第 14 部

OSI アプリケーション

第 1 章

1992 年度の活動概要

1.1 ISODE WG

WIDE ISODE ワーキンググループ (ISODE WG) では、ネットワークプロトコルを既存のものから他の新しいプロトコルへ移行 (マイグレーション) するための技術の蓄積と研究を目的としており、既存のプロトコルとして TCP/IP を、新しいプロトコルとして OSI を対象とした研究・実験を行なっている。

ISODE WG はその研究を行うに当たって、OSI プロトコルのプラットホームとして ISODE パッケージを使用している。ISODE は既存の TCP/IP 上に OSI トランスポート層を提供するものであり、OSI セッション層、OSI プレゼンテーション層、OSI アプリケーション層を構成する。本来は OSI を学習するために作成されたパッケージであるが、OSI サービスを利用するためのプロダクトや、TCP/IP スーツから OSI スーツへの移行手段と捉えることができる。OSI の規格は定められているが、その実装方法については何も規定されていない。そのため、OSI プロトコルの実装例の一例としてみることもできる。ISODE の現在の最新バージョンは ISODE 8.0 で、ディレクトリ・サービスを提供する QUIPU の他、ファイル転送、仮想端末、ネットワーク管理のモジュールが含まれている。また、ISODE 用の MHS (メッセージハンドリングシステム) を提供する PP、セキュリティ機能を提供する OSISEC パッケージも用意されている。

1.2 研究・実験内容

昨年度 (1991 年度) は OSI ディレクトリサービスを対象とした実験環境の構築と、基礎実験、および構築した OSI ディレクトリ上での各種の研究を行なった。

本年度 (1992 年度) は、OSI ディレクトリサービスの研究を引き続いて行なったのに加え、MHS (Message Handling System) および OSI セキュリティについての研究のための調査と基礎実験を行なった。

具体的な実験・研究内容は以下の通りである。

- OSI ディレクトリサービスによる不特定情報管理の試み
- OSI ディレクトリサービスによるネットワーク情報管理の試み

- MHS (メッセージハンドリングシステム) の調査と基礎実験
- OSI セキュリティの調査と基礎実験

第 2 章

OSI ディレクトリサービスによる不特定情報管理

2.1 はじめに

ネットワーク研究に関する国際会議 INET'92 が 1992 年 6 月神戸で行われた。ISODE WG では、これまでの WIDE インターネット上での OSI ディレクトリサービス運用実験で得た知識を生かして、会場の端末室で利用できる会場案内サービスの提供を試みた。ここでは、既存のディレクトリサービスによる、より一般的な不特定情報の管理の試みである、会場案内サービスの構築についてその方法と問題点を述べる。

2.2 背景

ISODE WG では、1991 年 7 月から OSI ディレクトリサービスの研究の一環として、ISODE パッケージ [147] 中の CCITT 勧告 X.500[49] インプリメンテーションである QUIPU[148] を用いて運用実験を行って来た。その中で、WIDE インターネット上での構築・実験を始めとして、1991 年 11 月国際的な OSI ディレクトリ運用実験プロジェクトである WhitePages プロジェクトにも参加し、今後の本格的な運用、あるいはディレクトリサービスを用いた様々な応用実験への準備を行ってきた。現在国内ではエントリ数にして 2,300 ほどが接続されている。

また、ISODE WG では、より使いやすいシステム・実用的なシステムを目指して独自に QUIPU の拡張を試みている。例えば QUIPU の日本語化作業を行い、属性値・属性タイプ・DN(Distinguished Name) などの日本語入出力を可能にした。あるいはまた、ディレクトリへの簡易なアクセスを可能にするために Emacs インタフェースの試作を行った。さらに、新規オブジェクトクラス・属性タイプの定義、設定の実験も行い、既存タイプのオブジェクト・属性では提供できない情報をディレクトリを用いて参照・検索するシステムを試作し、QUIPU 上でこれらを実現するために必要な技術的情報を蓄積してきた。

ISODE WG では、これらの実験・運用を通して、ディレクトリサービスの様々な応用を考えていき、ディレクトリの特徴を生かしたシステムの構築あるいは既存システムからの移行を進めていこうとしている。

2.3 本実験の動機と目的

2.3.1 動機

1992 年 6 月 15 日から 18 日にかけて INET'92 国際会議が神戸のポートアイランドで行われた。この INET'92 の会場 International Conference Center Kobe では WIDE プロジェクトの支援のもとでマシン室・端末室が設置され、192Kbps の専用回線で慶応大学藤沢キャンパスからネットワーク接続することにより入場者に電子メイル・telnet などのサービスを提供することになった。

ISODE WG では、ディレクトリサービスの新しい応用として、世界中から集まる研究者に対し、ディレクトリサービスを応用した何らかの有益なサービスを提供する事を考えた。これは、ISODE WG の実験であるとともに WIDE プロジェクトのディレクトリサービス研究活動の紹介も兼ねることになる。

提供する情報としては、期間中に行なわれるセッションの内容や、そこで発表される論文の内容、あるいは発表者のプロフィール、会場の案内図、セッションのスケジュールなどとし、QUIPU を使用して入場者に案内することにした。WhitePages プロジェクトに参加している関係上、提供する情報は海外など他のディレクトリユーザから参照できるべきものであるが、ディレクトリシステムに不慣れなユーザを想定し、会場の端末室で手軽にアクセスできる環境が望ましいと思われた。そこで、検索機能などを強力にした DIB やインタフェースを独自に構築することとした。

2.3.2 目的

本実験の目的は次の通りである。

- これまで蓄積された運用技術その他を実用システムに応用する
- ディレクトリサービスのデモンストレーションを行なう
- ディレクトリサービスの将来的な応用面の研究の一環として、ネットワークユーザに対し新しい利用形態の一つを提示し、実際に使用することによって評価を行う

2.4 会場案内サービス

今回実験した会場案内サービスでは、次の情報をユーザ (INET'92 来場者) に提供した。

- 発表者に関する情報 (氏名、所属組織)
- 発表論文に関する情報 (タイトル、要約、キーワード、著者)
- セッションに関する情報 (タイトル、日時、場所、座長)

- 地図 (会場へのアクセスマップ、会場の平面図)

本会場案内サービスのために ISODE WG では次のような環境を新しく用意した。

1) INET'92 用に新たに起動したマスターおよびスレーブの DSA (Directory System Agent)

ディレクトリ管理者の立場から、意味的なまとまりのあるデータを一括管理する DSA を新たに起動するのが良いと考えた。本会場案内サービス用に新たに起動された DSA は、hskgw.hitachi-sk.co.jp 上で起動されたマスター DSA と、会場のワークステーション (r36.inet.wide.ad.jp) で起動したスレーブ DSA の 2 つである。

2) INET'92 のために追加されたディレクトリ内サブツリー

今回来場者に参照してもらうデータは、新たに作成した DIT (Directory Information Tree) 中のサブツリーに全て登録した。具体的には、sh.wide.ad.jp 上で管理している c=JP@o=WIDE の下に ou=INET92 というエントリを作成した。さらに、この下に

ou=map

ou=paper

ou=author

ou=session

ou=schedule

というエントリを作成し、各データのエントリを分類してこれらの下に登録した。

3) 発表者、セッション、論文、会場マップの各データ

4) 簡易検索のための改造 DUA(Directory User Agent) : INETPOD

QUIPU が提供している DUA の一つである POD を改造し、ユーザの検索・参照作業の簡略化を計った。

2.5　INETPOD の構築

ディレクトリサービスの最も一般的な機能の一つとして、組織や人に関する情報、例えば電話番号や住所、電子メイルアドレスなどを分散管理し、ネットワークを介して検索・参照するというものがある。

しかし、世の中にはこのような情報に関連する他の様々な情報もあり、それらもディレクトリサービスによって管理・提供する事ができるであろう。

本会場案内サービスは、その一応用例として企画された。

ISODE WG はこの中で、ユーザインタフェースもサービスの一環と見なし、会場案内サービスにふさわしいインタフェースを構築することにした。例えば、地図イメージの上をマウスでクリックすると、そのクリックした場所に関連する別のエントリの情報を出力してくれるインタフェースが考えらる。ISODE WG は、このようなインタフェースをディレクトリの機能をうまく利用した形で実現する方法を検討した。

しかし、X.500 あるいは QUIPU の実装の範疇でそれを行うのは現実には難しく、前項の 3),4) に関連していくつかの改造を必要とした。

以下に本会場案内サービス用インタフェース INETPOD の構築について述べる。

2.5.1 ディレクトリインタフェースの問題点

本会場案内サービスでは、ディレクトリサービス自体に馴染みのないユーザを想定し、欲しい情報を簡単な操作で得ることができるサービスを目指した。そこで、QUIPU が用意している複数の DUA のうち、よりユーザフレンドリであると思われる X Window System をベースにマウスによるオペレーションが行える POD を基本的なユーザ・インタフェースとすることにした。

しかし提供する情報を単にディレクトリ中に用意するだけでは、ユーザは欲しい情報を見つけ出すために複雑な操作を行なわなければならない。

X.500 ディレクトリサービスでは、ユーザが欲しい情報を得る方法に 2 通りある。

1) DN あるいは RDN (Relative Distinguished Name) から目的とするエントリを推測する。

2) X.500 で規定されている、ディレクトリの検索 (Search) サービスを利用して、キー情報を与えることで目的とするエントリを検索する。

1) の場合、DIT の構造が意味的に理解しやすいものであれば、ユーザ自身による検索操作はさほど複雑なものではないだろう。例えば、今回のように、c=JP@o=WIDE@ou=INET92 の下に関連情報が全てある場合など、欲しい情報が DIT 中のどこにあるのかを推測するのは簡単である。しかし、発表者情報が世界中に散らばっているような場合には、多数のエントリを DN あるいは RDN を元に探すのは負担の大きい作業となる。

2) の場合、ユーザが自分の欲しい情報を検索するための適切なキー情報を知っていれば、簡単・確実に目的のエントリを探しだすことが出来る。例えば、電子メイルアドレスを知っていて、その人に関する他の情報を得たい場合には、電子メイルアドレスをキー情報にして検索要求を出せば良い。ちなみに POD では、任意の文字列をキーワードとしてキー入力し、それに合致するエントリの情報を検索して出力するよう実装されている。

しかし、この検索サービスを利用したとしても、特定のエントリを検索するためのキー情報 (キーワード) の入力は、ユーザにかなりの試行錯誤を強いることになろう。特に本会場案内サービスのようにキー情報を特定しにくい場合はそうである。例えば、あるセッションの情報を見ていたユーザが、そのセッションに関する他の情報を得たいと考えた

とき、そのような情報としてどんなものが用意されていてそれはどのようなキー情報で検索できるのかを知るのは難しい。X.500 ディレクトリサービスではエントリの属性として「参照 (seeAlso)」があり、関連するエントリを記述することができるが、

- 関連するエントリが複数存在する場合、各エントリの参照操作を繰り返すことになり、操作全体が繁雑である。
- エントリの名前ではその内容を十分表し切れない場合、ユーザの判断基準となりにくい。

などのため、十分とは言えない。

2.5.2　解決方法

関連情報を簡単に得るための方法として、次の 2 つを検討した。

1. 関連するエントリをまとめて一つのサブツリーに配置する。関連エントリは重複することが考えられるので、エントリのエイリアス機能を利用する。
2. エントリの属性値として、関連エントリを指すポインタをあらかじめ用意する。ポインタとして、関連エントリに含まれるキーワードを用い、関連エントリの参照操作はディレクトリの検索サービスを利用する。

1) の場合、ユーザはあるエントリの関連エントリを DIT 中の同じ場所を探すことによって得られる。しかし、同一エントリが複数のエントリに関連するという重複がかなり起こることが予想される上に、エントリのエイリアスの設定を行なうには、そのエントリの実体の DN を得る必要があり、DIT の構築時の負担が大きい。2) の場合、あらかじめ関連していると思われるエントリのキーワードを属性値として持たせ、検索サービスによってそのエントリを参照することになる。これなら検索サービスを介してあたかも直接リンクを張っているような効果を得られる。しかし、関連エントリのキーワードを持たせられる属性タイプは定義されておらず、独自に作成する必要がある。

本実験では 2) を採用した。その理由は、キーワードをポインタとしているので、ISODE が提供するライブラリ (DAP の実装) とのインタフェースが取りやすいことと、複数エントリへのリンクが取れるので、DIT 構築時の拡張性に富むからである。

そこで、POD を改造し、次のようなインタフェースを用意することにした。

- メニューによるキーワードの提示、選択、検索、参照
- 地図やスケジュール表の上をマウスでクリックすることによる、関連情報の検索や参照

ユーザの検索・参照作業は図 2.1のようになる。

この INET'92 会場案内サービスを目的として ISODE WG によって作成された X Window System ベースの DUA を INETPOD と呼ぶ。

