper, fiber optic, etc. */
            speed :: numericStringSyntax,
                        /* nominal bandwidth, e.g. 64 kbps */
            traffic :: numericStringSyntax
                        /* (average) use in percent of nominal bandwidth
                        [ this needs more specification later ] */
            configurationDate :: uTCTimeSyntax,
                        /* date when network was configured in current
                        shape */
            configurationHistory :: caseIgnoreStringSyntax
                        /* list of configuration changes, like:
                        added/removed nodes, lines */
            }
```

Node The node object describes any kind of device that is part of the network, such as simple nodes, printer, bridges.

```
node OBJECT CLASS
    SUBCLASS of PhysicalCommunicationObject
    MUST CONTAIN {
            nodeName :: caseIgnoreStringSyntax }
    MAY CONTAIN {
            machineType :: caseIgnoreStringSyntax,
                       /* e.g. main frame, work station, PC, printer;
                       might include manufacturer */
            OS :: caseIgnoreStringSyntax,
                       /* e.g. VM, UNIX, DOS; might include release */
            }
```

Port Each node object will have one or more *port* objects as subordinates. Port objects provide information about interfaces of the node and connectivity.

```
port OBJECT CLASS
    SUBCLASS of PhysicalCommunicationObject
    MUST CONTAIN {
            portName :: caseIgnoreStringSyntax }
                       /* It is suggested that the port name
                       is derived from the name of the logical
                       device this port represents for the
                       operating system, e.g. le0, COM1 */
    MAY CONTAIN {
            portAddress :: caseIgnoreStringSyntax,
                       /* this should contain a protocol-independent
                       interface address, e.g. Ethernet board number */
            connectedNetwork :: distinguishedNameSyntax,
                       /* pointer to object of network which this port is
                       connected to */
            }
```

2.2.4 Logical Elements

An abstract view of a physical element is called image in this document. The word image gets appended to the object type, leading to the new objects networkImage, nodeImage and portImage. Images will either build Directory trees of themselves or be stored as part of the physical network tree (see subsection 2.3.9.).

Image objects inherit properties of the ImageCommunicationObject class.

Each image object has specific attributes which vary depending on the point of view (IP, OSI, ...). Also, the user and provider of an image will view an object differently;

further a user of an object may also be providing the services of the same object to another user.

Therefore, in the following a complete and general list of attributes cannot be given. We recommend to define subclasses of Image classes for each logical view. These subclasses inherit all attributes defined with the object classes below and add more specific attributes.

Application to an IP-view are given in 2.3.1.

Network There may be several network images for one and the same physical network: one for each protocol, application, etc.

```
networkImage OBJECT CLASS
    SUBCLASS of ImageCommunicationObject
    MAY CONTAIN {
            externalGateway :: distinguishedNameSyntax,
                    /* points to one or more nodes that act
                    as gateway for the protocol application
                    this images referes to */
            speed :: numericStringSyntax,
                    /* nominal bandwidth for the channel dedicated
                    to this protocol or application ,
                    e.g. 64 kbps */
            traffic :: numericStringSyntax,
                    /* (average) use in percent of nominal bandwidth
                    [ this needs more specification later ] */
            charge :: numericStringSyntax
                    /* amount of money that has to be paid to
                    service provider for usage;
                    [ it is felt that this needs further definition:
                    e.g. monetary unit / time unit, monetary unit /
                    data unit ] */
            }
```

Node Name and functionality within the network might vary for a node from protocol to protocol considered. In particular, a node might act as gateway for one protocol but not for the other. Routing policy is stored in the case of policy gateways.

```
nodeImage OBJECT CLASS
    SUBCLASS of ImageCommunicationObject
            /* no attributes common for all nodeImages have been
            defined yet */
```

Port As with physical nodes, nodeImages have ports (portImages) which describe connectivity to other network elements. PortImages are only given if the protocol is establishing connections via this port.

```
portImage OBJECT CLASS
    SUBCLASS of ImageCommunicationObject
    MAY CONTAIN {
            portAddress :: caseIgnoreStringSyntax,
                    /* this ports address in the context the
                    image referes to, e.g. IP number, NSAP */
            connectedNetwork :: distinguishedNameSyntax
                    /* pointer to networkImageObject that describes
                    the network this port is attached to in terms
                    of the protocol or application the image
                    indicates */
            }
```

2.3 IP networks in the Directory

Information related to the Internet Network Infrastructure is stored and created by a number of different organisations, such as the Network Information Center (NIC), the Internet Assigned Numbers Authority (IANA), and the NSFNET Network Operations Center (NOC). The information is in general "mastered" (stored and maintained) by these organisations on a centralized basis, i.e. there is a single place to look for a definitive list of entries for these categories. This has worked well in the past but given the tremendous growth of the Internet and its number of users and networks, it is essential that a distributed scheme be used.

Applying the general method of storing network infrastructure information in the X.500 Directory (as outlined in 2.1.3) to IP networks, the following goals can be accomplished:

1. Provision of IP specific images of network elements

2. Mapping from Network Number to network, host, owner, etc.

3. Support of delegation of IP address blocks

4. Support of "classless" network address formats

5. Provision of several organizational images of a network

In the following images of network elements for an IP specific view are defined.

2.3.1 IP network image

IP network image is one application of network images and therefore inherits the networkImageClass.

An IP network combines all network elements that have a common IP network number. Presently, IP network numbers fall into one of the classes A, B, or C. However, subnetworking is already done broadly and classless networks numbers are expected to be assigned soon. The definition of IP network, therefore, is always related to common network number and network mask. IP networks have a very close relationship to the Domain Name System. Several attributes are introduced to take care of these relations.

```
ipNetworkImage OBJECT CLASS
    SUBCLASS of NetworkImage
    MUST CONTAIN {
       ipNetworkImageName :: CaseIgnoreString,
              /* common name */
       ipNwNumber :: IPStringSyntax,
              /* the IP network number for
              this (sub)network */
       ipNwMask :: IPStringSyntax
              /* mask that applies to ipNwNumber
              in order to define classless
              network number; also used for routing */
       }
    MAY CONTAIN {
    /* DNS related info ; e.g, - */
       associatedDomain :: CaseIgnoreStringSyntax,
              /* the domain name associated to this IP network;
              As there is no 1:1 mapping between traditionell
              IP numbers and domain names, the name given here
              should contain the "closest match".
              1) (sub)network does not have a domain name
              of its own, but is part of a bigger domain:
              take name of that domain
              2) network is divided into several domains,
              therefore having more than one domain name:
              list all of them.
              Note: a reverse mapping of domain names to
              networks/nodes can be achieved by a modified
              implementation of RFC1279 */
       inAddrServer :: DistinguishedNameSyntax,
              /* referes to the ipNodeImageObject of the
              inaddr Server that holds information about
              this network */
       primaryNameServer : DistinguishedNameSyntax,
              /* referes to the ipNodeImageObject of the
```

```
                primary domain name server for this network */
        secondaryNameServer : DistinguishedNameSyntax,
                /* referes to the ipNodeImageObject of the
                secondary domain name server for this network */
    /* routing policy; e.g, - */
        acceptedUsagePolicy :: caseIgnoreStringSyntax,
                /* semantics to be defined */
    /* Any other - */
        onlineDate :: uTCTimeSyntax
                /* date when network got connected to the Internet */
        }
```

2.3.2 IP node image

If a node in the network is running the IP protocol, an ipNodeImageObject should be created for this node. This image is a subclass of nodeImageClass and holds IP specific information.

The ipNodeImage is defined as follows:

```
ipNodeImage OBJECT CLASS
    SUBCLASS of NodeImage
    MUST CONTAIN {
        ipNodeName :: CaseIgnoreString
                /* common name, it is advised to use
                the hostname for this purpose */
    MAY CONTAIN {
        protocol :: CaseIgnoreString,
                /* name and version of IP protocol running */
        domainName :: CaseIgnoreString,
                /* the complete domain name of this node;
                CNAMEs can be stored additionally to the
                DNS A record name;
                further relationships, like MX record entries,
                should be taken care of by the domain name tree
                according to RFC 1279 */
        }
```

2.3.3 IP port image

The most important IP related information of a node (its IP addresses) are registered with ipPortImageObjects. This picture is more accurate as a node can have several IP addresses, but only one per interface. Furthermore, it shows clearly the relationship of neighbor IP network and connected port.

IpPortImage is a subclass of portImageClass. Note that for ipPortImage all references are drawn in the context of IP, i.e. portAddress becomes an IP address, connectedLine and connectedNetwork point to ipImageObjects.

Additionally to portImageClass, ipPortImageObject has the following properties:

```
ipPortImage OBJECT CLASS
    SUBCLASS of PortImage
    MUST CONTAIN {
            ipPortName :: CaseIgnoreStringSyntax
                       /* It is suggested that the port name
                       is derived from the name of the logical
                       device this port represents for the
                       operating system, e.g. le0, COM1 */
    MAY CONTAIN {
            ipNwMask :: IPStringSyntax
                       /* mask that applies to portAddress for
                       routing of packets to nodes on the connected
                       (broadcast) network;
                       Note: This is only a fraction
                       of the routing table information
                       for this port, namely for one hop. */
            }
```

2.3.4 IP namespace objects

The following Directory objects have been defined to represent the IP namespace (in contradiction to network elements using IP) in the Directory. Their purpose is to provide

1. mapping from IP number to IP network element (network or node),
2. assignment and delegation information.

Describing all IP numbers with one of the new objects delegatedBlock, ipGroup and ipReference leads to the desired information. Furthermore, all assigned numbers have some properties in common. Therefore, the objectclass assignedNumberClass is introduced. This class exports attributes to delegatedBlock, ipGroup and ipReference.

AssignedNumberClass is defined as follows ("number" always refers to IP number of delegatedBlock, network, host):