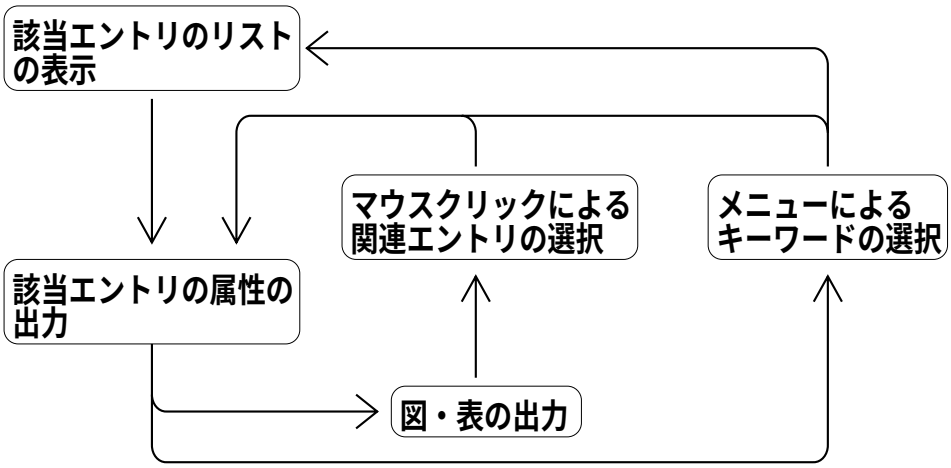

図 2.1: 検索・参照作業

2.5.3 INETPOD

INETPOD の画面の一例を図 2.2に示す。
INETPOD を使ってユーザは次のことが行なえる。可能な検索手順を図 2.3に表す。

- プルダウンメニューから、ある一日のスケジュール表を検索する。
- プルダウンメニューから、あるセッションのタイトル・スケジュールを検索する。
- プルダウンメニューから、会場の特定のフロアの案内マップを検索する。
- プルダウンメニューから、神戸アクセスマップを検索する。
- マップ上をマウスでクリックすることで、その場所で行なわれる予定のセッションのタイトルその他を検索する。
- スケジュール表上をマウスでクリックすることで、その時間帯に行なわれるセッションのタイトルその他を検索する。
- ある時間帯のセッションの一覧表をクリックすることによって、そのセッションの行なわれるフロアのマップ、あるいはそのセッションの詳しい情報を検索する。
- POD のウィンドウ内に直接キーワードをキー入力することにより任意のキーワードによる検索を行なう。

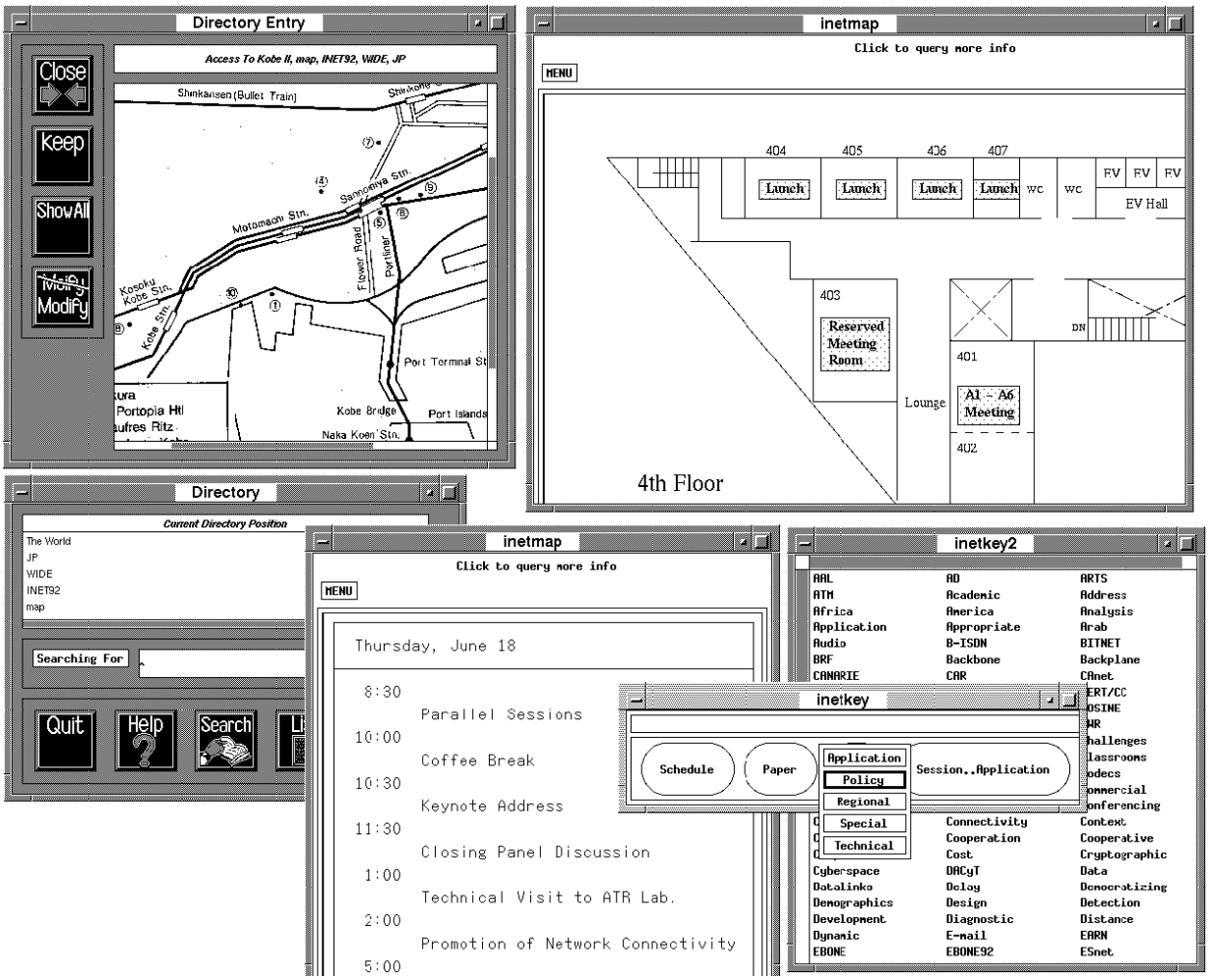

図 2.2: INETPOD の画面

2.5.4　INETPOD 環境の構築

新規オブジェクトクラス・属性タイプの作成

今回のデモシステムで用いたエントリのオブジェクトクラス・属性タイプのほとんどは新規に作成した。理由として、

- 限られた用途のエントリであることを明確にする。
- キーワードを直接属性値として持たせるなど、既存のタイプにはない性質を持つ属性を扱う。

がある。
以下に本実験で追加した属性タイプ・オブジェクトクラスを示す。

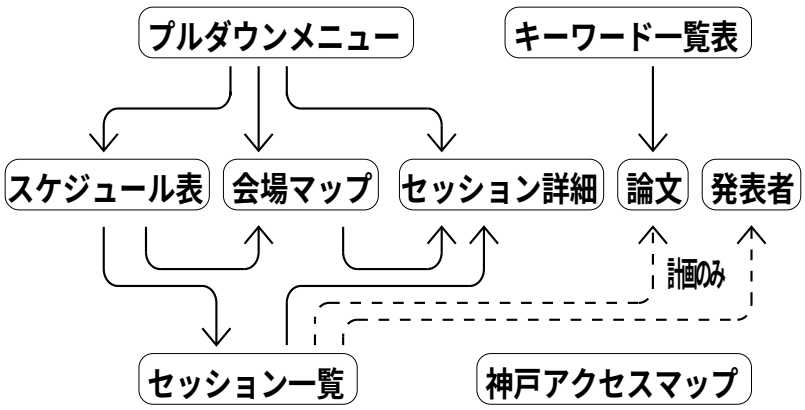

図 2.3: 検索・参照手順

新規オブジェクトクラス

INETPOD のために表 2.1の新しいオブジェクトクラスを定義した。

表 2.1: 新規オブジェクトクラス

名称	内容	属性
inetpaper:	発表論文情報	タイトル、要旨、ページ、著者、キーワード
inetsession:	セッション情報	タイトル、日時、座長、場所
inetmap:	会場案内図情報	タイトル、キーワード+イメージ
inetauthor:	発表者情報	発表者名、所属

新規属性タイプ

INETPOD のために表 2.2の新しい属性タイプを定義した。

このうち、paperKeyword,sessionTitle,sessionPlace,mapTitle,authorKeyword を関連エントリへのポインタ (キーワード) に用いた。searchInfoOnImage 属性には、イメージ上の矩形領域とキーワードの対応関係、およびキーワードによる検索を行う DIT 中のサブツリーを記述できるよう、新たに属性構文とエンコーディング処理を実装した。

2.6 評価

本案内サービスは、ディレクトリに組織・人情報以外の情報を登録し、有用なサービスとして提供する初めての試みであり、これまでディレクトリに触れたことのある参加者の一部には非常に好評であった。また、分散化されたエントリそのものに、関連エント

表 2.2: 新規属性タイプ

属性名	オブジェクト識別子	属性構文名
paperAbstract:	inetAttributeType.131	:CaseIgnoreString
paperPage:	inetAttributeType.132	:CaseIgnoreString
paperAuthor:	inetAttributeType.133	:DN
paperKeyword:	inetAttributeType.134	:CaseIgnoreString
paperTitle:	inetAttributeType.135	:CaseIgnoreString
sessionDate:	inetAttributeType.151	:CaseIgnoreString
sessionPlace:	inetAttributeType.152	:CaseIgnoreString
sessionTitle:	inetAttributeType.153	:CaseIgnoreString
searchInfoOnImage:	inetAttributeType.172	:RegionSearchParamater
mapTitle:	inetAttributeType.173	:CaseIgnoreString
authorO:	inetAttributeType.191	:CaseIgnoreString
authorKeyword:	inetAttributeType.192	:CaseIgnoreString
speaker:	inetAttributeType.193	:DN

リへのポインタを登録しておくという方法によって、ディレクトリサービスの簡易インタフェースとして満足できるレベルのものが実現できたと考える。

2.7　考察

今回の実験を通して表面化した問題について以下に考察する。

2.7.1　ディレクトリサービスの有効性

今回のようなサービスでディレクトリを用いることには次のような効果があると考えられる。

第一に、この目的のために新規に登録するデータの登録作業を分割することが出来る。例えば、今回は行えなかったが、発表者の顔写真などをデータベース化するために短期間でデータ収集するのは大変な労力が必要であろう。そこで、発表者の写真をあらかじめディレクトリ内に発表者自身に登録しておいてもらう、あるいは論文のキーワードを発表者自ら抽出して、ディレクトリ内に登録しておいてもらう。こうすることで、ローカルなディスク上に全世界の情報を記録しておく必要がなくなり、最新・適切な情報を簡単にデータベース化でき、データ収集の労力を非常に軽減できるであろう。また、キー

ワード情報や地図情報など、作成者・作成日時によってデータの所在がまちまちになりがちなものについても同様に、例えば会場のディスク上などの一箇所に集めるという繁雑な作業を簡略化できるだろう。この上で、INETPOD のようなインタフェースをかぶせれば、データへのアクセスの点では見かけ上なんら差を生まないで済む。

第二に、今回の会場のような一時的なマシン環境でのデータベースの構築を確実に行える。会場の端末室のマシンのように、非常に短期間だけの一時的なマシン環境では、マシンの設置後にデータベースの構築作業に入ることになる。本来なら二次媒体あるいは単純なファイル転送などを用いて既存の環境からの移行を行うことになるが、ディレクトリシステムを用いることで、確実で迅速な移行を行うことが出来た。今回の DIT の構築は試験的に WIDE インターネットの 1 ゲートウェイマシンで行なっており、この状態で動作 (接続) の確認を行なっている。そして後日このゲートウェイマシンの DSA をマスターとして会場でスレーブ DSA を立ち上げるという方法をとっている。

2.7.2 DIT の構築

今回構築した DIT には多少問題があった。map,paper などの情報の格納場所として、ou=map などを作成したが、ディレクトリが組織・人情報を検索・参照するためのシステムであるため、map などの特殊なデータの格納先として適当なオブジェクトがなく、複数オブジェクトクラスのエントリを一つのエントリの下に登録してしまい、ユーザの混乱を招くなどの問題を生じた。map などの組織・人情報以外のものに対し、ou(Organizatinal Unit) というクラスを与えるのは統一性に欠ける。

2.7.3 新規属性タイプの作成

本案内システムは、ユーザインタフェースの点からトップダウン式に問題を解決してきたため、全く独自に属性タイプを作成する作業が必要になった。しかし、新規属性タイプを作成して属性の登録を行なったこと自体問題がある。テキストのような単純な情報であれば、既存の DUA を用いて全てのディレクトリユーザが同じ情報を出力・利用することが可能である。しかし、情報が複雑化し標準化も行なわれていない今回の属性では、その出力をサポートした DUA を用意しない限り情報は入手不可能となってしまう。現在は、新規属性タイプの標準化はオフラインで行なわれるが、ローカルに定義した属性タイプの Syntax 自体をディレクトリ中に登録し、それをもとに参照を行えるような仕組みが必要と思われる。

2.7.4 属性構文

これは QUIPU の実装レベルの問題である。

QUIPU では属性構文 (AttributeSyntax) の実装はハードコーディングによって行なっている。この AttributeSyntax とは、例えば、大文字子文字区別なく扱う CaseIgnoreString などのような属性データの扱い方 (具体的にはマッチングのためのルール) を記述したも

のである。INETPOD の searchInfoOnImage 属性は、一つのビットストリーム中にキーワード、座標、サブツリーの DN、検索フィルタタイプといった複数のデータを持つので、特別な Syntax によって記述する必要がある。しかし、かりに ASN.1 コンパイラなどによってこの AttributeSyntax を実装したところで、それを全 DUA に反映することは現実的に不可能である。