```
assignedNumberClass OBJECT CLASS
    SUBCLASS of top
    MAY CONTAIN {
            assBy :: DistinguishedNameSyntax,
                /* referes to organization or organizationalRole
                that assigned the number to assTo (see below) */
assTo :: DistinguishedNameSyntax,
                /* referes to organization or organizationalRole
                that the number was assigned to. This does not
                imply that assTo owns this number now. */
assDate :: uTCTimeSyntax,
                /* date of assignment for this number */
relNwElement :: DistinguishedNameSyntax,
                /* the network element related to this number
                (network or node) */
}
```

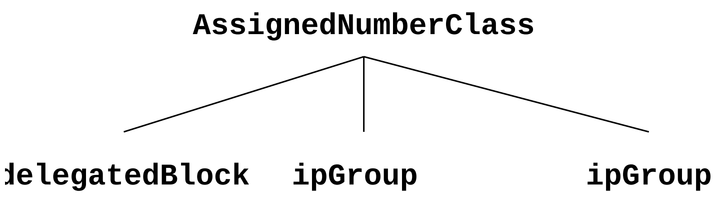


図 2.4: Class inherition for assignment objects

2.3.5 Delegated Block object

This object provides information on a block of IP addresses delegated to some local-authority or service provider. Only contiguous blocks can be represented with the following scheme. If an organization (say, a NIC) has been assigned several IP network numbers which do not form a contiguous block, it might want to use a different form of representing that fact (e.g. using imageNetworks). The delegatedBlock object holds lower and upper bounds of the block.

Note that the above does not make any assumption about the network masks being constrained by byte boundaries. It can thus represent in the same framework the subnetting within a "network (number)" that often happens within an organization.

This scheme does lead to some granularity in the otherwise flat IP-number space. Further, the granularity is significant as it may be used to identify the administrator of the block - a service provider or a domain manager. E.g. it fits well into the scheme of aggregating networks for routing purposes as has been proposed in [50].

The object delegatedBlock is of the form:

```
delegatedBlock OBJECT CLASS
```

```
    SUBCLASS of AssignedNumberClass
    MUST CONTAIN {
       lowerBound :: IPStringSyntax,
             /* smallest IP address belonging to the
             block, e.g. 195.100.0.0 */
       upperBound :: IPStringSyntax
             /* highest IP address belonging to the
             block, e.g. 195.103.255.255 */
       }
```

The attribute relNwElement (inherited from AssignedNumberClass) can point to a networkImage covering all networks within the block.

2.3.6 IP Group object

This object provides information for an IP network number. Similar to delegatedBlock, an IP group summarizes a block of IP numbers. However, in this case the connection to "real" IP networks is emphasized. If an organisation has a conventional class-B-network number, then the IP group object for this class-B number will refer to this organisation's network. Regardless of the actual value of x, IP group objects may exist for IP numbers x.0.0.0, x.y.0.0 and x.y.z.0. This approach includes "classical" class-A, -B and -C network addresses as well as subnetworking for classes A and B.

The IP group object is a subclass of assignedNumberClass. The attribute relNwElement points to an ipNetworkImage.

```
ipGroup OBJECT CLASS
    SUBCLASS of AssignedNumberClass
    MUST CONTAIN {
       ipGroupName : IPStringSyntax,
             /* common name; x.0.0.0 or x.y.0.0 or x.y.z.0
             where x, y, z in {1..255} */
       ipNwMask :: IPStringSyntax
             /* mask that applies to all numbers
             within the group; used to define
             classless networking; */
       }
```

2.3.7 IP Reference object

There is one IP reference object for each IP address. The purpose of this object is to

- tell that this IP number is already assigned to a node

- give a pointer to a ipNodeImageObject

The IP reference object is a subclass of assignedNumberClass. The attribute relNwElement points to an ipNodeImage.

```
ipRef OBJECT CLASS
    SUBCLASS of AssignedNumberClass
    MUST CONTAIN }
        ipReferenceName : IPString
                /* common name; value is always
                the IP address */
        }
```

2.3.8 IPString Syntax

A new attribute Syntax has been introduced for several attributes related to IP numbers. This syntax is defined below:

```
ipStringSyntax ATTRIBUTE SYNTAX
    SEQUENCE OF
    {   BYTE,
        SEPARATOR,
        BYTE,
        SEPARATOR,
        BYTE,
        SEPARATOR,
        BYTE
    }
    MATCHES FOR EQUALITY ORDERING
```

with:

```
    SEPARATOR   := '.'
    BYTE        := [0..255]
```

This syntax is used for IP number matching. More important than equality matching (which could have been done with CaseIgnoreString as well) is ordering IP numbers. More-less-relations have to be resolved in searches correctly.

As an example, let
IP1 := 1.2.0.0
IP2 := 1.10.30.0
IP3 := 1.4.255.8.

The following inequality has to be true:

$IP1 < IP3 < IP2$

As a preliminary approach, implemetations can use CaseIgnoreStringSyntax if BYTE is always expressed in 3 digits, e.g.

IP1 := 001.002.000.000
IP2 := 001.010.030.000
IP3 := 001.004.255.008

will give the relation correctly even with CaseIgnoreStringSyntax rules.

2.3.9 Position of new objects in the DIT

It has been mentioned several times before that the X.500 Directory is organised in a tree structure. Some rules, known as the Directory schema, prevent entries from being 'misplaced' in the tree. These rules define what object classes can build subordinate entries to other object classes. For example, organizationalUnit can be subordinate to organization but not the other way around.

With the definition of new object classes the need to define a proper position for them within the Directory Information Tree (DIT) arose. A proposal is outlined below.

2.3.10 Networks in the DIT

Information about networks usually will be contained in the DIT as subordinate of the organisation maintaining the network. The network model gets mapped into a tree structure for network elements. There is one *network* object giving general descriptions of the network. Subordinates of this network object are *node* objects for each node element present in the network. Node objects hold *port* objects as subordinates. A network can be physically or logically subdivided into several (sub)networks. In this case, a network entry will have network objects as subordinates which again build the same structure. These entries may be kept as subordinates of organizationalUnit entries as well, with pointers from the "root" network.

This structure holds for physical and logical elements. Physical elements are named network, node and port, and logical elements are named networkImage, nodeImage and portImage.

2.3.11 IP namespace in the DIT

The new objects introduced in subsection 2.3.4 build a single tree in the Directory. It is suggested that this tree will have a root of type organizationalUnit within @o=Internet.

```
 objectClass= organizationalUnit
organizationalUnitName= IP networks
description= root of IP number tree
```

The tree is built under an administrative and an implementational view. Nowadays, network numbers usually are assigned to organizations by (national) Network Information Centers (NIC) which themselves have got a block of IP network numbers assigned from another authority (e.g. IANA at top level). This concept of delegated blocks falling apart in smaller delegated blocks and IP network numbers is used to model the Directory tree. Thus, an ipGroup object is always subordinate of a delegated block object (namely the delegated block including this IP number).

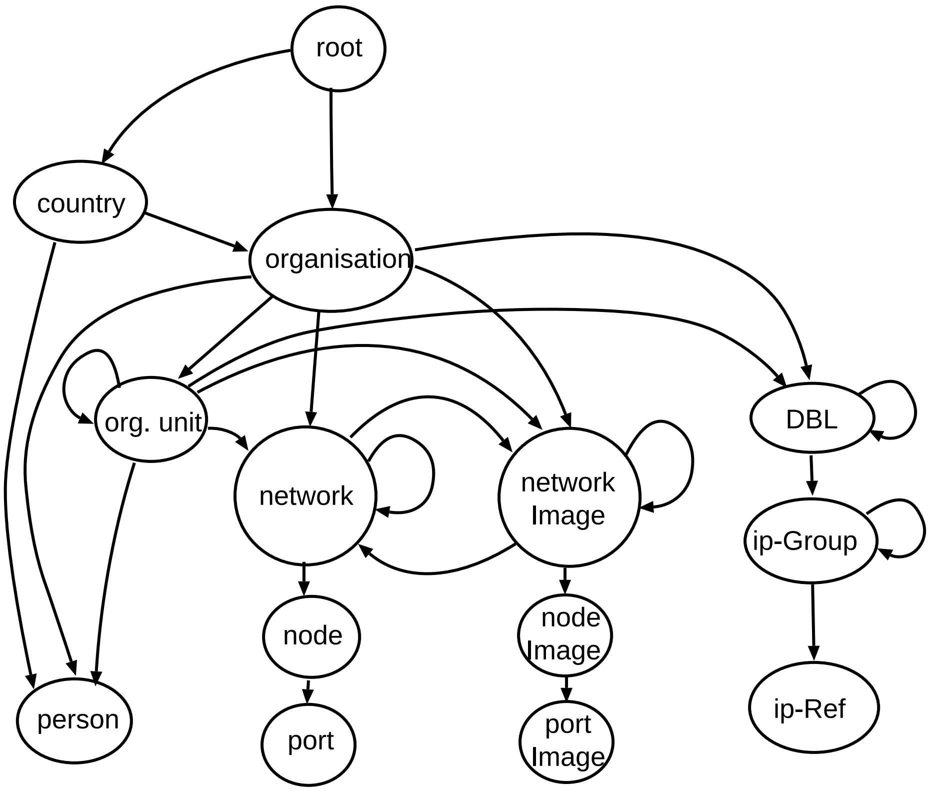


図 2.5: Part of the DIT with network and IP namespace objects

Network numbers that were directly assigned by a top-level authority, i.e. have not been object of a delegation to a local assigning authority, will all be at one level in the Directory. Already today, however, we find many delegations within the traditional class A-, B- and C-addresses. Such a delegation is represented by a delegated block object, having the assigned IP network numbers as subordinates. Also, part of the block can be further delegated to another authority, leading to another delegated block object within the parent delegated block's tree. Usually, subordinates of ipGroup objects are ipReferences, i.e. single IP addresses as assigned to nodes.

To support subnetworking, it is also allowed to divid ipGroups into several subnetwork ipGroups, representing each an IP subnetwork. In such a case, subnetwork numbers are given as subordinates to the assigned IP network number. Network masks clarify what the subnetwork addresses.

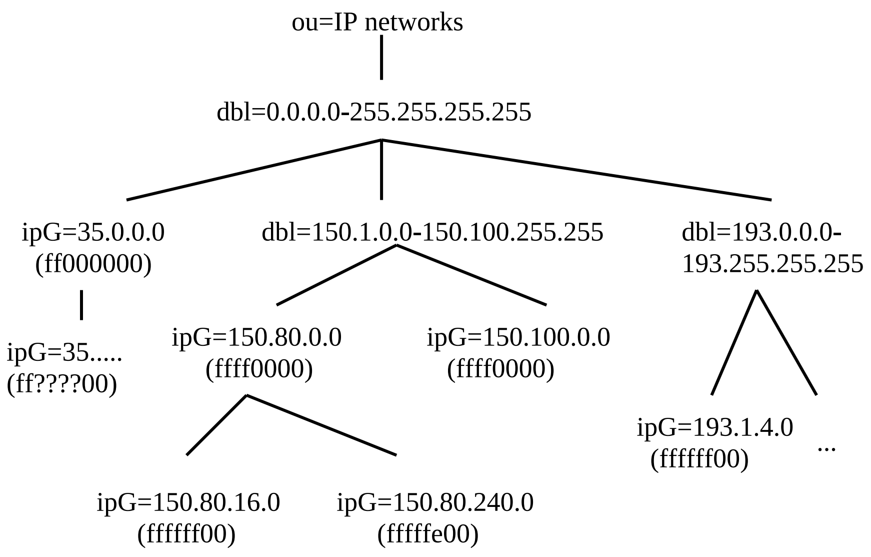

ou=IP networks
dbl=0.0.0.0-255.255.255.255
ipG=35.0.0.0 (ff000000)
dbl=150.1.0.0-150.100.255.255
dbl=193.0.0.0-193.255.255.255
ipG=35..... (ff????00)
ipG=150.80.0.0 (ffff0000)
ipG=150.100.0.0 (ffff0000)
ipG=193.1.4.0 (ffffff00)
...
ipG=150.80.16.0 (ffffff00)
ipG=150.80.240.0 (fffffe00)

図 2.6: Example population of IP namespace tree according to delegation and subnetworking.

2.4 Bootstrapping, Operation & Maintenance

The startup, operation and maintenance of the proposed directory services is a very complex task and deserves careful consideration. In the following, some of the strategies adopted are described.

2.4.1 Bootstrapping

Populating the directory is possibly one of the most challenging tasks. It has become more difficult as the network services are already registered and in place. Asking for additional information about network configuration etc. is not a very pragmatic solution. However, fortunately enough one does not have to start from the scratch in gathering information. There are already pools of data available e.g. the network registry, various NICs, the DNS ... . Tools were developed/employed to make it easier for administrators to cooperate in the task of populating the directory. e.g. configuration information was extracted, on a smaller/local scale using discovery mechanisms [51], whitepages information was extracted from existing user registration databases.

2.4.2 Operation & Maintenance

The distributed directory poses some special problems due its nature which distributes control and maintenance. The various parts of the directory which are under separate administration do interact. Moreover even within the same administrative domain the different components do interact. For example the "network manager" attribute of a network in the network component of the directory will in general point to a "person" entry in the whitepages component. The "manager Person" may relinquished hia/her job at the organization leaving the "network manager" pointer dangling in the air. Tools are necessary to periodically verify the integrity of the database. And this is necessarily a cooperative task which essentially depends on collaborative effort.

The aspect of *reliability* is another problem area. Some providers donot quite like the idea of having to rely on the services of another DSA to provide it with information. This leads to a tendency of information duplication that runs against the principle of the design and adds to the problems of maintaining data integrity. The compromise in this aspect has been struck by duplicating the minimal set of information at the preferred DSA and periodically checking the consistency/correctness of the duplicated data.

2.5 Applications

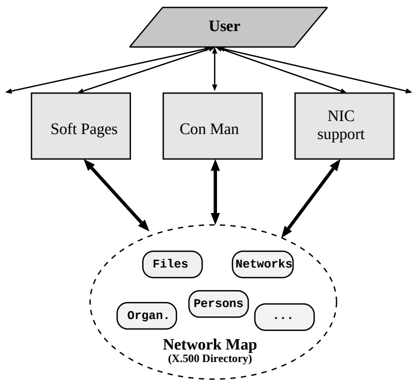


FIG. 4

As can be expected a whole range of applications are coming up based on the infrastructural information provided. The *annotated network map* is finding diverse applications ranging from finding documents from their *cheapest* source to suggesting alternate paths in case of problems and also replying queries about managers/point-of- contacts of networks. Some of the projects are briefly described below.

2.5.1 SoftPages[SPP]

The Soft Pages Project [52] started with the wish to reduce ftp traffic on crowded overseas links and national backbones. The basic idea is to query for files always on filestores that are nearest/cheapest to the users site. For that purpose, it is necessary to evaluate distance/cost between the users host and possible servers. Also, it is necessary to search the contents of the servers.

SoftPages makes use of the network infrastructure information concerning network configuration, location of servers and their contents, the links and their charges, ... as is described in this document. SoftPages has in fact been the test bed for the ideas in this scheme. A detailed discussion of the SoftPages Project is given in chapter 3

2.5.2 NIC services

The proposed scheme supports the framework to provide information similar to the "whois" database of Network Information centers (NICs). It is obvious that the service based on X.500 is much more powerful due to the global and distributed database and the connection to White Pages information.

Presently, an experimental user agent using an e-mail interface based on a development by University of Michigan is operational. Upon a request for a person, an organization, a network, a node or an IP number, the answer is sent back by e-mail. This mail responder is using the extended X.500 database of our pilot Directory to resolve queries. For implementation details please refer to section 4.1.2.2.

2.5.3 ConMan

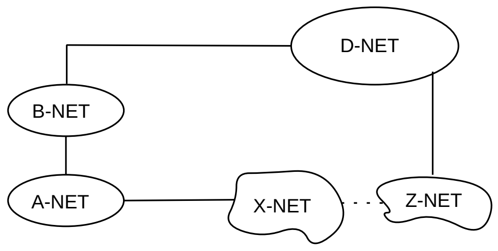


FIG. 6

Given the mesh-type connectivity of networks, it is very important to have a clear picture about the connections; for example, A-net has potential conectivity to D-net via multiple routes. The topological map of the network is useful; to find alternate paths in case of failure, to know the transit policy of the provider on the alternate path, to know the point of contact for the provider. Needless to say the map can be used to detect non-otimal routes and to evaluate the quality of the connectivity in terms of redundancy.

A project for Configuration Management [53] involving route,traffic, policy, .. management, using the proposed framework,is presently ongoing, as a collaborative effort among AIC, Tohoku University and the WIDE Internet, Japan. The global map for the

Japan Internet has been generated. The lower level details for organizational networks have been filled in depending on availability. The primary source of information was the NICs followed by network administrators. The project essentially envisages "intelligent management" where network configuration information supplements other management information. The maps are displayed for design/administration/planning. It also answers queries about contact persons /administrators, and suggests bypass routes in case of problems.

第 3 章

The SoftPages Project

3.1 Introduction

There are a huge number of filestores in the Internet. Sharing software, documents, and other kind of information by copying files from and to these filestores is a major application of the network. Some studies reveal that ftp data packets take as much as 75 % of the amount of data being transferred through the Internet.

We believe this huge amount of traffic can be reduced when users are supported with better tools taking care of network configuration.

The Soft Pages Project consists of two parts, which are both needed to provide help for efficient and network optimized file retrieval. One part is the storage and use of network connection properties and the other is the representation of contents of fileservers in the Directory.

3.1.1 Locating files

Filestores in the context of this paper are databases for files, regardless what these files contain and whether a set of files has a special meaning for itself or not. Most of these filestores actually are anonymous ftp servers which can be accessed by everyone using the ftp protocol.

In order to obtain a certain file, users have to know its name, path in a filestore and the network address of this filestore. There are several tools helping users to locate files, some of them being explaned below. An additional method - the representation of filenames in the X.500 Directory - is presented in a later subsection of this paper.

Most anonymous ftp-servers hold a summary of the directory structure in the form of a "ls-lR" text file. A filename can be found by locally searching or browsing this file (after it is retrieved from the server). Once the location of the file within the fileserver's directory structure is known the actual retrieval (ftp) can be started. If someone does not know whether the desired file is stored at a fileserver at all, he/she has to copy ls-lR files from several fileservers and search them until the filename is found, eventually.

That method is very costly and inefficient with respect to time and bandwidth. A step toward improving the behaviour was to hold ls-lR files from several fileservers at "central" servers. By that way, users could access more information within one ftp-session and learnt about the most important fileservers of a country, for example.

Archie does exactly this summarizing of fileserver contents info and provides an user interface for searches [54]. There are a couple of archie servers which each maintains a centralized database for registered ftp sites. The problem is that not all ftp sites are registered with every archie server. Thus, no global database exists. Furthermore, archie can not answer the connection property question.

Another concept of maintaining centralized databases is WAIS [55]. Each database can be searched for keywords and document retrieval is integrated. While searching capabilities are more powerful than that of archie (archie can not search file contents, only file names), queries are restricted to documents/files stored at the specified WAIS server. There is no interaction between servers and databases. Recently, a new WAIS database went into operation containing ftp site information. While this "readmes"-database is supposed to hold global information it still remains on the user to decide which ftp site to connect to.