2.8　終りに

今回、WG として INET'92 の会場で多数の来場者にディレクトリサービスを使って会場案内サービスを提供する機会に恵まれ、少ない時間と少ない労力で、多々問題は残されているものの、ISODE WG としての基本的なコンセプトを実現したシステムとすることが出来た。この実験を通して、これまでのディレクトリの運用実験から獲得した技術情報の確認や、一時的な目的のために構築する DIT の問題に考察を加えるなど様々な活動に結び付いた。特に、ディレクトリを受身の情報源でなく、情報の提供サイドから積極的にユーザに利用してもらうという形をとったことは、ディレクトリの応用を考えていく上で参考になる例としての意味が大きいと思う。

第 3 章

OSI ディレクトリサービスによるネットワーク情報管理

3.1 はじめに

近年、計算機を相互接続することにより広域なネットワークが構築されている。計算機上ではさまざまな情報が蓄えられ処理されている。このような情報をデータベース化し、ネットワーク環境において互いに共有し利用することでより高度な情報処理を行うことの要求が高まっている。

このような広域分散環境に適した特定目的のデータベースを提供するものに X.500 ディレクトリサービスがある。

ISODE WG では、X.500 ディレクトリサービスにおいて、その第一の目的であるネットワーク情報の管理のためのインタフェースについて考察し、新たな X.500 ディレクトリサービスの利用方法を提案し、X.500 ディレクトリサービスに存在する欠点を補うために、ディレクトリアクセスに際し、広く利用されている NIS のインタフェースを利用する機構の設計を行った。

3.2 背景

現在、計算機同士を接続しネットワークが構築され、計算機上の情報をデータベース化し共有し利用することでより高度な情報処理が行なわれている。そのために、さまざまな分散環境におけるデータベースサービスが提供されている。各種の用途に応じて特定のデータベース機能を提供するのがディレクトリサービスである。

ディレクトリサービスとは、ディレクトリ内の**キー (名前や属性)** と、**その値**との対応関係の情報を提供することである。各種の分散情報を必要とするアプリケーションや管理システムがそれを利用していく。

現在利用されている主なディレクトリサービスを以下に示す。

- BIND [149]
  BIND(Berkley Internet Name Domain) は、DARPA[1]インターネット・ネームサー

[1]米国の国防総省高等研究プロジェクト局 (Defence Advanced Reserch Project Agency)

バを UNIX 上に実装したもので、分散環境に存在するネットワークに関する情報を共有し提供することが可能である。4.3BSD UNIX においては、ホスト名/ドメイン名と IP アドレス/電子メイルのあて先情報の対応つけなどの機能を提供している。

- NIS [150]
  NIS(Network Infomation Service) は Sun Microsystems 社によって開発され、基本的なネットワーク情報をある一定のネットワークの範囲内 (名前空間) で統一的に管理する分散型のデータベース・サービスである。扱う情報としては、パスワード情報、グループ情報、ホスト名と IP アドレス情報などがある。

- NIS+ [151]
  NIS をさらに改良し、管理対象範囲を木構造にして配置することを可能とし、名前空間を広げるなどの改善を加えている。

- X.500 ディレクトリサービス
  OSI (開放型システム相互接続: Open Systems Interconnection) ディレクトリサービスは、CCITT 勧告 X.500 シリーズ/ISO 国際標準 9594 として規定され、OSI プロトコル群の最上位に位置する OSI アプリケーションとして提供されている。OSI ディレクトリサービスの世界的な実験である White Page Pilot Project では、主に組織や人の情報の管理が行なわれている。

これらのディレクトリサービスは、広域分散環境においてそれぞれの用途に適したディレクトリの提供が行なわれている。

しかし、近年のネットワークの規模の拡大により、設計当時の分散環境と合わなくなったために運用が難しくなったり、広域分散に適したものであっても、ディレクトリへのアクセス機能が不十分なために、システムへの応用範囲が非常に狭かったりと、問題も多い。

そこで、広域分散環境に適した X.500 ディレクトリサービスの普及と新たな応用範囲の拡大のために、その構成や問題点を調べ、それを補うために広く利用されている NIS との連係機構を設計した。これによって、NIS の広域分散環境への対応と、X.500 ディレクトリサービスの応用範囲の拡大を試みる。

3.3　ディレクトリサービスの実態

3.3.1　NIS

NIS は、ネットワーク環境において基本的なネットワーク管理ファイルを統一的に管理する分散型のデータベース・サービスであり、Sun Microsystems 社によって開発された [150]。

NIS が管理を行う範囲のことを**ドメイン**といい、計算機の集合である。一つのネットワークは複数のドメインから構成されている。ある計算機は必ず最低ひとつのドメインに属する。

3.3.1.1 特徴

- 計算機の運用に必要なデータベース・ファイルの共有機能と自動更新機能を提供している。
- 共有される情報以外に、ローカルな情報の提供も可能である。

3.3.1.2 NIS の構成

NIS は、データベース・ファイルを集中管理している NIS サーバと、それを利用する NIS クライアントという構成の**サーバ・クライアント方式**で実現されている。

3.3.1.2.1 NIS サーバ データベースを集中管理し、クライアントからの要求に対して情報を提供する機能または計算機。対象となるファイル (`/etc/passwd`, `/etc/host` など) は、データベース化しマップとして格納する。

しかし、サーバが情報を集中管理するために、なんらかの原因でサーバが利用できなくなると、サーバに頼っている全てのクライアントが動作できなくなる。これを回避するためにはサーバを複数台用意しなくてはならない。このために次の問題点を解決する必要がある。

- 複数の NIS サーバ間でのデータベースの一貫性の確保
- 使用中の NIS サーバが利用できなくなった時のサーバの切り替え方法

複数のサーバに対してデータベースの一貫性を保つために、NIS は**マスターサーバ**と**スレーブサーバ**という 2 種類のサーバを用意している。

- マスターサーバ
  全てのマップを管理している。マップの更新はこのサーバ上のみで行なう。
- スレーブサーバ
  マスターサーバ上で管理されているマップのコピーを保持する。マップの更新は行なえないが、NIS クライアントに対してマスターサーバと同様に情報の提供を行う。

一つのドメイン上には必ず一つのマスターサーバが存在し、この一つのマスターサーバに対して 0 個以上のスレーブサーバを設定できる。

マスターサーバ上でマップの更新が行なわれると、マスターサーバはただちにスレーブサーバに対してそのマップファイルを送り、両者でのデータベースの一貫性を保証している。マスターサーバがダウンしたとしても、データの更新が行なえないだけで、データベースの情報の提供機能と一貫性は保証される。

3.3.1.2.2　NIS クライアント　NIS サーバと通信して必要な情報を受け取るアプリケーションプログラムあるいは計算機。

3.3.1.3　NIS の動作

3.3.1.3.1　デーモン　NIS には 2 つの重要な役割をするデーモンがある。

- ypserv
  NIS サーバとなる計算機で起動されるデーモン。このデーモンの仕事は大きくわけて 2 種類に分けられる。
  - NIS サーバとして持っている情報をマップ化する。
  - NIS クライアントからの要求を待ち、要求を受け取ったらそれにしたがって NIS マップを検索し情報を提供する。サーバとクライアント間の通信には Sun RPC (Remote Procedure Call) を用いている。
- ypbind
  ネットワーク上の NIS サーバを見つけるために、NIS クライアントとなる計算機で起動されるデーモン。NIS クライアントのアプリケーションは、ypbind に対して NIS サーバ (ypserv) がどれなのかを問い合わせ、返された IP アドレスを受け取り、サーバに要求を出す。ypbind は NIS サーバを見つける時にブロードキャスト通信を用いる。

3.3.1.3.2　情報の問い合わせ手順

1. NIS サーバ計算機では ypserv、NIS クライアント計算機では ypbind が起動される。
2. ypbind はブロードキャスト通信を用いてネットワーク上にある ypserv に対してメッセージを送り、最初に応答を返して来た ypserv をその計算機で利用する NIS サーバと決め、NIS サーバの計算機の IP アドレスを保存する。
3. NIS クライアントのプログラムは、ypbind に対して NIS サーバの IP アドレスを問い合わせる。
4. 問われた ypbind は ypserv に対して利用が可能かを問い合わせ、応答があればその NIS サーバは利用可能と判断する。応答がなければ、再度ブロードキャスト通信を用いて ypserv を見つける。
5. ypbind は、ypserv の動いている計算機の IP アドレスを NIS クライアントのプログラムに返す。
6. NIS クライアントのプログラムはその IP アドレスを使用し、RPC を用いて ypserv と通信し、情報を得る。

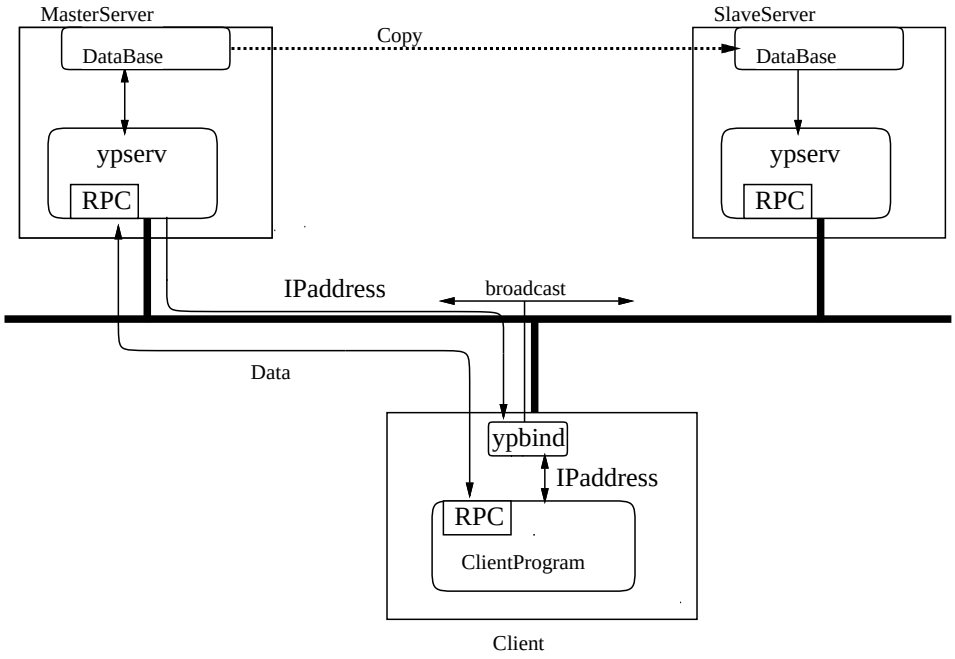

図 3.1: NIS 問い合わせ手順

このようにして、使用中のサーバになんらかの障害が起きても、クライアント側は別のサーバを自動的に探すことができる。

3.3.2 NIS+

NIS+は NIS の欠点を補う目的で作られた。以下にその特徴をあげる。

- ドメインの階層化
  ドメインを木構造として配置し、木構造の名前空間をアクセスできる。また、シンボリックリンクで別の部分の部分を指し、別名を提供することも可能である。

- データの構造化
  NIS マップは縦 2 行の表であったが、その両方で検索できるように 2 つのマップが生成されていた。NIS+では、これを 1 つのマップで検索できるようになっている。また、表の中が構造を持っているため、表の中から別の表を参照することもできる。

- セキュリティ強化
  アクセス制御と認証確認を導入し、作成・削除・変更・読み出しのアクセスタイプに対し、各々所有者、グループ、全てに許すか拒否するかというアクセス権を持っている。また、公開鍵暗号法を用いて、クライアントの認証確認を行なっている。

これによって、ユーザーレベルにおいては、yppasswd などの一部のコマンドでしか行なえなかったデータの更新が可能になった。

3.3.3　X.500 ディレクトリサービス

3.3.3.1　ディレクトリサービスの特徴

OSI ディレクトリサービスは以下の 3 点を仮定しており、これにより汎用データベースとは異なっている。

- 検索 (ディレクトリの読み出し) は更新 (ディレクトリへの書き込み) に比べ頻度が高い。
- ディレクトリの情報の一貫性は、ある任意の時点では保証することはできない。
- 名前付けには関係的というよりは、階層的な方法が利用される。

3.3.3.2　ディレクトリへの問い合わせ

ディレクトリに対し必要な情報を問い合わせる **ディレクトリの利用者**は、ディレクトリを利用するシステムプロセス、もしくはアプリケーションプロセスである。ディレクトリへのアクセスは **DUA(Directory User Agent)** により行なわれる。DUA が利用者に代わってディレクトリと通信を行い、ディレクトリの内部構成は利用者から完全に隠蔽されている。

3.3.3.3　ディレクトリの保持情報

ディレクトリが保持する情報は、ディレクトリ情報ベースとそれを関係づけるディレクトリ情報木によって規定される。

3.3.3.3.1　ディレクトリ情報ベース:DIB　ディレクトリが保持する情報全体をディレクトリ情報ベース (DIB: Directory Information Base) という。ディレクトリで操作される全ての情報は DIB に含まれている。

DIB はオブジェクト[2]から構成され、これらの情報を**エントリ**という。各エントリは 1 つ以上の**属性**から構成される。各属性は 1 つの**型**と 1 つ以上の**値**から構成される。

OSI ディレクトリでは共通するオブジェクトの型を定義しており、利用者が自分のオブジェクトを定義することもできる。オブジェクトの定義はオブジェクト指向におけるクラス継承の概念に基づいて行なわれる。あるオブジェクトは、より基本的な属性を持つオブジェクトのサブクラスとして、差分を詳述する形式で定義することができる。サブクラスとして新しく定義されたオブジェクトは、新しく定義された属性の他に、上位クラスで定義されたすべての属性を継承して使用することができる。