The World Wide Web concept [56] hides document location and retrieval method completley from the user. However, search remains restricted to registered documents.

X.500 Directory can provide a non-centralized, global database for contents of anonymous ftp sites and other documents. Files are represented by a handle in the Directory. While file contents are not stored in the Directory there is no problem in holding additional description for each file.

3.1.2 Optimizing file retrieval

A lot of files exist in several copies, i.e. on several filestores throughout the world. That leads to a variety of retrieval possibilities for one and the same file. Some file discovery tools will list several sources to retrieve a file with a certain filename from.

As a result of the fascinating global reachability we have today, files principally can be picked up nearly everywhere and transferred to the user's machine. However, global reachability does not mean that there are no differences between connections. As a matter of fact, some international trunk routes are very crowded and cause a considerable decrease of transfer rate. As a result file retrieval will show different behaviour for different locations of contacted sites. Common sense tells us to keep network load as low as possible especially for exposed lines.

While there is no choice to alter use of trunk lines for remote login on dedicated servers, we generally do have the choice to retrieve a document from one of several filestore sites. The problem of finding information thus is extended to finding the particular file on a site which has the best retrieval characteristics.

As far as we know, no approach has been mentioned to offer an integrated solution to the choice problem, yet. Our proposal will be outlined in subsection 3.2.

In order to provide an integrated solution for filename-lookups, the need for a global file directory was felt. As outlined in subsection 3.3, such a directory should be distributed for management reasons. A very simple implementation based on X.500 is proposed in the remainder.

3.2 Using network properties

Proceeding on the assumption that retrieval of a file is possible from several sources, we want to reduce network traffic by a sensible selection of the filestore which actually will be used for retrieval. There are many criteria speaking for or against a particular choice among a given set of alternate servers and users should be guided in the selection process.

The basic idea behind SoftPages is to evaluate fileservers from the user's point of view, taking care of technical and political network parameters. To simplify comparison, each fileserver (i.e. the connection between the user's site and the fileserver's site) is attached a related cost index. Connections with a low cost index will be favored for file transfers. The calculation of this cost index is explained below.

3.2.1 Network configuration information

In order to compare network connections we need a database providing us with the necessary information about network links and their properties. Pieces of this information exist here and there, but there is no comprehensive and global database covering Internet links available to the public yet.

A method to store network configuration information in the X.500 Directory is presented in 2.1.3. We hope to see this method employed in the near future and base further discussion on it.

In the proposal mentioned above, the global network is modelled as a number of smaller networks interconnected by gateways. Traffic going from one host to another has to pass at least one network. In case the hosts do not belong to the same network, one or more gateways and possibly transit networks will be involved. Following the route data packets take, we can name the networks that are passed.

For each of these networks related attributes can be obtained from the database. Attributes contain, but are not limited to, speed, average traffic, charge, and accepted usage policy.

3.2.2 Cost calculation

The Internet community is still living in a time where network connections are free for the single user in most cases. Economically speaking, response time is in most cases the only criteria for remote login, and so is transfer time for file transfer from remote sites. Therefore, users are basically only concerned about connection properties when they realize a long delay. We believe that this will change, however, when the Internet becomes more and more commercial and real accounting starts.

In both cases, awareness for the way network packets take will help reducing time and/or money expenses. But not only users are concerned about network connections. System administrators try to keep traffic on dedicated external links of their sites as low as possible to avoid congestion for important applications.

Usually users cannot choose between different ways if they open a network connection between host A and host B. System administrators might have some control about outgoing routes, but cannot influence much either.

So while this cannot be changed, we are in a better position concerning anonymous filetransfer. Users in most cases have the choice connecting to host B or C or D when looking for a certtain file. With the cost calculation given below, a help is provided for users and system administrators to take a reasonable choice.

For easy comparison, we calculate a single parameter called cost from all network properties of concern for each connection. The higher this cost parameter (or cost index) is, the more "expensive" it is to use the connection. Expensive in this case does not necesarily mean that one has to pay a lot of money, it rather expresses in an abstract way expenses for resources like bandwidth, time, etc.

These resources will be of different value for different users. Therefore it seems to be difficult if not impossible to find a formula to calculate **the** cost index for everybody and all purposes.

We are aware that the following is a very first approach only. Experiments will show what other input parameters are of importance and how these parameters should be weighted against each other.

Parameter values are read from the network database and represent:

speed - the bandwidth (in kbps) theoretically available in a network

traffic - the average use (as percentage) of this network, thus telling something about congestion

charge - monetary units (as abstract integer number) to be paid for the transmission of one packet; this should be used to express relationship between several charges rather than absolut amount of money (not to speak of currency, exchange rates, etc.)

priority - sometimes system administrators want to keep certain connections free for important traffic like mails. They can do this by increasing the priority (integer value) of the network connection in comparison to another.

Cost is calculated as follows:

$$cost = f(speed, traffic, charge, priority)$$
$$cost = a * (1/speed) + b * traffic + c * charge + d * priority$$

whereas the weights a, b, c, d can be chosen freely by the administrator of a site. This way an individual evaluation is possible. For some sites speed has a higher weight than charge, for others it may be the other way around.

With the formula given above, a cost index can be calculated for one network connection. If traffic has to pass several networks, their cost indexes will be added to an overall cost index for the end-to-end connection. Thus, the number of network hops goes into the cost index, too.

$$cost_{end-to-end} = \sum_{i=1}^{n} cost_i$$

Now, if a user at site A has the choice to connect to one of the sites B, C, and D, the cost indexes cost(A-B), cost(A-C), cost(A-D) easily can be compared and the site with the lowest cost index chosen.

3.3 Representing Documents

For anonymous filetransfer, the above mentioned strategy to select low-cost fileserver sites to connect to can be used in combination with a variety of file-discovery tools. A possible scenario is the combination of archie ([54]) and cost-based site selection. Archie

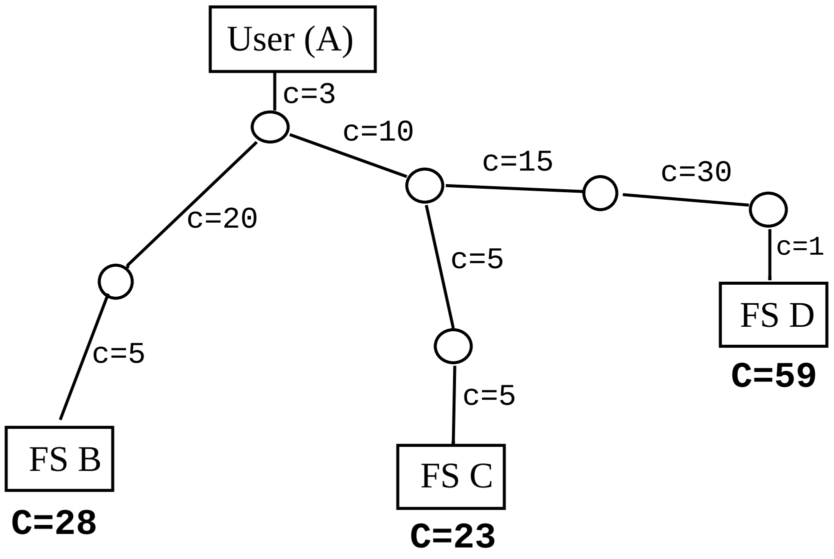


図 3.1: Sample network with costs

returns the names of all fileservers holding a copy of the file wanted. One could process these fileserver names in order to determine to best. However, archie as a centralized database has some disadvantages, like data maintanence and scalability.

Lots of work has been done by several groups ([57], [58], [59], [60]) to represent documents in the X.500 Directory. The main advantages of this approach are distributed management of data and no limitation for the number of document repositories that are registered. File repositories and files can be considered as a special kind of document repositories and documents, respectively, in a broader sense. Therefore, considerations done for documents can be used for files and their repositories as discussed below.

We decided to use the Directory for file-discovery for several reasons.

- Distributed management of data will become more and more important as the network is further growing. File representation in the Directory is based on and encourages distributed management.
- Holding network connectivity and file repository data in one database (namely the X.500 Directory database) allows building a homogenous user agent with DUA functionality.
- Populating the Directory tree with fileserver information will promote X.500 usage for other purposes too, thus encouraging more data input and better mainanence.
- To our knowledge no large-scale implementation has been done yet (apart from a small subsection under @o=Internet@ou=FTP archives). Therefore gathering

operational experience was a goal.

There is no standardized set of information attributes which should be provided for a document yet. Libraries use title, author, publisher, keywords, and so on for books and the like as search index. The need for universal document identifiers has been addressed in [61].

However, in the special case of "everyday ftp" search functions using keywords are not provided and users are supposed to know what they are looking for. There is no doubt that information description would be useful but this is beyond the scope of this paper. Our starting point is that many users already know the name and/or directory of the file they want to copy. They only need the address of the ftp site holding the file and a complete pathname.

Being aware of more complex data structures being modelled, we intentionally choose a very simple definition for filestores and files to get things started. These definitions are likely to become part of a more comprehensive definition for documents later.

Two new object classes are defined for fileserver and file, respectively:

```
fileServer OBJECT CLASS
   SUBCLASS of FileStoreObject
   MUST CONTAIN
         {
           fileServerName : CaseIgnoreString
                – e.g. hostname
         }
   MAY CONTAIN
         {
           fsAddress : IA5String,
                – within the addressing scheme, e.g. IP number
           fsUserid : CaseIgnoreString,
                – login ID for anonymous access
           fsUserPassword : CaseIgnoreString,
                – password for anonymous access
           fsManager : DN
                – reference to Directory entry of
                – maintainer of fileserver
         }


file OBJECT CLASS
   SUBCLASS of FileStoreObject
   MUST CONTAIN
         {
           fileName : CaseExactString
```

```
                  – complete filename as registered with fileserver
            }
    MAY CONTAIN
            {
              fileInformation : CaseIgnoreString,
                  – additional information about file like:
                  – size, date, owner, access rights
                  – e.g. in ls -l format
              fileType : CaseIgnoreString
                  – identifier for type of file,
                  – either file suffix (.txt, .com, etc.)
                  – or descriptive (tex, ascii, binary, etc.)
            }
```

Both object classes are subclasses of FileStoreObject, which is used to inherit common attributes and to put things in a better way in the DIT.

```
FileStoreObject OBJECT CLASS
    SUBCLASS of top
    MAY CONTAIN
            {
            description,
            }
```

3.4 Information Processing

So far SoftPages components have been described. The databases for network configuration and file repositories provide all necessary information for cost-sensitive file retrieval. However, a dedicated user agent (Fig. 3.2) should be built hiding the necessary processing.

In principle, processing should go along this road:

First, a set of fileservers is chosen. This set contains all servers that will be considered for a filename search. It is possible to generate such a set from the Directory by searching within a certain scope (e.g. "all file servers in Japan"). By this way, all fileservers within a certain region can be covered as these will most likely be the ones with the lowest cost index. But an user can also generate his own set of fileservers and thus include some remote sites that might resolve his questions. Theoretically it would possible to include "all fileservers in the world" in that list. Information about the fileserver set could be held in a text file, thereby easily to be edited by the user.

Next, according to the connection parameter preference the user has set (speed, charge, etc.) all fileservers of the above mentioned set are evaluated. This evaluation for cost

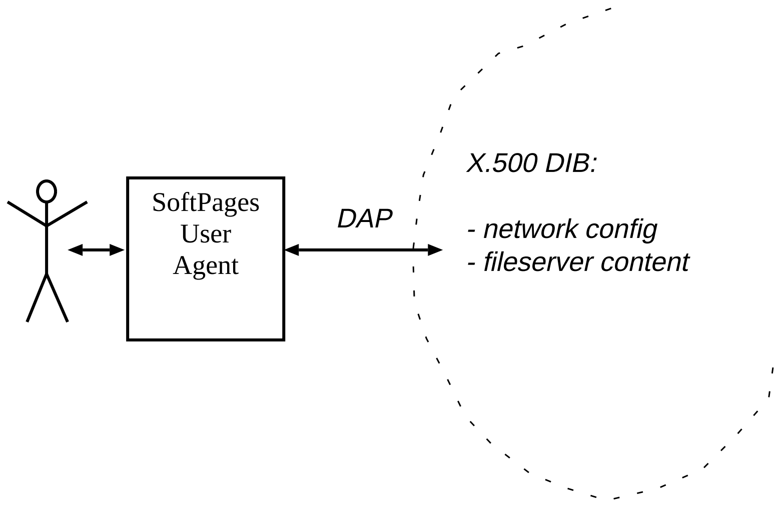


図 3.2: SoftPages user agent

index will create a sorted list of fileservers. Generelly, Directory file enquiries go along this sorted list, starting with the fileserver that has the lowest cost index. If no file name match can be found, the search goes on to the next fileserver (with the next higher cost index). This process will continue until a file matching the search pattern has been found or information for all fileservers has been scanned unsuccessfully.

第 4 章

Directory mini-pilot for Network Management

4.1 Experimental set-up

The proposals and ideas presented in the previous chapters are related to the X.500 Directory. Consequently, experiments with the Directory have been carried out in order to prove the applicability of our proposals. The X.500 Directory consists of Directory System Agents (DSAs) holding the database and Directory User Agents (DUAs) querying the database by binding to DSAs. As we have extended the set of object class and attributes type definitions beyond a predefined set, changes had to be made in DSAs as well as DUAs.

4.1.1 DSAs

It was decided that experiments should not be carried out on Directory System Agents (DSA) that are connected to the global Directory. Therefore, a mini-pilot Directory "world" had to be set up. This would have been possible with one DSA only. To gather operational experience with a distributed Directory, however, our pilot includes three DSAs.

While DSA Sazae was configured as root master and master for the country Japan, the other two DSAs look very much like real level-2-DSAs. This was done to be prepared for an easy connection of these level-2-DSAs to the global Directory at a later stage of the experiment.

All DSAs run with an extended OID-table. This table includes objects and attributes as defined in the theoretical chapter.

Operational basis for all experiments are UNIX workstations (Sun Sparc 4/2 with SunOS 4.1.1/4.1.2) which have to do more jobs than just running the DSAs.

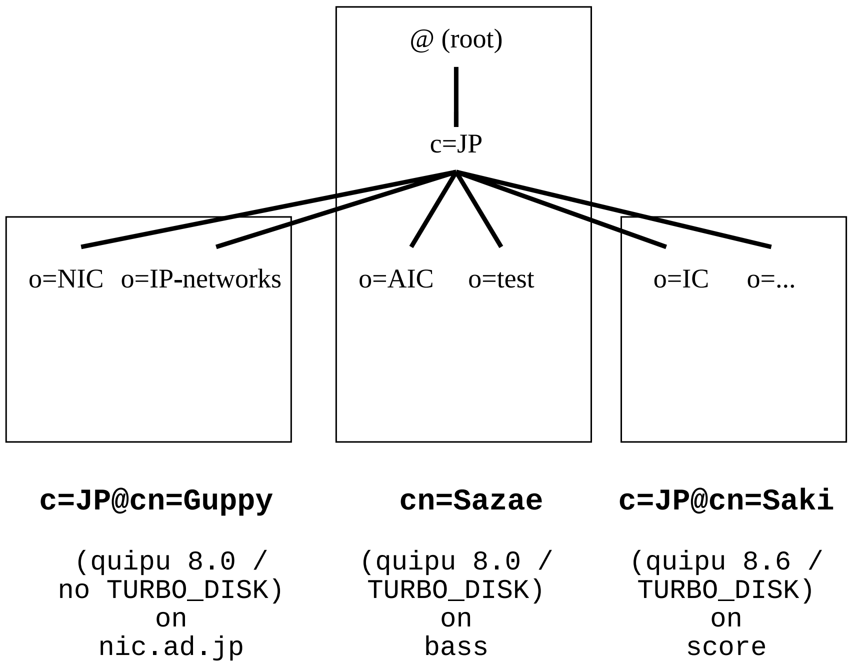

@ (root)
c=JP
o=NIC
o=IP-networks
o=AIC
o=test
o=IC
o=...
c=JP@cn=Guppy
(quipu 8.0 / no TURBO_DISK) on nic.ad.jp
cn=Sazae
(quipu 8.0 / TURBO_DISK) on bass
c=JP@cn=Saki
(quipu 8.6 / TURBO_DISK) on score

図 4.1: Pilot Directory configuration

4.1.2 DUAs

4.1.2.1 Feed of NIC data to X.500

One of the objectives for representing network information in the Directory was supporting NICs in administering IP network information. To get the project started with real life data, we built a filter converting present whois records to newly defined X.500 objects. The translation used by that filter is shown in tables 4.1 and 4.2.

whois	X.500
Network Information	ipNetworkImage
Domain Information	domainComponent
Personal Information	pilotPerson
(none)	organization
(none)	organizationalUnit
(none)	ipGroup

表 4.1: whois records → X.500 objects

It turned out that this translation could not fully be automated. A reason was inconsistent entries in the whois database for organizations and divisions. In many cases slighty

whois	X.500
Network Name	ipNwName
Network Number	ipNwNumber
Domain Name	domainName
Technical Contact	technContact
Administrative Contact	adminContact
Description	description
Organisation	organizationName
Division	organizationalUnitName
Address	postalAddress
TEL	telephoneNumber
FAX	facsimileNumber
E-Mail	mail
Last, First	surname / commonName
Internet Nameserver	secondaryNameserver
Domestic Nameserver	primaryNameserver

表 4.2: whois attributes → X.500 attributes

different names were used in different records to refer to one and the same object. This had to be fixed manually in order to build a White Pages Directory tree for all entries of `organizations`, `organizationalUnits`, and `persons`. It was necessary to create many new objects of the mentioned classes, as the Directory tree for Japan was not yet populated to this extent. There was, however, no problem adding the newly defined network related objects in the Directory.

Networks in Japan have been assigned 22 delegated blocks and many more single network numbers. First we entered all delegated blocks, as this kind of information helps the NIC most (It actually turned out that nobody there really could tell us what blocks were assigned to Japan, thus showing that there is no database holding this information now.). From the total of 1993 registered IP networks (i.e. class-B and class-C network numbers), we selected a part for experimental transformation and addition. To go with the concept of delegated blocks, we choose IP numbers from the block 133.1.0.0-133.254.0.0 for addition. The numbers of X.500 objects resulting from this addition are given in table 4.3.

While `organization`, `organizationalUnit`, `pilotPerson` , and `IPnetworks` were added to the White Pages namespace tree (below `@c=JP` and `@c=JP@o=...`, resp.), `delegatedBlocks` and `IPgroups` went to one subtree named `@c=JP@o=IP networks` for the experiment. This subtree is supposed to be shifted to

X.500 object	number
organization	194
organizationalUnit	247
pilotPerson	429
IPnetwork	228
IPgroup	228

表 4.3: Number of converted entries for IP networks 133.1-133.254

`@o=Internet@ou=IP networks` later.

4.1.2.2 X.500 based whois responder

We wanted to provide an easy-to-use interface to data in our pilot X.500 Directory. As long as the pilot is not connected to the global Directory space, users explicity have to know the presentation address of one of the pilot DSAs. Usually, every DUA binds to a default (near) DSA, queries are sent to other DSAs from there, and users do not even have to know what a presentation address is. For similar simplicity in using our data, we decided to modify and use an e-mail-responder. Such a responder has been developed at University of Michigan and code is freely available. When e-mail is sent to a dedicated address, a response program is invoked. This program processes the input e-mail (looking for keywords specifying the operation wanted to be done by the user), acts as DUA when sending the operation to a DSA, and pastes the result from the DSA into an answer e-mail to the originator.

Furthermore, some processing can be done in that responder program. A typical case is resolving of X.500 references. When a user asks for information about a network, the DSA will return only the object for this network, containing references to administrative persons rather than detailed information about these persons. The responder program, in turn, can take these references and ask the DSA to send the specified objects. Thus, the user will get back not only information about the network itself, but also about closely related objects, like managers, organisation the network belongs to, etc.

To use the e-mail responder for experimental X.500 network information, send an e-mail to **x500-query@aic-wide.aic.co.jp** with the word `help` in `Subject:` or body of the mail. Figures 4.2 to 4.6 show a few examples of queries answered by this e-mail responder.

```
From: Directory Query Process <X500-Query@aic-wide.aic.co.jp>
Subject: you asked "help"

Welcome to the Experimental X.500 Email Network Query Service!

How to use this service:

By sending electronic mail to the address:

        x500-query@aic-wide.aic.co.jp

you can access the Network X.500 Directory.  This Directory contains
information about most networks in Japan, their addresses, contact
persons, file server repositories, and more.

Please note: The experimental Directory is not yet connected to the global
Directory.  Therefore you cannot retrieve information about countries or
entries which might be defined in the global Directory but are missing in
this experimental Directory.

To query the service, send a piece of email to this address with one
of the following commands in the 'Subject:' field or mail-body:

help
find person       xxx
find org          xxx
find organization xxx
find nw           xxx
find network      xxx
find IP           xxx
find handle       xxx
list nw           xxx
list network      xxx

In all cases xxx should be replaced with the name or identifier of the
object you are looking for.