[2]ディレクトリの扱う情報の一単位のこと

ディレクトリは標準で多くのオブジェクトクラスと属性を定義している。OSI アプリケーションも新しいオブジェクトクラスや属性を定義することができる。

3.3.3.3.2 ディレクトリ情報木:DIT エントリはディレクトリ情報木 (DIT:Directory Information Tree) と呼ばれる階層構造を用いて、互いに関係付けられている。図 3.2に OSI ディレクトリの名前空間の例を示す。

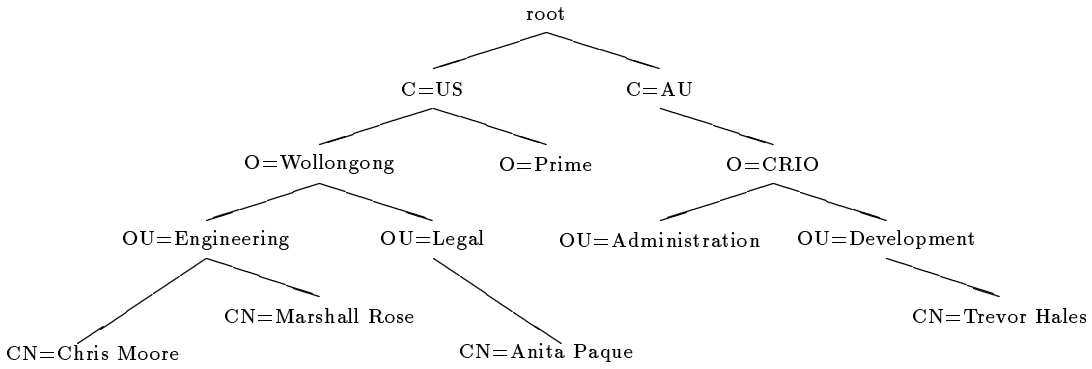

図 3.2: OSI ディレクトリの名前空間の例

節点である各エントリは、DIT におけるエントリの位置を示す**識別名 (DN: Distinguished Name)** により一意に識別される。例えば図 3.2における Anita Paque の DN は以下のようになる。

```
C=US,O=Wollongong,OU=Legal,CN=Anita Paque
```

また、各エントリは**相対識別名 (RDN:Relative Distinguished Name)** と呼ばれる単一値の属性を持っている。たとえば、属性が `countryName` と呼ばれるもので、JP という識別値を持っているとすると、そのエントリに対する RDN は

```
countryName = JP
```

となる。2 つのエントリの直接の上位エントリが同じであるなら、RDN は互いに異なっていなければならない。

3.3.3.4 ディレクトリの機能モデル

ディレクトリは 1 つ以上の **DSA(Directory System Agent)** から構成されており、その内部的な相互動作は 4 つの**機能モデル** に分類される。

ディレクトリのアクセスにおいて、DUA は 1 つ以上の DSA と通信を行なう。最も簡単な要求応答の相互動作は、要求された情報を最初にアクセスされた DSA が持っている場合である (図 3.3)。

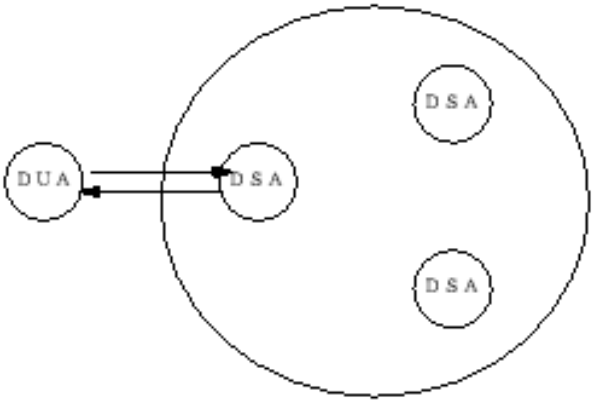

図 3.3: 機能モデル

DSA が要求された情報を持っていない場合、DSA は DUA に**委託 (referral)** を返す (図 3.4)。委託にはなんらかの形で、DUA が要求した情報に近い情報を持つ他の DSA のプレゼンテーションアドレスを含んでおり、DUA はそのアドレスが示す DSA と通信を行い、通常の要求応答型の相互動作を行なう。

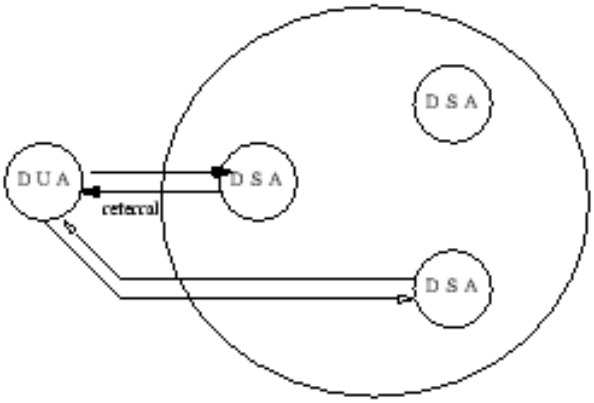

図 3.4: 機能モデル – Referral

あるいは、最初の DSA が要求した情報を持っていなかった場合に、この DSA が直接 2 番目の DSA に接続する (図 3.5)。この型の機能モデルを**連鎖 (chaining)** という。

または最初の DSA が、要求された情報を見つけるために複数の DSA と接続する。この場合、並列に処理が行なわれる。このような機能モデルを **Multicasting** という (図 3.6) 。

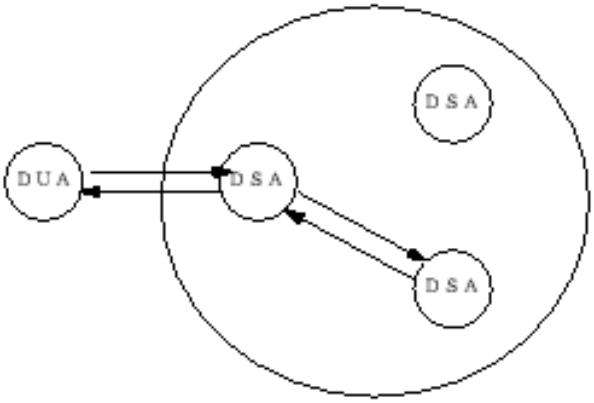

図 3.5: 機能モデル – Chaining

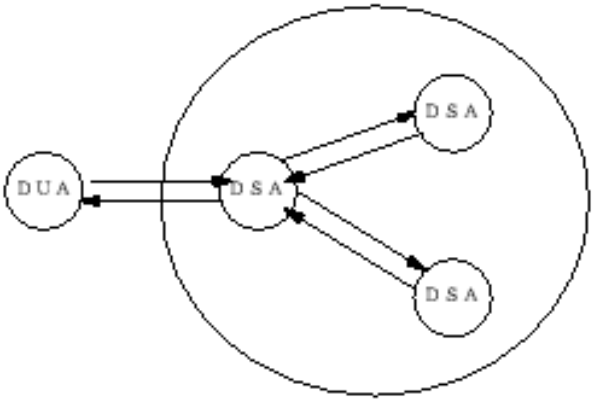

図 3.6: 機能モデル – Multicasting

3.3.3.5 ディレクトリの構成

ディレクトリは複数の DSA により構成されているが、ある DSA がディレクトリ・ツリーのどの部分ツリーをどのように管理しているかは、**ディレクトリ管理ドメイン (DMD: Directory Management Domain)** により定義される。これは、複製とアクセス制御の問題を含んでいる。DMD は以下のものにより構成される。

- ディレクトリ・ツリー (DIT) の部分ツリーをまとめて保持する 1 つ以上の DSA
- 0 個以上の DUA
- DMD 内の複数の DSA のうち、外部の DMD に対してどの DSA から見せていくかという DMD の外部動作の定義

3.3.3.6　セキュリティモデル

ディレクトリのセキュリティは、**認証 (authentication)** と**承認 (authorization)** によって規定されている。

- 認証

  ディレクトリの DSA と利用者を識別するために利用される。**なし、弱い、強い**の 3 種類の認証がある。弱い認証は、パスワードによる防御に基づく。強い認証は暗号の公開鍵に基づいて行なわれる。

- 承認

  アクセス制御リストを利用して、アクセス権を明確に、施行し、管理する。

標準のディレクトリでは、実際にはセキュリティの方式を定義しておらず、独自のセキュリティ方針を採用することができる。

3.3.3.7　ディレクトリのサービス

ディレクトリがユーザに提供するサービスには次のようなものがある。

- 質問要求: DIT または DIB に関する情報を返す
- 更新要求: DIT の構造や DIB の内容を更新する

ディレクトリは各要求に必ず応答を返す。その応答は、検索の結果またはエラーの指示である。通常の結果では、その形式は要求に固有のものとなる。

3.3.3.7.1　質問要求　DIT や DIB に対して 4 種類の質問要求がある。

- 読み出し
  特定のエントリを目的とし、そのエントリのいくつかまたは全ての属性を返す。DUA が 1 つまたは複数の属性を知りたい場合、読み出し要求によって指定できる。要求時に属性を指定しなければ、全ての属性が返される。

- 比較
  特定のエントリの特定の属性を目的とし、DUA が提示した属性値とエントリの属性値の比較を行なう。たとえば、パスワードチェックであり、この場合、ディレクトリが持っているパスワードを読み出しすることはできないが、比較はできる。

- リスト
  DIT の特定の名前を持つエントリの直接下位のエントリの相対識別名 (RDN) の一覧を返す。

- 検索
  検索のパラメータを、特定のエントリから始まる DIT の部分木の属性に適用し、検索基準を満足するエントリの名前を返す。

さらに、要求を中断するための要求がある。

- 中断要求
  未完了の要求に対し適用される中断要求は、ディレクトリに対し、要求発信者が、もはや、その実行されている要求に興味を持ってはいないということを知らせる。ディレクトリは、要求の処理を止めたり、それまでに成し遂げられた結果を捨てたりする。

3.3.3.7.2 更新要求 DIT の構成や DIB の情報を更新するために、4 種類の要求がある。

- エントリ追加
  DIT に新しいエントリを追加する。追加するエントリは末端エントリに限られる。

- エントリ削除
  DIT からエントリを削除する。削除するエントリは末端エントリに限られる。

- エントリ修正
  特定のエントリに対する一連の変更を実行する。その全てを変更してもかまわない。属性または属性値の追加、削除、置換えが可能である。

- RDN の変更
  エントリの RDN を変更する。RDN を変更するエントリは末端エントリに限られる。エントリは DIT における位置を変えないので、移動要求ではない。

3.3.3.7.3 サービス制御 様々なサービスに対して適応できるいくつかの制御がある。これは、ユーザが資源の使用について制限を定めることができるようにする。サービス制御には次のようなものがある。

- 時間に関する制御：結果の返ってくるまでの時間を指示したり、無制限にできる。

- 結果のサイズに関する制御:結果の数を制限する、または無制限などの指示をだせる。

- 検索の範囲に関する制御：1 階層だけ、ある階層から下を全部、などを指示することで検索の範囲を限定でき、待ち時間の短縮ができる。

- 相互動作モードに関する制御：要求が解決できない時に連鎖 (Chaining) を行なうかを指定する。

- 要求の優先順位に関する制御

3.3.3.7.4　エラー　どのサービスも失敗する可能性がある。たとえば、ユーザが与えたパラメータに問題がある場合や、セキュリティ方針、サービス制御などによって生じる。問題を修正する助けとなるように、可能であれば、エラーとともに情報が返される。

また、DUA が接している特定のアクセス点を持つ DSA が、独自に要求を解決できず、かつ、連鎖 (chaining) によって要求処理を実行できない場合、委託 (referral) が返されることがあるがディレクトリはこれを、ある種のエラーとして扱っている。

3.4　NIS および X.500 の問題点

3.4.1　NIS の問題点

- データの分散管理ができない。
  一つのドメイン内の情報の更新管理は、マスターサーバ 1 台でしか行なえない。しかし、大規模分散環境においては、管理者は複数おり、それぞれローカルで管理したい情報があっても、NIS では実現ができない。

  NIS+によって、情報の更新管理はユーザーができるようになり、ドメインを階層化することにより情報管理も階層化され分散管理できるようになったが、扱う情報は必ずドメインの中に存在しなくてはならないため、世界的に共通な情報の管理も結局はドメインの管理者が行なわなくてはならない。

- 一つの NIS ドメインの範囲に制限がある
  ypbind は ypserv を探す際にブロードキャスト通信を用いているために、ypserv は同じネットワーク上 (IP 的にブロードキャストが可能なネットワーク内) になくてはならない。複数のサブネットワークにまたがって NIS ドメインを構成したいのであれば、 NIS サーバがゲートウェイを介して通信を行なえるように設定しなければならない。

3.4.2　X.500 ディレクトリサービスにおける問題点

- アプリケーションの不足
  DUA としてディレクトリにアクセスするアプリケーションは、その多くが人間が手で操作を行なうものを仮定したものであり、他のアプリケーションから利用したり、UNIX におけるさまざまなデータベースをアクセスするためのプロトコルに対応したものは少ない。そのため、応用範囲が狭い。