The 'find' command will show you all objects that match the string
supplied by you with respect to the objectclass (i.e. person,
organization, network, etc.).

The 'list' command rather gives a view of all elements that belong to a
certain hierarchy. For example a 'list network' command returns all
elements that belong to the network in question.

You can use wildcards ('*') when specifying the object identifier. If you
do not use wildcards, an exact search and an approximate search are
carried out.
```

図 4.2: X.500 e-mail responder: help message

```
From: Directory Query Process <X500-Query@aic-wide.aic.co.jp>
Subject: you asked "find ip 150.80"

Resolution of IP number:

 Delegated Block: 150.001.000.000-150.100.255.255
   assigned from (X.500 handle): (null)
   assigned to (X.500 handle): o=NIC, c=JP

 IP network address: 150.080.000.000
   IP network mask: ffff0000
   assigned to (X.500 handle): o=AIC Systems Laboratories, c=JP
   related IP network (X.500 handle):
           IPnw=AIC-NET1, o=AIC Systems Laboratories, c=JP
    External gateway (X.500 handle):
           IPnd=ss2, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP
    External gateway (X.500 handle):
           IPnd=hotaru, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP
    Technical Contact Person (X.500 handle):
           cn=Shuji Sasaki, ou=Network Management Lab.,
           ou=Research and Development, o=AIC Systems Laboratories,
           c=JP
    Administrative Contact Person (X.500 handle):
           cn=Kenichi Higuchi, ou=Network Management Lab.,
           ou=Research and Development, o=AIC Systems Laboratories,
           c=JP
```

図 4.3: X.500 e-mail responder: find an IP network address

```
From: Directory Query Process <X500-Query@aic-wide.aic.co.jp>
Subject: you asked "find ip 150.80.254.1"

Resolution of IP number:

 Delegated Block: 150.001.000.000-150.100.255.255
   assigned from (X.500 handle): (null)
   assigned to (X.500 handle): o=NIC, c=JP

 IP network address: 150.080.000.000
   IP network mask: ffff0000
   assigned to (X.500 handle): o=AIC Systems Laboratories, c=JP
   related IP network (X.500 handle):
           IPnw=AIC-NET1, o=AIC Systems Laboratories, c=JP

 IP host address: 150.080.254.001
   related IP node (X.500 handle):
           IPnd=bass, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP

 IP host name: bass
 IP host domain name: aic-wide.aic.co.jp
 Technical Contact Person (X.500 handle):
           cn=Glenn Mansfield, ou=Network Management Lab.,
           ou=Research and Development, o=AIC Systems Laboratories,
           c=JP
 Administrative Contact Person (X.500 handle):
           cn=Glenn Mansfield, ou=Network Management Lab.,
           ou=Research and Development, o=AIC Systems Laboratories,
           c=JP

Ports of this host:

 Port name: le0
 Port IP number: 150.80.16.1
 Connected IP network (X.500 handle):
           IPnw=2KEN-NET, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP

 Port name: le1
 Port IP number: 150.80.254.1
 Connected IP network (X.500 handle):
           IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP
```

図 4.4: X.500 e-mail responder: find an IP host address

```
From: Directory Query Process <X500-Query@aic-wide.aic.co.jp>
Subject: you asked "list nw IPnw=AIC-NET1, o=AIC Systems Laboratories, c=JP"

 IPnetwork 2KEN-NET
  (X.500 handle for detailed search):
     IPnw=2KEN-NET, IPnw=AIC-NET1, o=AIC Systems Laboratories, c=JP

  IP number / mask:  150.80.16.0 / fffff000
  External IP-Gateway (X.500 handle):
           IPnd=bass, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP
  External IP-Gateway (X.500 handle):
           IPnd=B2, IPnw=2KEN-NET, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP


 IPnetwork AIC composite
  (X.500 handle for detailed search):
           IPnw=AIC composite, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP

  IP number / mask:  150.80.0.0 / fffff000
  External IP-Gateway (X.500 handle):
           IPnd=B2, IPnw=2KEN-NET, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP

 IPnetwork AIC-WIDE-1
  (X.500 handle for detailed search):
           IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
           o=AIC Systems Laboratories, c=JP

  IP number / mask:  150.80.240.0 / fffff000
  External IP-Gateway (X.500 handle):
IPnd=ss2, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
o=AIC Systems Laboratories, c=JP
  External IP-Gateway (X.500 handle):
IPnd=hotaru, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
o=AIC Systems Laboratories, c=JP
  External IP-Gateway (X.500 handle):
IPnd=bass, IPnw=AIC-WIDE-1, IPnw=AIC-NET1,
o=AIC Systems Laboratories, c=JP
```

図 4.5: X.500 e-mail responder: list a network

```
From: Directory Query Process <X500-Query@aic-wide.aic.co.jp>
Subject: you asked "find person Thomas Johannsen"

Description of person:
  Full name:        Thomas Johannsen
  E-mail address:   thomas@aic.co.jp
  Fax number:       +81 22 279 3640
  Business phone:   +81 22 279 3310

  X.500-ufn:        Thomas Johannsen, Network Management Lab.,
                    Research and Development, AIC Systems Laboratories, JP

Works for:

Description of organization:
  Organization Name: AIC Systems Laboratories
                     AIC
  Locality:         Sendai,
  Street address:   6-6-3 Minami Yoshinari
  Postal address:   AIC Systems Laboratories
                    POB
                    Sendai, 989-32
                    Japan
  Description:      Communications Research and Development
  phone:            +81 022-279-3310
  fax:              +81 022-279-3640
  General mail address: aic.co.jp
  Business Category: not-for-profit research

  X.500-ufn:
        AIC Systems Laboratories, JP
```

図 4.6: X.500 e-mail responder: find a person

4.1.2.3 Functional interface using LDAP

With the help of function libraries for Directory calls new DUAs can be build. There is a lightweight access available which has become known under the name LDAP (Lightweight Directory Access Protocol, [62]). To make writing DUAs easier, we built on top of LDAP a network access library, called **nwlib**. Thus, calling a limited number of special functions is all one has to do to use the extended, X.500 based, database. A short description of this library is given below.

Database initialization

Before you call the first read or write function you have to initialize database access. Note: This is a one-time activity. You have to call this function only once at the beginning of your program.

```
int x500_init(ldap_host, user, password);
```

If you get LDAP_ERROR returned, the Directory bind failed for some reason. In that case you *cannot* do any successful read or write operation.

When you are going to quit your program, you should unbind from the Directory. To do this, call

```
void x500_exit();
```

Remember to call x500_init() next time you want to access the database.

Reading data

There are functions to

- list all networks, nodes and ports of the nodes which are immediate subordinates of search base,
- list all networks that match a pattern,
- list all nodes that match a pattern,
- list all ports that match a pattern.

All of the read functions are available for physical and image objects. These functions copy information from the Directory into lists of structures as defined in `nwlib.h` (provided with the library). Memory for list elements is allocated by library functions. You should free the memory when you don't need a list any longer.

```
int list_network_elements(nw_dn, nws, nds, pos)
char *nw_dn;
struct nw_list **nws;
struct nd_list **nds;
struct po_list **pos;
```

Input:	nw_dn	search base as user friendly distinguished name
	nws	address of pointer to list of networks

nds	address of pointer to list of nodes
pos	address of pointer to list of ports

Output: exit status (LDAP_OK, LDAP_ERROR)
nws, nds and pos filled with pointers to lists.
NULL if list has no elements.

Writing data

Write access to the database is not yet supported. If you have the need to add or modify entries, please contact spp-support@aic.co.jp.

Library usage

For compiling DUAs that use one of the nwlib-functions calls, one has to add the following line to the C module:

```
#include ⟨nwlib.h⟩
```

Furthermore, the Makefile should have the following entries:

```
  NWINC = /home2/local/isode-8.0/include # path to nwlib.h
NWLIB = /home2/local/isode-8.0/lib # path to nwlib.a
LDAPLIB = /home2/local/isode-8.0/lib # path to libldap.a, liblber.a

INCLUDES = -I$(NWINC)
LIBS = -L$(NWLIB) -lnw -L$(LDAPINC) -lldap -llber
```

4.1.2.4 AIMS network map converter

Within the ConMan Project an intelligent network manager program has been developed (AIMS[1]). One module of this program package is reading network configuration information from the X.500 Directory and processes it for displaying network maps on the screen. Based on this configuration information configuration management tasks (like checking routing tables, etc.) are carried out.

[1]© by AIC Systems Lab.

第 5 章

ネットワーク停止予定データベースシステム

5.1　はじめに

インターネットワークにおいてネットワークの機能が停止する原因には、ネットワーク機器のハングアップというような真の障害ではない、電源工事・点検による停電や機器のリプレース、OS のバージョンアップのためのゲートウェイ停止といった “予め予定された停止” が比較的多い。

現状、管理者はネットワークを停止させる必要が生じた場合には、その停止予定を例えば図 5.1に示すような文書にして、ネットワーク管理者間連絡用のメーリングリスト、あるいはネットワークニュースにその旨を報告して他の管理者やユーザに知らせるようにしている。

```
ネットワーク管理者の皆様

電源の点検に伴う停電のため下記の通り当社設置の機器を停止しますので
報告致します。御迷惑をおかけしますが、よろしくお願い致します。

    予定日時：  4 月 1 日 (木)  9:00 ～ 12:00
    停止機器：  wnoc-snd.wide.ad.jp
               wnoc-snd-ss2.wide.ad.jp
               aic-wide.aic.co.jp