- データの一貫性と情報の伝搬速度
  X.500 の扱うディレクトリは 3.3.3.1に挙げた仮定に基づいている。すなわち、ディレクトリサービスにおいて、データの一貫性が保証されておらず、2 つの異なる DSA から同時に得られる情報がかならずしも一致しない事を示している。これは、ディ

レクトリでは更新した情報が穏やかに伝搬していくために情報に遅延が生じるからである。したがって、情報の一貫性を必要とするものには不適当である。

3.5 NIS との連係

X.500 ディレクトリサービスの応用の一つとして、また、3.4で挙げた問題点を解消するために、本研究では X.500 ディレクトリサービスを NIS のインタフェースにより利用できるようにする。

X.500 ディレクトリサービスと NIS との連係を行なう事により、以下のような利点が考えられる。

- X.500 ディレクトリサービスを提供するアプリケーションの拡大
  DUA としてディレクトリに問い合わせを行なうのに、専用のアプリケーションで実際に人が手で問い合わせを行なうものが主であった。だが、NIS との連係で、多くの計算機上で現在動いている NIS を利用したデータベースを提供するアプリケーションを全くの無変更で、X.500 ディレクトリサービスが利用できるようになる。

- NIS の提供するデータベースの拡大
  いま現在多くの計算機において NIS によるデータ検索が可能である。NIS が X.500 ディレクトリサービスと連係をはかることにより、広域分散のデータベースが利用可能になる。

- NIS の欠点を補う
  NIS は 3.4.1で述べたような欠点を持ち合わせている。また、 NIS+には広域分散環境にかなり適した構成になった。しかし、複数のドメインをまたぐようなデータのサービスの提供など、若干の問題も残されている。NIS と X.500 ディレクトリサービスとの連係で、それらの問題も補った広域分散に適したディレクトリサービスが可能になる。

- X.500 ディレクトリサービスのデータの拡大
  X.500 ディレクトリサービスにおいて、今までは主に人や組織の情報を扱ってきたが、NIS の扱うデータを扱うことによって、X.500 ディレクトリサービスの扱うデータの幅が広がり、新たな応用が見い出せる。

3.6 設計

3.6.1 基本的な設計方針

X.500 と NIS の連係を実現するにはいろいろな方法があるが、本研究では ypserv を DUA として機能するようにする。

このようにすることで、NIS クライアント側は無変更で X.500 ディレクトリを利用することができる。必要な情報のサーバとなる DSA を構築し、ypserv を置き替えるだけで、クライアント側は何の変更もなく今まで以上のディレクトリサービスが受けられる。

また、X.500 側も既存の DSA をそのまま生かすことが可能となる。

3.6.2　DUA 機能を持った ypserv の動作

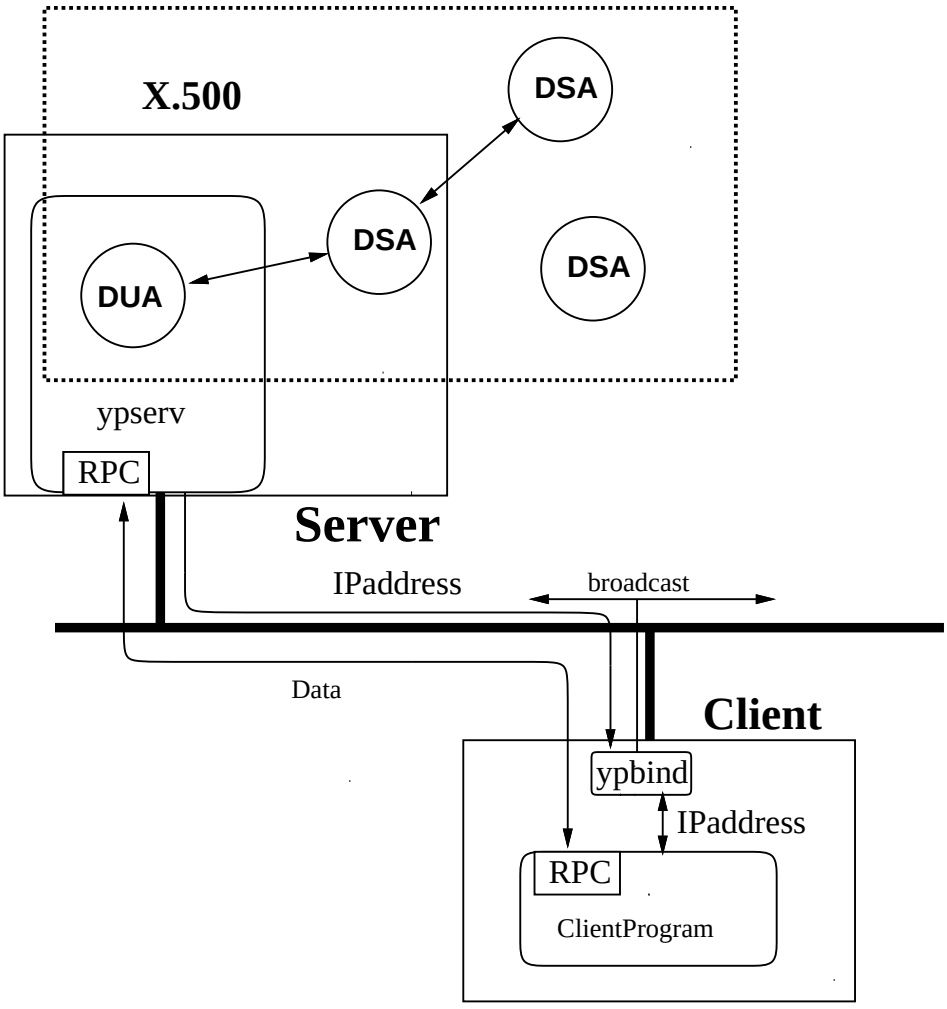

図 3.7: NIS と X.500 の連係

NIS クライアントは 3.3.1.3.2で説明したように、NIS サーバに問い合わせを行なうが、ypserv は必要であれば、DUA として DSA から情報を得るようにする (図 3.7)。

3.6.3 NIS の情報と X.500 ディレクトリサービスとの対応

3.6.3.1 階層的な管理法

/etc/services

/etc/services はインターネット上で利用できるサービスのサービス名とプロトコル名、ポート番号の対応表である。

```
iso-tsap 104/tcp tsap
```

これはサービス名 iso-tsap，tsap がプロトコル名 tcp、ポート番号 104 で利用できることを表している。

NIS の情報に対して新たなエントリを追加する。その中で、世界共通で使用しているものは DIT の最上位で定義し、以下階層が下がるごとにローカルな設定を追加していくこととする (図 3.8)。

```
"@c=JP@o=University of Electro Communications@ou=PDL@cn=NIS@cn=services"
   services= 1178/tcp:skkserv &
    2003/tcp:cfinger &
        :
"@c=JP@cn=NIS@cn=services"
   services= 1/tcp:tcpmux &
    7/tcp:ech &
        :
```

図 3.8: services の定義例

検索時は最下層から上に向かって検索を行なう。例えば最初に"@c=JP@o=University of Electro Communications@ou=PDL@cn=NIS@cn=services"を検索し、必要なものが見つからなければ上位の DIT を検索していくことでことで必要なものが見つかる。

これにより、世界中の計算機内に存在している同じ情報が、効率良く階層化され配置される。

3.6.3.2 分散的な管理法

/etc/hosts 等の管理

/etc/hosts はホスト名と IP アドレスとの対応表である。

```
130.153.128.32 bigbird.cs.uec.ac.jp bigbird
```

これはホスト名 bigbird.cs.uec.ac.jp, bigbird と IP アドレス 130.153.128.32 が対応していることを示す。

これにも対応する新たなエントリを追加する。 例えば、NIS の hosts.byaddr などは以下のようなエントリに定義する。

```
"@c=JP@o=University of Electro Communications@ou=PDL@cn=NIS@cn=hosts"
    hosts2addr= bigbird.cs.uec.ac.jp:130.153.128.32 &
    twoheaded.cs.uec.ac.jp:130.153.128.25 &
    penelope.cs.uec.ac.jp:130.153.128.10
```

これは、IP アドレスからホスト名への対応を記述したもので、NIS サーバーにおいて、hosts.byaddr を以下のように記述したことと解釈する。

```
130.153.128.32        bigbird.cs.uec.ac.jp bigbird
130.153.128.25        twoheaded.cs.uec.ac.jp twoheaded
130.153.128.10        penelope.cs.uec.ac.jp penelope
```

/etc/passwd の管理

/etc/passwd はユーザーのログイン名とパスワードなどとの対応表である。

```
shigeo:bULuI2Pv7bONc:9618:96:Sakuma Shigeo:/home/pdl/shigeo:/usr/local/bin/bash
```

各項目は ":" で区切られており、それぞれ、ユーザー名、暗号化されたパスワード、ユーザー ID、グループ ID、gcos フィールド、ホームディレクトリ名、ログインシェルを表している。ユーザー名、ユーザー ID をキーにして検索される。

広域なネットワークにおいて、複数のドメインにアカウントを持つ人が自分のアカウントのパスワードなどを自分で管理する場合などが考えられる。

- /etc/passwd のエントリの記述
  ユーザーネームの先頭に "=" を持つエントリは X.500 で管理されているものとする。識別名 DN は/etc/passwd 内の ":" で区切られた 5 番めのフィールドである gcos フィールドに記述する。
  例

  ```
  =inada:*:9700:101:@c=JP@o=WIDE@ou=ISODE WG@cn=Ryu Inada: /home/isode:/bin/csh
  ```

- DIT のエントリ

  分散管理されたユーザーは自分のエントリの下に cn=NIS@cn=passwd といったエントリを作成し、そのエントリ中に自分の暗号化されたパスワードやログインシェルなどを記述する。

  例

```
"@c=JP@o=WIDE@ou=ISODE WG@cn=Ryu Inada@cn=NIS@cn=passwd"
        passwd=YYYYYY
        loginshell=/bin/bash
```

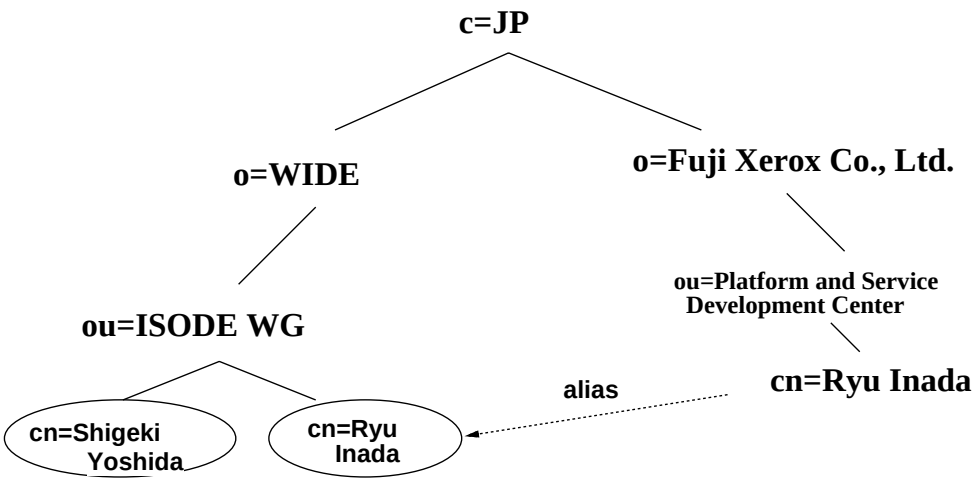

図 3.9: ISODE WG のプロジェクトの DIT の形

- 検索アルゴリズム
  /etc/passwd エントリ中で、“=” で始まるユーザーネームはエントリにおける gcos フィールドに記述されているエントリを検索する。そのエントリから passwd の情報作成に必要な属性値を取り出し、新たな passwd エントリを生成する。NIS サーバーはその生成したエントリをクライアントへ提供する。

  上記の設定例で、ユーザー名 inada でログインする場合

  1. inada のユーザー名は/etc/passwd において、“=” で始まっている。
  2. ypserv は DUA として gcos フィールドにかかれている
     “@c=JP@o=WIDE@ou=ISODE WG@cn=Ryu Inada”
     の下の DIT を検索し、パスワードエントリに必要な情報を得る。
  3. “@c=JP@o=WIDE@ou=ISODE WG@cn=Ryu Inada” にマッチするので、ここから “passwd=YYYYYY” と “loginshell=/bin/bash” より新たなエントリ

     ```
     inada:YYYYYY:9700:101:c=JP@o=WIDE@ou=ISODE WG@cn=Ryu Inada:/home/isode:\
     /bin/bash
     ```

     をつくり出す。これによって、ログイン名 inada で、X.500 ディレクトリサービスで管理を行なっているパスワードとログインシェルでログインすることが可能になる。

以上より、/etc/passwd に対して広域なネットワークにおいて複数のドメインにアカウントを持ったとしても、パスワード等を自分の特定の DSA 一箇所で管理することが可能になる。

3.7　利点と問題点

3.6の実装をすることにより、以下の X.500 および NIS に対する利点と問題点が予想される。

3.7.1　利点

- X.500 に対する利点
  X.500 ディレクトリサービスにおいて、今まではそのために作られた検索のアプリケーションを手で入力し検索するものが主であった。多くの計算機で稼働している NIS に対応したアプリケーションやライブラリを利用することで、X.500 ディレクトリサービスを利用することが可能になる。

- NIS に対する利点
  NIS+によって、ドメインを階層的に配置することで管理が分散され改善はされた。しかし、この実装でも/etc/services などのような世界共通で使用している情報とローカルな情報が混在するものに対して、それぞれの管理者が管理すべき情報を明確にし、情報の階層化を行なえる。また、世界的に共通な情報に関しては、ドメイン内に縛られず管理するのが適当な場所へ分散できる。

  また広域分散環境でのアカウント管理においては、自分のパスワード等を特定の DSA で管理できる。そして、ログイン名がそれぞれのドメインで異なっていたとしても、そのユーザが誰であるのかは、/etc/passwd に記述した X.500 ディレクトリサービスのエントリによって一意に特定できる。

3.7.2　問題点

- X.500 による問題点
  この NIS サーバを運用し広域分散環境でアカウント管理を行なうと、暗号化されているとはいえ、ユーザーのパスワードをエントリから参照しなくてはならない。セキュリティ的に安全な DUA の認証とアクセス制御を実装しなくてはならない。

  また情報の伝搬が穏やかであるので、更新した情報がすみやかに伝わらない。NIS 自身で扱う情報もそう頻繁に更新するものではないが、一度更新した情報がなかなか反映されないのは、特にパスワード管理などでは問題である。

  検索速度の対策として、ローカルな情報や頻繁に参照され更新がされにくいようなデータに対しては X.500 ディレクトリサービスのディレクトリをアクセスするので

はなく、すぐに情報が引き出せる形としてディレクトリサービスから独立した所に持つ必要があるだろう。

- NIS による問題点
  クライアントのディレクトリのアクセスに対しては、NIS のプロトコルを使用している。これはクライアント側の変更が全く必要ないが、その反面、3.4.1であげたように NIS サーバの配置の制限がそのまま残ってしまう。

3.7.3 DNS との比較・評価

DNS とは Doman Name System の略で、インターネット全体にわたるホスト名と IP アドレスとの対応を階層的に管理するシステムのことである。階層の各レベルにおいてはネームサーバによって管理されている。これはちょうど、DSA が階層的に配置されそれぞれの情報を管理していることに対応できる。本研究の実装でも、ホスト名と IP アドレスの対応の管理を試みている。

しかし決定的に異なるのは、DNS の階層化がそのままドメイン名の階層に対応しているが、X.500 ディレクトリサービスによる階層化はドメイン名には全く対応していないことである。

これは検索の効率に対して差が生じることになる。DNS はドメイン名を解釈し、効率的に分散されたサーバを検索していくのに対し、X.500 ディレクトリサービスにおける検索は確かに必ずドメイン名と IP アドレスは見つかるが、3.3.3.4で説明したような機能モデルを用いて情報検索を行なうので、決して効率的な検索とはいえない。これに対しては検索効率の良い DIT への格納方法や、DSA の配置を効率的に考えなくてはならない。

3.8 結論と今後の課題

X.500 ディレクトリサービスと NIS を連係させることで、X.500 ディレクトリサービスの分散環境を NIS の機構に取り込むことができる。これにより、扱えるデータの量が飛躍的に増大する。そして、NIS+よりもより柔軟な分散管理が可能になる。

また、X.500 ディレクトリサービスのアクセス方法に NIS のインタフェースを使用することにより、ディレクトリにアクセスすることが可能なアプリケーションとして NIS のものをそのまま利用できるために、X.500 ディレクトリサービスを提供できるアプリケーションが増大した。

しかし、X.500 ディレクトリサービス自身の検索速度は決して速いものではない。DSA にはキャッシュ機能があり情報を内部的に保存するが、それ以外にも X.500 ディレクトリから得た情報を NIS サーバ内のハッシュ表に取り込むような機能も必要であろう。

今後は、まずここで議論した X.500 ディレクトリサービス機能を持った NIS サーバの実装と評価を行なう。さらに、新しく採り入れた NIS の情報をより効率良く配置できるような格納の仕方を検討し、検索に対してより効率のよいものを調べ配置する。

第 4 章

MHS の調査と基礎実験

4.1 背景と目的

WIDE ISODE WG は、TCP/IP プロトコルより OSI への移行の方法を探る目的で現在活動をしている [152]。前年度は、X.500 Directory Service[49] の全国実験を行ない日本における DSA 網をたちあげた [153][154][155][156]。

今年度から、新しいアプリケーションとして **X.400 MHS**[157] を対象として、TCP/IP ベースの電子メイルシステムからの移行についての研究を開始した。

ISODE WG では、MHS システムとして、ISODE (ISO Development Environment) をベースとした X.400 MHS の実装である **PP** を使用して研究・実験を進めている。研究・実験に使用した PP の現在のバージョンは 6.0 である[1]。現時点で、PP が動作しているのは富士ゼロックス/日立ソフトウェアエンジニアリング/富士通研であるが、残念ながら各々別々に動作している状態で、相互接続は行なっていない。

本報告の目的は

1. X.400 についての理解を得る
2. PP における X.400 の実装の概要を説明する

の二点である。

4.2 X.400 とは

X.400 は、OSI における Mail Service を規定したものであり、サーバ・クライアントモデルを採用している。

X.400 では「メイル」は「メッセージ (Message)」と称せられ「エンベロープ (Envelope)」と「コンテント (Content)」よりなる。

エンベロープはいわゆる「封筒」であり、メッセージの転送に必要な情報 (例えば、宛先とか発信日付/速達とか) を運ぶ役目を持っている。

[1] ISODE Consortium 版の PP は 6.0+αということである。また、ISODE Sonsortium 版とは別に PP 7.0 がちかじかリリースされる予定である。

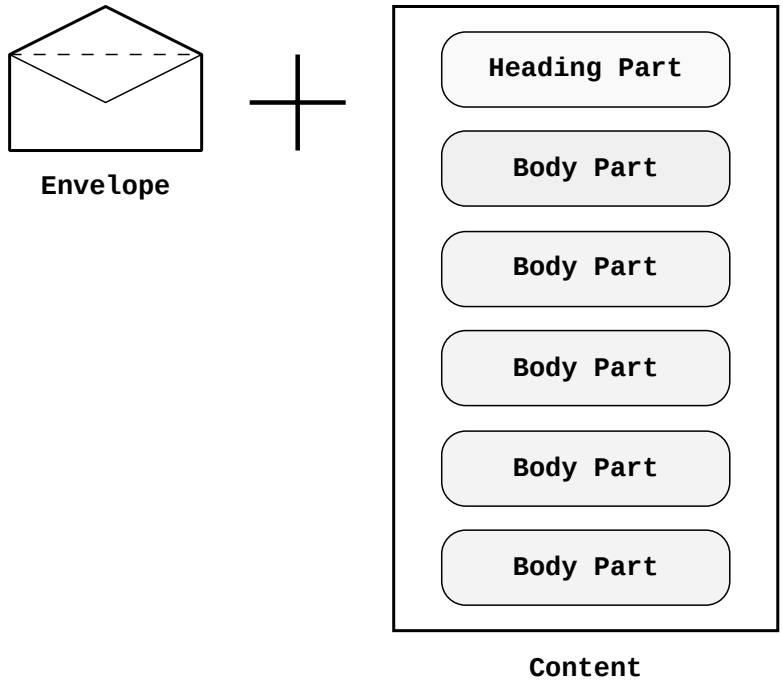

図 4.1: メッセージの構造

「コンテント」は、発信者が受信者に送信する情報で原則として宛先にトランスペアレントに送られる。

「コンテント」の中身は「ボディパート (Body Part)」と呼ばれるものの集合体である。一番はじめの「ボディパート」は特殊な用途で用いられており、特に「ヘッデイングパート (Heading Part)」と呼ばれている。

メッセージの構造を図 4.1に記する。

「ボディパート」には、実際に送る情報を格納する。

ボディパートには、いくつかの種類があり、種類毎に「何が」「どのように」エンコードされているか規定されている[2]。

「ヘッディングパート」には、宛先 (To/Cc)、件名 (Subject) 等のいわゆるヘッダの情報が格納される。

以後、OSI の規定にしたがってメイルのことを「メッセージ」と記述する。

X.400 MHS は、図 4.2のようなモデルを採用している。

Mail Service は、**MTA/MS(Optional)/UA/AU(Optional)** の 4 つのサービスにわかれており、各々

[2]例えば ia5 などがある。ia5 とは ISO646IRV(通常の ASCII キャラクタと思えばそれほど外れていない) を使った構造を持たないテキストである。

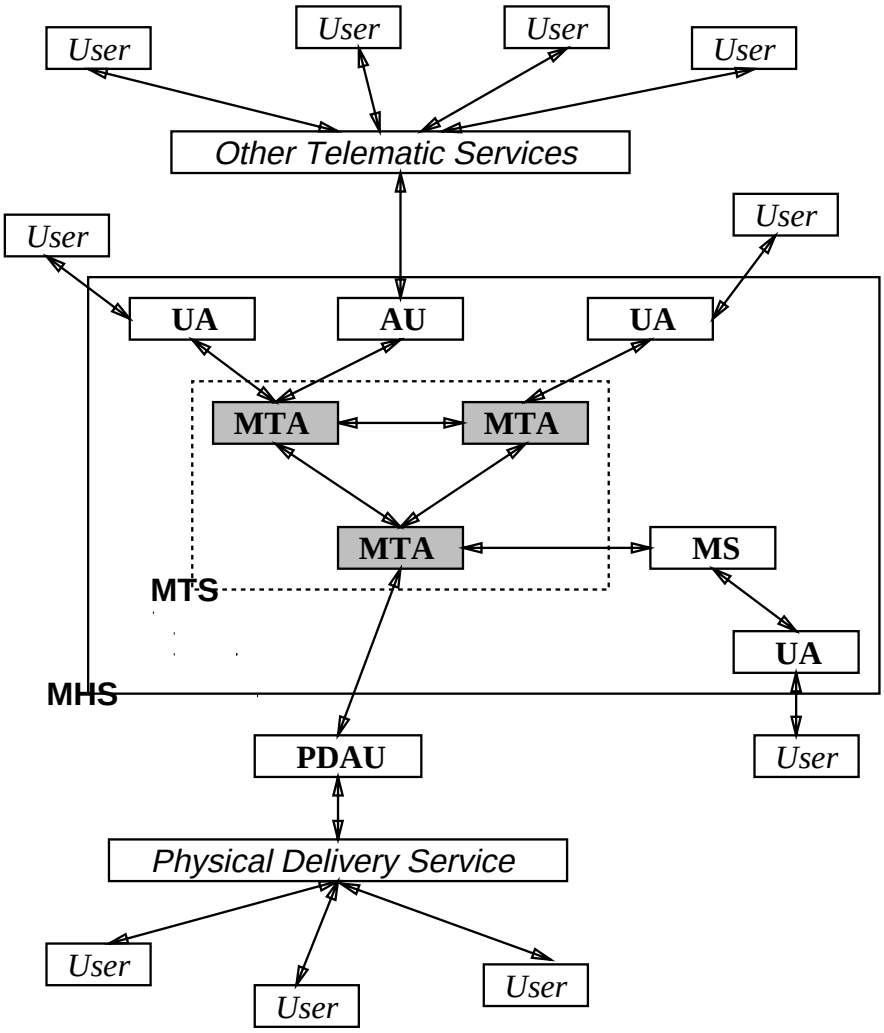

図 4.2: MHS のモデル

MTA *(Message Transfer Agent)*	メッセージの中継機能を持つ
MS *(Message Store)*	MTA からユーザに送られたメッセージを保持しておく機能を持つ (1988 年版で追加)。いわゆるメイルボックスに相当
UA *(User Agent)*	ユーザと MTA 間の仲立ちを行なう
AU *(Access Unit)*	MHS を間接的に使う時のインタフェース (1988 年版で追加)

にわかれている。

この他にも、**PDAU** *(Physical Delivery Access Unit)* と呼ばれるものがある。これは「物理的な配送手段」を示し、例えば郵政省の物理的なメイルサービス (郵便) までもが MHS の範疇に入っている。日本の場合は、郵便法による郵便制度と電気通信事業法によ

る電気通信サービスは制度的にはっきりと区別されてるが、ヨーロッパをはじめとした多くの国々では、これらのサービスが一つの公共事業体が行なっている例が多いことが背景のようである。日本でも、MHS のデモとして PDAU を使った例がある。具体的には、MHS で出した電子メイルを紙に打ちだし郵便で送るといったデモを行なった。

TCP/IP における sendmail は X.400 MHS における MTA の機能と UA の機能を兼ね備えていることが図 4.2を見ればわかる。

sendmail は、通常 MH などのプログラム[3]からのメッセージをパースし、必要に応じて別のマシン上の SMTP Server[4]に送る。この SMTP Server に送る部分が MTA の機能となる。

4.3 PP

4.3.1 PP の構成

PP は、ISODE をベースとした X.400 MTA の実装である。しかしながら、PP は X.400 MTA のみならず AU をも含んだ実装となっており、実際開発元である *University of Colladge London* では Mail Gateway はこの PP を用いている。PP の実装は図 4.2中の MTA/AU の部分を実装したものであり、UA/MS は含まれていない[5]。

PP は Queue を一つ持っており、この Queue を管理するのが *qmgr* と呼ばれるプロセスである。必要に応じて起動され各種のプロトコル処理をおこなう *Channel Program* の集合体である。このような構造を取ることにより、容易に各種プロトコルに合わせた処理を容易に追加可能である[6]。