尚、下位サイトはありません。
---
高度通信システム研究所　村田眞人（murata@aic.co.jp）
```

図 5.1: 停止予定報告文書の例

しかしながら、インターネットワークの大規模化に伴ってその数も増加しており、さらに報告される予定内容も数時間後のものから、一カ月後のものまで及ぶため、ある特定日時の各ネットワークの稼働状況を正確に把握するのは難しくなってきている。

そこでこのような情報をデータベース化し、必要な時に必要なデータを検索できるようにしておくことが望まれる。例えば、あるネットワークへのアクセスが不可能となった

場合に、このネットワーク停止予定データベースを検索することにより、それが障害によるものなのか、予定された停止によるものなのかを区別できることが可能となる。これは管理者にとって対処をする上で大きな助けとなる。

本章では、このネットワーク停止予定データベースシステムに関し、電子メールを利用したアクセスによりデータの登録や検索等を実現する方法等について報告する。

5.2 実現方法

本節では、ネットワーク停止予定データベースシステムの実現方法について述べる。先ず、本データベースシステムの実現は次に示す方針で行なった。

1. 各ドメインのネットワーク管理者自身が遠隔地にあるデータベースに対して、停止予定の登録、削除、変更作業を行なえること。
2. ネットワークの停止によって影響をうける可能性があるすべてのユーザからの検索が可能なこと。
3. 上記 1、2 の各作業が自動的に行なわれる (データベースサーバ側の人手を介さない) こと。
4. 移植性が高いこと。

将来的には、障害時にも確実に情報が取れるようにするために、本システムはインターネットワーク上に分散配置されるのが望ましい。よって移植性の高さは一つの重要なポイントである。

5.2.1 データベースサーバへのアクセス方法

インターネットワークにおいて、遠隔地に存在するデータベースサーバへのアクセス方法としては、一般的に以下のような方法が考えられる。

1. データベースサーバのホストにログインして直接操作する方法。
2. サーバ・クライアント方式で直接遠隔操作する方法。
3. 電子メールを利用して間接的に遠隔操作する方法。

ここで、上記 1,2 の方法はユーザが利用するホストとサーバ間での IP 接続を必要とするが、インターネットワーク上のネットワークが停止することによって影響を受けるのはインターネットワーク上のサイトだけではない。例えば UUCP によって接続され、電子メールやネットワークニュースのサービスを受けているサイトも含まれる。

したがって、このようなサイトからもアクセスを可能とし、実現方針の 2 に挙げた「ネットワークの停止によって影響をうける可能性があるすべてのユーザからの検索が可能なこと。」を実現するために、本データベースへのアクセス方法には電子メールを利用することにした。

5.2.2　データベース操作のための処理要求文書

電子メールに記述された処理要求文書からデータベース操作を自動的に行なうためには、文書の型(フォーマット)を明確に規定しておく必要がある。本システムにおいては、文書型の定義はISOで標準化されたSGML(the Standard Generalized Markup Language)[63, 64]を用いて行なった。

ここで簡単に SGML について説明する。SGML は文書交換を可能とする目的で設計された文書記述のための ISO 標準である。SGML を使用して記述された文書 (SGML 文書と呼ばれる) は図 5.2に示すように、

- 下記の文書型定義および文書インスタンスで使用する文字の定義を行なう SGML 宣言。
- 文書構造を定義する文書型定義 (DTD: Document Type Definition)。
- DTD で定義された構造に基づいてマーク付けされた文書。

の 3 つの部分より構成されている。SGML 処理システムでは、先ず SGML パーザが構文解析を行ない、文書インスタンスが DTD で定義された文書型に合致しているか否かを検証する。合致している場合には、省略されたタグの復元を行ない、フルマークアップデータを生成し、アプリケーションプログラムに渡す。

今回、処理要求文書として、実現方針の機能を満たすように、以下の 4 つの文書型を定義した。

- ネットワーク停止予定登録文書 (outagein)。
- ネットワーク停止予定登録取消文書 (outagede)。
- ネットワーク停止予定更新文書 (outageup)。
- ネットワーク停止予定検索文書 (outagequ)。

なお、実際にどのような情報を停止予定として登録するかについては、ネットワーク管理者間のメーリングリストに報告された過去約 1 年間の停止予定の報告例を参考にして決定した。例として、図 5.3にネットワーク停止予定登録文書の文書型定義を示す。

ここで、本来 SGML 文書は既に述べたように SGML 宣言、DTD そして文書インスタンスから構成されるが、通信トラヒックを軽減するため電子メールでは SGML 文書インスタンスである処理要求文書だけを送るものとしている。

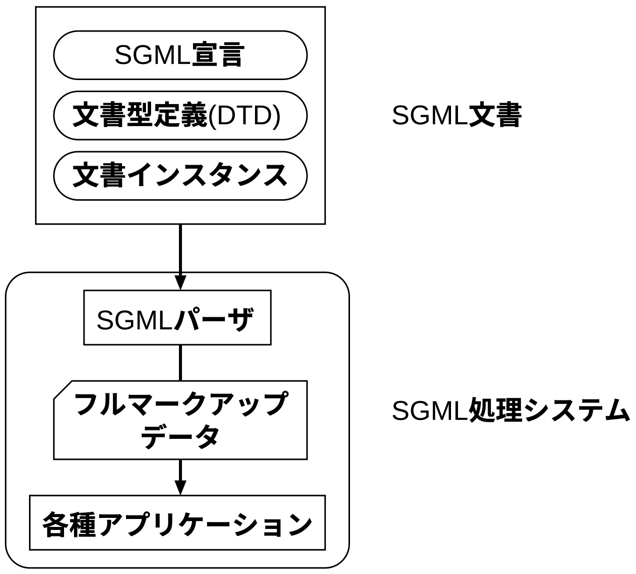


図 5.2: SGML の概要

```
<!DOCTYPE outagein [
<!ENTITY % TEXT "#PCDATA" >
<!ELEMENT outagein - -  (start & finish & sysname & sysipadr? &
cause? & effect? & note? & author & org & email & tel? & fax?)>
<!ELEMENT start    - O  (%TEXT;)>  <!-- 停止開始予定日時 -->
<!ELEMENT finish   - O  (%TEXT;)>  <!-- 停止終了予定日時 -->
<!ELEMENT sysname  - O  (%TEXT;)>  <!-- 停止予定システム名 -->
<!ELEMENT sysipadr - O  (%TEXT;)>  <!-- 停止予定システム IP アドレス -->
<!ELEMENT cause    - O  (%TEXT;)>  <!-- 停止原因 -->
<!ELEMENT effect   - O  (%TEXT;)>  <!-- 停止による影響 -->
<!ELEMENT note     - O  (%TEXT;)>  <!-- その他連絡事項 -->
<!ELEMENT author   - O  (%TEXT;)>  <!-- 報告者 -->
<!ELEMENT org      - O  (%TEXT;)>  <!-- 報告者の所属組織 -->
<!ELEMENT email    - O  (%TEXT;)>  <!-- 報告者の電子メールアドレス -->
<!ELEMENT tel      - O  (%TEXT;)>  <!-- 報告者への連絡電話番号 -->
<!ELEMENT fax      - O  (%TEXT;)>  <!-- 報告者への連絡 FAX 番号 -->
]>
```

注) `<!-- -->` の部分はコメントである。

図 5.3: ネットワーク停止予定登録文書の文書型定義

5.2.3　処理手順

図 5.4に本データベースサーバでの処理の流れを示し、その各処理ステップについて説明する。
処理プロセスは、cron を利用して、定期的 (現状では 1 分毎) に起動される。

処理 P1： メールスプールから着信メールを読み込む。

処理 P2： 着信メールから送信者の電子メールアドレスと、処理要求文書を抽出する。そして、データベースに変更を加える処理要求の場合には、送信者に対する認証処理を行なう。

処理 P3： P2 で抽出した処理要求文書に SGML 宣言と DTD を付加して SGML 文書を再構成する。

処理 P4： P3 で再構成された SGML 文書の構文解析を行ない、処理要求文書が DTD に定義された文書構造に合致しているかどうかを検証する。合致している場合には、省略タグが復元されたフルマークアップデータを出力する。

処理 P5： P4 ではチェックできない個々のデータ値の妥当性 (時間情報が 24 時間の範囲にあるか等) や、データ間の値の整合性 (開始時間は終了時間より早いか等) のチェックを行ない、妥当な場合にはフルマークアップデータから処理要求に応じた SQL コマンドを作成する[1]。

処理 P6： P5 で作成された SQL コマンドによって RDB を操作する。そしてその処理結果から返信メールを作成し、送信者に発信する。

5.3　利用例

この節では、本システムの利用例を紹介する。
本システムは、1993 年 4 月 12 日より試験運用を開始している。処理要求メールの送り先は、