PP を構成するプロセスの構造を、図 4.3に示す。

QMGR は、PP のすべてのオペレーションのスケジュールを行なう。QMGR はメモリー内にすべての Queue の状態を記憶し、作業を行なうプロセスを制御する。QMGR にこの情報を集中することにより有効なスケジューリングと管理オプションが得られ、パフォーマンスも改善される。QMGR が主に制御するものは *Channel* である。Channel には、

1. ローカルもしくはリモートからのメッセージ転送プロトコルを利用してメッセージの配送[7]

2. Queue 内のメッセージの処理[8]

---

[3] UA という観点から見れば、MH/Mail 等と sendmail の一部が UA であろう。

[4] sendmail とは限らない

[5] PP 7.0 では X-Window ベースの UA が実装されているそうである。

[6] 実際 PP 5.2 より PP 6.0 へのバージョンアップの際 DEC Net 用の Channel Program が追加された。このプログラムは Contribute Software である。

[7] X.400 P1/SMTP 等

[8] メッセージのフォーマット変換など

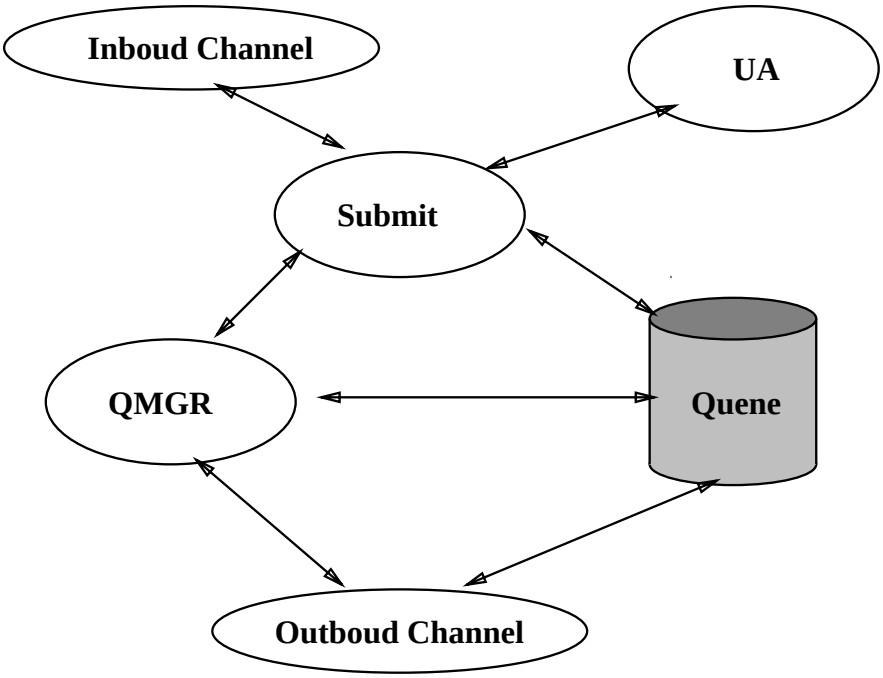

図 4.3: PP のプロセス構造

という 2 つの機能がある。

Submit は、受けとったメッセージを簡単な宛先/整合性のチェックの後 *QMGR* に渡すプロセスであり、入力は *UA* もしくは *Inbound Channel* である。Sumbit は、現在の PP では RFC822 アドレスもしくは X.400 O/R アドレスを理解する。いずれのアドレスもテキストエンコーディングされたものを扱う[9]。Submit はアドレスのパースを行ないメッセージが送られるべき MTA を決定する。その後に、適当な Outbound Channel が選択される。この際必要に応じてメッセージの変換が行なわれる。

メッセージは図 4.4のように処理される。

Channel は機能別には、

[9]X.400 O/R アドレスは RFC987[158]/RFC1148[159] の domain-oriented エンコーディングを用いる。

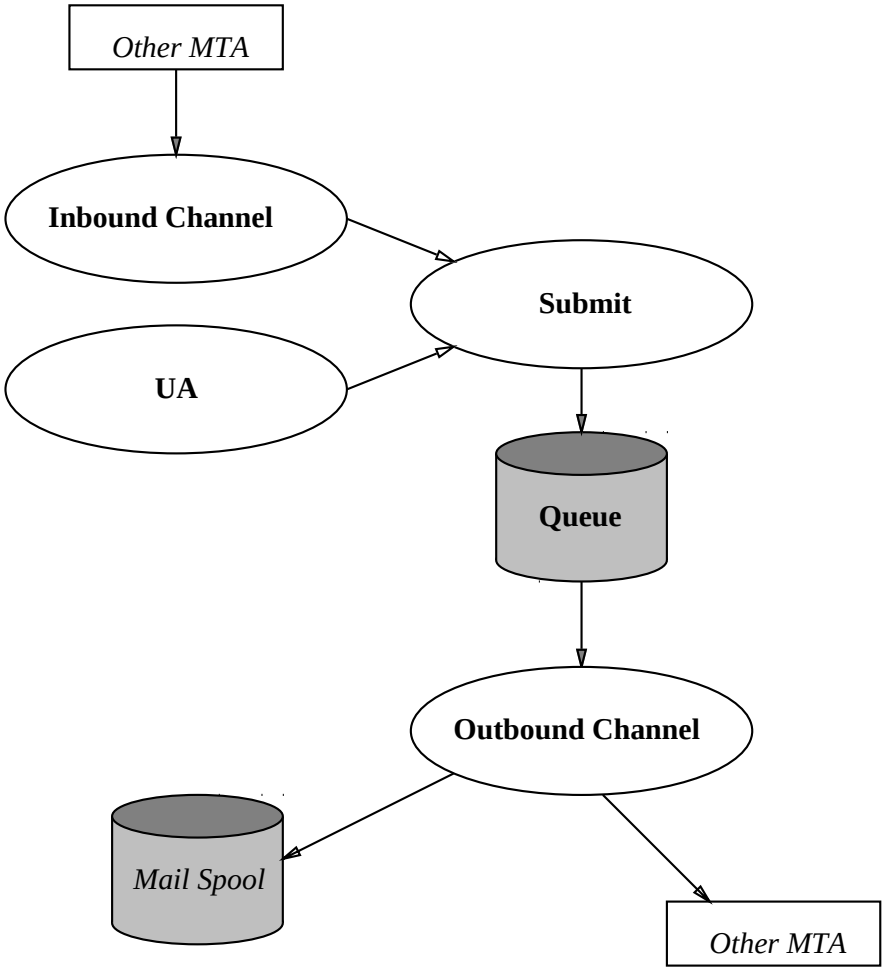

図 4.4: メッセージの流れ

Outbound Protocol Channel	Queue よりメッセージを取り出しメッセージ転送プロトコルを用いてメッセージの転送を行なう。
Other Outbound Channel	ローカルに対しての配信/Distribution List の展開など。
Inbound Channel	*Outbound Channel* の逆でサーバ的なことを行なう。
Message Reformatting and Protocol Conversion Channel	メッセージの変換、プロトコルの変換を行なう。
Housekeeping Channel	Queue 管理のための Channel

がある。

また、Channel には *Mandatory Channel* (PP が動作するために必要な Channel 群) と、

Optional Channel (実際にメッセージの配送を行なったり、メッセージのコンバートを行なったりする) の 2 つがある。

Mandatory Channel は、

qmgrload	PP のモデルである Queue 構造のコンシステンシィを保つ。
msg-clean	メッセージの配送が終った時点でメッセージを Queue から取り除く。
trash	ゴミ処理。
timeout	メッセージの Timeout 処理を行なう。
dr2rfc	Delivery Report の作成を行なう。
warning	Delivery Report 作成において Delay 等のための Report を作成。
splitter	Submit 時に複数の受信人がいる場合でかつ、単一の受信人ベースで処理を行なう場合に使う。

の 7 つがある。Optional Channel には、

822-local	**RFC822** フォーマットのメッセージをローカルのユーザに配送する。
slocal	構造のあるメッセージをローカルのユーザに配送する。
X400-84	**X.400 P1(84)** のメッセージを配送する。
X400-88	**X.400 P1(88)** のメッセージを配送する (*Beta Test* 中)。
SMTP	SMTP からの配送を扱う。
JNT Mail	X.25 を使った JNT Mail を扱う。
UUCP	UUCP Protocol を使ったメッセージを扱う。
DEC MAIL11	DECnet のメッセージを扱う。
p2explode	P2 のメッセージを複数の Body Part に分解する。
p2flatten	複数の Body Part を P2 メッセージにする。
P2toRFC	X.400 の P2(84/88) のヘッダを RFC822 に変換する。
RFC2P2	RFC822 のヘッダを X.400 の P2(84/88) ヘッダに変換する。
rfc934	RFC934 に規定されているメッセージのカプセル化に従った Forward Message を RFC822 形式に変換する。詳細は RFC934 を参照のこと。
list	*Distribution List* を展開する。
dirlist	X.500 を使って *Distribution List* を展開する。
shell	指定されたプロセスをコールする。
splitter	宛先一つ一つにメッセージを分解する。
fcontrol	Filter を制御する。

の 18 個がある。

4.3.2 PP の管理・運用

PP は、Queue/Channel の管理のために特殊なプログラムが用意されている。X Window System ベースの運用ツールは MTAconsole と呼ばれ図 4.5の様な概観をしている。通常は、このツールを用いて Queue 内にあるメッセージの数/動いている Channel の状態を Monitor/Control 出来る。

MTAconsole は、カラー化されており、必要に応じて表示色を変えるようになっている[10]。

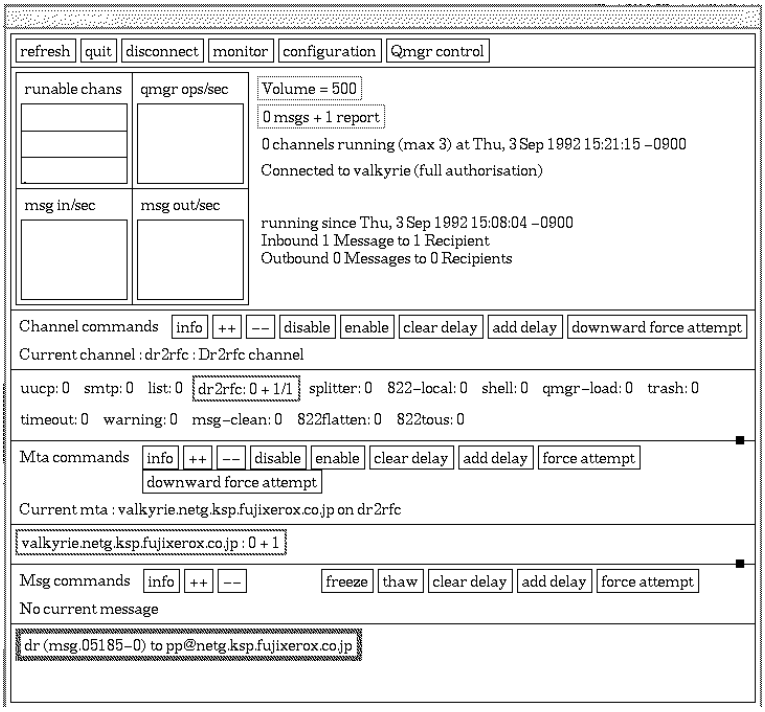

図 4.5: MTAconsole その 1

また MTAconsole は、QMGR に接続する際 Unix の Password と同じ形式の Authentication を行なう[11]。ユーザの Authentication を行なうことによって、権限のレベルを設定できる。PP では、none/limited/full の 3 つの権限の設定が出来るようになっている。各々の権限は、

none	Association(connection) を拒否する。
limited	Queue を examine できる。paramter の変更は許さない。
full	Queue を examine できる。paramter の変更ができる。

[10] 例えば、特定の Channel に多くのメッセージが未処理のまま溜っていた場合、該当する Channel の部分が赤く表示される。

[11] パスワードのは encrypt した形式で記入する。passwd の encryption のために mkpasswd というコマンドが用意されている

となっている。
この設定を行なうには、図 4.6のようなファイルを PP の table として作成する。

```
############################################################
#
# Auth.qmgr -- user ryu can do anythng about qmgr.
#
############################################################

anon:rights=limited
ryu:passwd=yB/HEx35rtKno,rights=full
```

図 4.6: auth.qmgr ファイルの例

このファイルの例では、ユーザ名の指定なしの場合は権限を制限し (*limited*)、ユーザ名が ryu でかつパスワードが正しい場合は権限の制限を行なわない (*full*) にしている。
Authentication を行なう事により、接続されている QMGR のコントロールに関する権限が設定され、不正な QMGR のコントロールを防いでいる。
Authentication を行なうには QMGR に Connect する際に Pop up する Logon Sheet (図 4.7) で Autentication を enable にすると User name/User passwd が出てくる (図 4.8) ので、User name/User passwd を入れると指定したユーザで QMGR と接続する。