```
outagedb@wnoc-snd-ss2.wide.ad.jp
```

である。
システムの利用にあたって、ユーザは文書型定義等の詳細を知る必要は特になく、タグ名と日時の入力フォーマット、必須記述事項さえ分かっていれば十分である。この詳細については付録 6を参照されたい。

[1]現在は DBMS に POSTGRES を使用しているので、正確には POSTQUEL コマンドを作成している。また、フルマークアップデータを解釈して直接データベースを操作するような実現方法も可能であるが、移植性を考慮して一旦標準である SQL コマンドに落している。

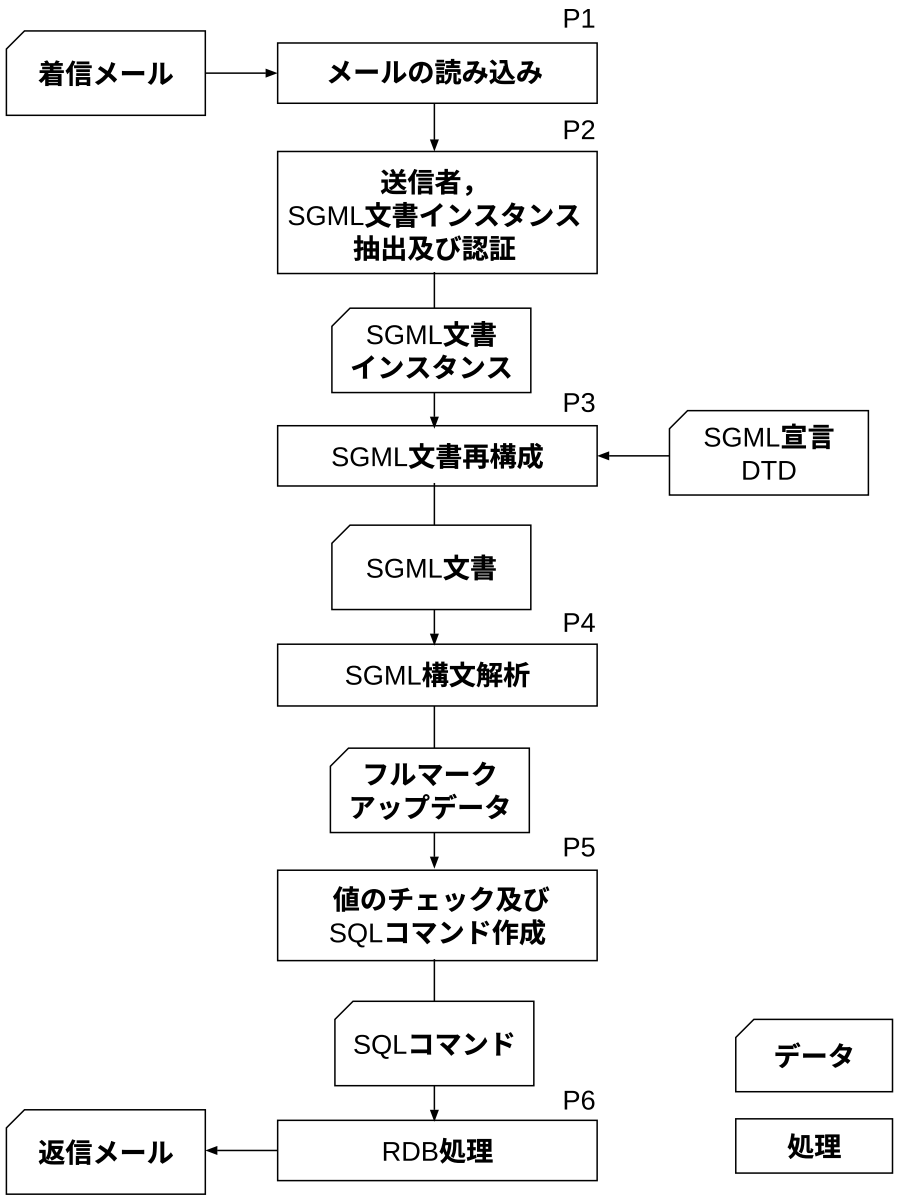

P1
着信メール
メールの読み込み
P2
送信者，SGML文書インスタンス抽出及び認証
SGML文書インスタンス
P3
SGML文書再構成
SGML宣言 DTD
SGML文書
P4
SGML構文解析
フルマークアップデータ
P5
値のチェック及びSQLコマンド作成
SQLコマンド
P6
RDB処理
返信メール
データ
処理

図 5.4: 処理の流れ

先ず、停止予定データの登録の例を示す。例えば 5.1節に紹介した例は図 5.5に示すような文書として記述することができる。開始タグから終了タグの間以外は処理されないので、何を記述しても構わない。

```
ネットワーク管理者の皆様

以下のように当社設置の機器を停止しますので報告致します。
御迷惑をおかけしますが、よろしくお願い致します。
<outagein>
  <start>    4 1 9:00 1993
  <finish>   4 1 12:00
  <sysname>  wnoc-snd.wide.ad.jp
             wnoc-snd-ss2.wide.ad.jp
             aic-wide.aic.co.jp
  <cause>    電源の点検に伴う停電のため。
  <effect>   下位サイトはありません。
  <author>   村田眞人
  <org>      高度通信システム研究所 (aic.co.jp)
  <email>    murata@aic.co.jp
</outagein>
```

図 5.5: ネットワーク停止予定登録文書例

このメールを上記メールアドレス宛に送付すると、登録番号とパスワードが返信されてくる。停止の予定が無くなった場合や、予定に変更が生じた場合にはこれらの番号を利用して取消やデータの更新を行なう。

次に停止予定の検索の例を示す。検索要求文書において、検索キーワード間の関係は and である。また、日時以外の検索は正規表現 “.*検索キーワード.*” でのパターンマッチングを利用して行なわれる。ゆえに、例えば図 5.6の例は、「1993 年 4 月 1 日の 10:00 に停止していて、かつシステム名に “wide.ad.jp” が付く停止予定を全て検索せよ。」という意味となる。この処理の結果、例えば図 5.7のような検索結果が返信されてくる。

```
<outagequ>
  <start>    4 1 10:00 1993
  <sysname>  wide.ad.jp
</outagequ>
```

図 5.6: ネットワーク停止予定検索文書例

```
 "ネットワーク停止予定検索要求" は正常に実行されました。
以下に検索要求とその検索結果を示します。

要求文書:
      <outagequ>
        <start>    4 1 10:00 1993
        <sysname>  wide.ad.jp
      </outagequ>

検索結果:
======== ネットワーク停止予定 ( Apr 1 9:00:01 1993 現在 ) ========

  System:  wnoc-snd.wide.ad.jp wnoc-snd-ss2.wide.ad.jp aic-wide.aic.co.jp
    Date:  Thu Apr 1 9:00 1993 --> Thu Apr 1 12:00 1993
   Cause:  電源の点検に伴う停電のため。
  Effect:  下位サイトはありません。
  Author:  村田眞人
     Org:  高度通信システム研究所 (aic.co.jp)
   Email:  murata@aic.co.jp
```

図 5.7: ネットワーク停止予定の検索結果例

5.4 本システムを利用した付加サービス

これまでに説明した方法により、電子メールを用いた登録、検索等の基本サービスが実現できた。しかしながら、ユーザが共通して持つような検索要求に対しては、各ユーザからの個々の検索要求を待って処理、返答するのではなく、サーバ側で定期的に処理を行ない、希望者に自動的に情報を送付する方がユーザに与える負荷も少なく、さらにサーバの負荷やトラヒックの軽減のためにも良い。そこで、ユーザに共通すると思われる検索要求を本システムの付加サービスとして実現することにした。現在実現している付加サービスは以下の 2 つである。

- 毎週日曜日に次の一週間分の停止予定を送付する。
- WIDE の NOC(Network Operation Center) やバックボーンのリンクがダウンする予定がある場合、前日にその予定を送付する。

5.5 まとめと今後の課題

本章ではネットワーク停止予定データベースの実現方法とその利用方法について述べた。主な特徴は以下の通りである。

- データベース操作のインタフェースに電子メールを利用し、サーバに直接 IP 接続ができない組織からの利用も可能としている。

- 処理要求文書を OSI 標準である SGML を用いて規定している。

本システムは、プログラミング言語 perl を用いて実装を行なった。DBMS には University of Calfornia, Berkley で開発された POSTGRES を、SGML パーザには SGML User's Group (SGMLUG) が配布しているパーザ用コードをもとに James Clark 氏が作成した sgmls をそれぞれ利用している。現在、WIDE 参加組織に対して実験運用を行なっている。

今後の課題としては、

- 電子メールインタフェースしか使えないのは逆に不便だという意見が多いことから、直接データベース操作を行なうコマンドの作成。
- 付加サービス機能の充実。
- ネットワーク管理システムとのリンク。
- データベースサーバの配置問題。

等が挙げられる。

第 6 章

付録: ネットワーク停止予定データベース操作一覧

6.1　処理要求メールの送付先

```
outagedb@wnoc-snd-ss2.wide.ad.jp
```

6.2　日時の指定フォーマット

```
month day hour:min [year] []:省略可

  month = 1,2, .. ,11,12 |
          Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
    day = 1,2, .. ,30,31
   hour = 0,1, .. ,22,23
    min = 0,1, .. ,58,59
   year = 西暦 (4 桁)。デフォルトはメールが処理された年。
```

6.3　停止予定登録文書

`<outagein>`	登録文書開始タグ (必須)
`<start>`	停止開始予定日時 (必須)
`<finish>`	停止終了予定日時 (必須)
`<sysname>`	停止予定システム名 (必須)
`<sysipadr>`	停止予定システム IP アドレス
`<cause>`	停止原因
`<effect>`	停止による影響
`<note>`	その他連絡事項
`<author>`	報告者 (必須)
`<org>`	報告者の所属組織 (必須)
`<email>`	報告者の電子メールアドレス (必須)
`<tel>`	報告者の連絡電話番号

`<fax>`	報告者の連絡 FAX 番号
`</outagein>`	登録文書終了タグ (必須)

6.4 停止予定登録取消文書

`<outagede>`	登録取消文書開始タグ (必須)
`<regno>`	取消す予定の登録番号 (必須)
`<password>`	取消す予定のパスワード (必須)
`</outagede>`	登録文書終了タグ (必須)

6.5 停止予定検索文書

`<outagequ>`	検索文書開始タグ (必須)
`<start>`	検索開始日時
`<finish>`	検索終了日時
`<sysname>`	検索システム名
`<sysipadr>`	検索 IP アドレス
`<cause>`	検索停止原因
`<effect>`	停止による影響に関する検索キーワード
`<note>`	その他連絡事項に関する検索キーワード
`<author>`	検索報告者
`<org>`	検索組織
`<email>`	検索電子メールアドレス
`<tel>`	検索電話番号
`<fax>`	検索 FAX 番号
`</outagequ>`	検索文書終了タグ (必須)

検索キーワードが一つも指定されない場合にはエラーとなる。

6.6 停止予定更新文書

`<outageup>`	更新文書開始タグ (必須)
`<regno>`	データを更新する予定の登録番号 (必須)
`<password>`	データを更新する予定のパスワード (必須)
`<start>`	新しい停止開始予定日時
`<finish>`	新しい停止終了予定日時
`<sysname>`	新しい停止予定システム名
`<sysipadr>`	新しい停止予定システム IP アドレス

`<cause>`	新しい停止原因
`<effect>`	新しい停止による影響
`<note>`	新しいその他連絡事項
`<author>`	新しい報告者
`<org>`	新しい報告者の所属組織
`<email>`	新しい報告者の電子メールアドレス
`<tel>`	新しい報告者の連絡電話番号
`<fax>`	新しい報告者の連絡 FAX 番号
`</outageup>`	更新文書終了タグ (必須)

更新データが一つも存在しない場合にはエラーとなる。