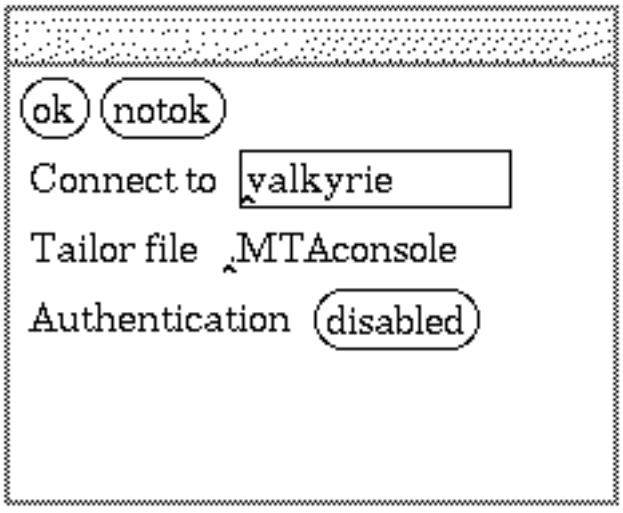

図 4.7: Anonymous での接続

Authentication Enable/Control モードでは、Qmgr control というボタンを押すことにより以下の操作ができる。

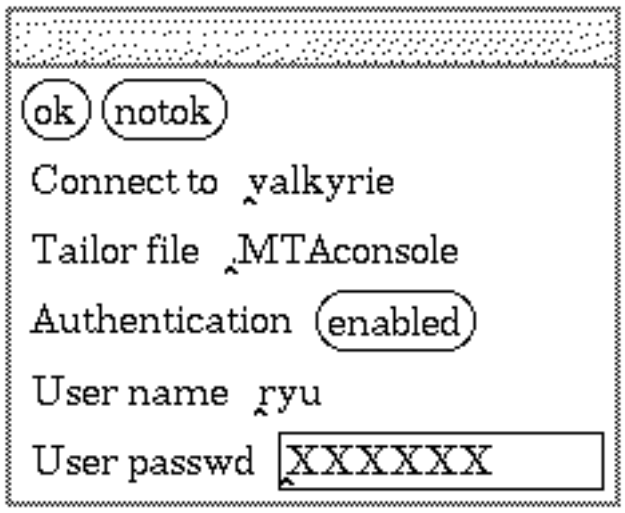

図 4.8: 特定ユーザでの接続

++maximum channel	一度に動ける Channel の数を増やす。
−−maximum channel	一度に動ける Channel の数を減らす。
enable submission	Submit OK にする。
disable sumbission	Submit を禁止する。
enable all/disable all	すべての動作を On/Off する。
rereadQ	Queue を読み直す。
restart Qmgr	qmgr を停止し、再起動。
gracefull stop	現在行なっている処理を終了後停止。
emegency stop	即時停止

この例では、配信報告が一つ Queue 内に入っていることがわかる。この配信報告の状態を詳しく見るには、Msg Command の info を押すことにより図 4.9の様にメッセージの細かな状態を見ることが出来る。また、注目しているメッセージに対して Delay の増加/減少、再度メッセージを処理させるなどの処理を指定し行なうことが可能である。

4.3.3 PP の問題点

PP は、非常に自由度が高く設計されており種々の設定を tailor ファイル [160] で設定できる。が、これが逆に仇となり設定が非常に難しい。また、X.500 の利用は考えられているが実際には *Distribution List* の展開に使っているのみである。より高度な X.500 の利用が望まれる。

PP の能力をフルに引き出す UA がないのもまた問題である。現状では、

1. PP のディストリビューションに含まれている UCB Mail とほぼコンパチな mail を使う

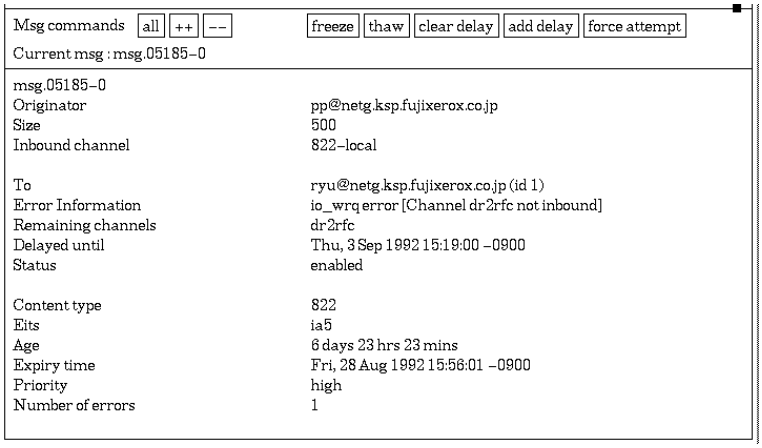

図 4.9: MTAconsole その 2

2. sendmail の Super Subset である fake sendmail を使う[12]

3. SMTP を使ったポストを行なう

4. 他の X.400 アプリケーションによってポストを行なう

のいづれかになる。

1. の方法では、「一応使える」というレベルでしかない。Install をはじめた時のテスト / 検証には使えるが実用に耐えるレベルではなく、また Multipul Body Part 等を扱う術もない。

2. の方法でも基本的には 1 と同じとなる。しかしながら、mh 等のプログラムを UA として使うことが出来る。Multipul Body Part 等を扱うには、MIME 等の変更などが必要であろう。

3. は、現時点でもっともいい解といえる。しかしながら 2 同様 Multipul Body Part 等の扱いには工夫が必要である。

4. に関しては、評価できる環境がなかったので評価はしていない。

4.3.4　現在の実験環境

富士ゼロックス内の WIDE の実験環境では、現在 PP が 4 台のマシンで動いており相互に X.400(84)/(88)/SMTP で通信しあっている。うち 2 台は SMTP/X.400 のゲートウェイマシンとして動作を行なっている。

他にも日立ソフトウェアエンジニアリングでは PP 6.5 が動作しており、富士通研でも PP 6.0 が動作している。日立ソフトウェアエンジニアリングでは、PP を用い UCL との間にプライベートリンク X.400 リンクをはり、幾つかのメールを X.400 で交換している。

[12]やはりディストリビューションに含まれている

現在行なっている実験は、

1. X.400/SMTP ゲートウェイ
2. *Table* による *Distribution List* の実験

である。これ以後、

1. X.500 Directory による *Distribution List* の実験
2. Multipul Bodypart の実験
3. **OSI Security Package** を利用して *Secured Body Part* の実験

の実験を予定している。

最終的には、富士ゼロックスの WIDE メイルゲートウェイを PP にリプレースする予定である。

4.4 研究題目

今後の研究題目としては、

1. WIDE ISODE WG に参加している組織間の接続
2. X.500 Directory との連係
3. Pilot Project 国際実験への参加
4. JP1(日本語化) に対する対応
5. MIME との間の相互変換機能の調査

を予定している。

1. のためには、WIDE における **ADMD/PRMD**[157] を決める必要がある。またこれに付随し各組織間でのプロファイルの設定などのコンヴェンションを決める必要がある。

2. のためには、国内 DSA 網に対して幾つかのデータを入れ、また各組織の DIT に対して一意な O/R 名をつける方法を決めなければならない。また、X.400(88) より追加された Distribution List(DL) の実験のため、各組織で動作している quipu の変更を行なう必要がある。また、DL 名の付け方に対して幾つかの合意を定める必要があるかも知れない。

3. は 2 のステップの後 Pilot Project が定める方法で参加実験の申し込みを行なうことになる。現在の X.400 PP のバージョンは 6.0 であるが、Pilot Project 等では新しいバージョンの PP(6.8βおよび ISODE Consortium Ver. 1.0) 等が既に動作している模様であり、これらのバージョンを用いた場合、より円滑な X.500 との連係が行なえる模様である。

4. は、おそらく Pilot Project の Comunity から WIDE に対して強く望まれるものであろう。PP におけるキャラクタコードの実装方法 [161] および INTAP/TTC 等日本の規格団体における日本語の扱いに対する規定 [162] の調査が必要であろう。また、新しいバージョンの PP では既に MIME などとの間の相互変換機能が実装されている模様でありそれに対して日本語に関する扱いを見い出したい。

5. は、4 とも関連するが Internet 上で円滑に構造を持った電子メイルを扱う手段を MIME は提供している。MIME は、機能の一部として X.400 に見られる複数のボディパートの扱いをサポートしており、X.400 と MIME との間の相互変換については非常に有用性が高い。この点を考慮し、4 の JP1 化を含め研究を進めていくつもりである。

第 5 章

OSI セキュリティの調査と基礎実験

5.1 はじめに

現在インターネットでは、一般的にパスワード認証が用いられている。しかし、パスワードによる認証は、ネットワークの盗聴、Crack などのツールによるパスワード解析などにより、パスワードを盗まれて簡単になりすましされる危険性があり、セキュリティ上十分な認証を提供しているとはいえない。

CCITT X.509 “The Directory - Authentication Framework”[163] では、ディレクトリモデルを用いて、分散環境における認証方式を勧告している。X.509 に勧告されている認証方式は、Simple Authentication と呼ばれるパスワードをベースとした認証方式と、Strong Authentication と呼ばれる公開鍵暗号法をベースとした認証方式がある。 Strong Authentication は上記のセキュリティ上の脅威を解決するものである。

OSISEC(The OSI Security Package)[164] は、OSI 上位層を提供するソフトウェアパッケージである ISODE を用いて、 X.509 の Strong Authentication を実装している。

ここでは、X.509 の概要と、OSISEC の概要を紹介し、OSISEC の動作実験の結果と今後の研究課題を報告する。

5.2 X.509 の概要

X.509 の Strong Authentication について簡単に説明する。この認証方式には以下のキーワードがある。

- Certification Authority
- Certificate
- Cross Certificate
- Certification Path

以下、それぞれについて説明する。

5.2.1 Certification Authority

Certification Authority(CA) は、ユーザの暗号鍵、ユーザを証明する Certificate（証明書）などを発行する、X.509 の中のオーソリティ的な存在である。当然 CA は信頼のおけるものでなくてはならない。

CA は識別名で認識され、図 5.1のように階層構造をとることができる。さらに、DIT[1]と一致させることもできる。

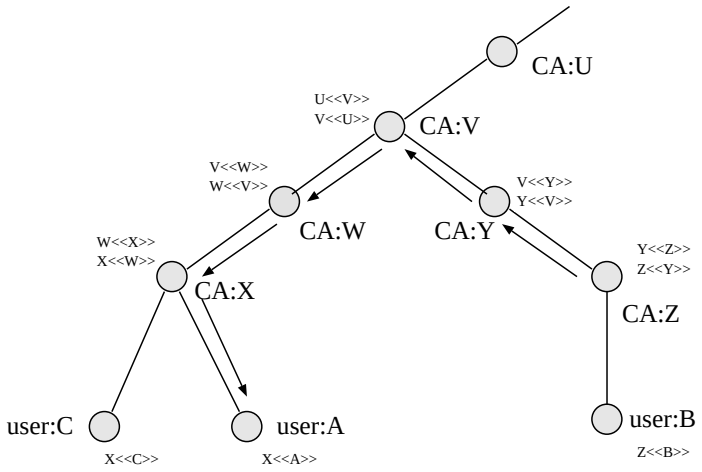

図 5.1: CA-hierarchy

CA が階層構造をとり、後述する Cross Certificate や Certification Path を駆使することによって、分散された計算機環境でも、世界のすみずみまでユーザを認証することが可能である。X.509 は広域分散環境を十分考慮していることがわかる。

5.2.2 Certificate

Certificate は、ユーザの身元を証明するものである。Certificate は、ユーザの公開鍵など図 5.2のような情報を CA の秘密鍵で暗号化して発行されるため、改竄されないようになっている。

Certificate には、CA 自身の CACertificate、ユーザのための UserCertificate、基本的に CA 同士が発行し合う CrossCerfificatePair の３種類がある。

また、CrossCerfificatePair には forward Certificate と reverse Certificate があり、図 5.1 の CA:X と CA:W に着目すれば、二者は図 5.3のような関係になっている。

CrossCertificatePair は、国を代表する CA と組織を代表する CA で発行しあったり、各組織同士で発行し合うこともできる。

[1] Directory Information Tree

```
Certificate ::=       SIGNED SEQUENCE{
       version                [0]Version DEFAULT 1988.
       serialNumber           SerialNumber,
       signature              Algorithmidentifier,
       issuer                 Name,
       validity               Validity,
       subject                Name,
       subjectPublicKeyInfo SubjectPublicKeyInfo}
```

図 5.2: Certificate

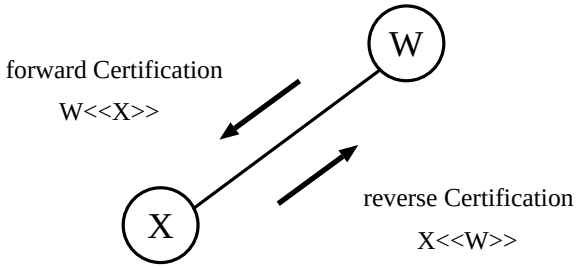

図 5.3: Cross Certificate

5.2.3 Certification Path

Certification Path(CP) は、相手の公開鍵を取得するために使われる Certificate のリストである[2]。図 5.1で、userB から userA までの CP は、

$B \rightarrow A = Z << Y >>, Y << V >>, V << W >>, W << X >>, X << A >>$

と表せる。

また、CP は DIT の階層構造に捕らわれることなく、図 5.4のように非階層的なパスを作ることもできる。

Certification Path の使い道については、5.2.4で述べる。

5.2.4 例：一方向性認証

具体的なユーザ認証として X.509 の中にある一方向性認証 (One way Authentication) を例として示す。X.509 には他に二方向認証、三方向認証の例が記述されている。

例えば図 5.1で、user:B が user:A を認証する時は図 5.5のようになる。

図 5.1の手順は以下のようである。

[2]公開鍵の取得方法については 5.2.4参照

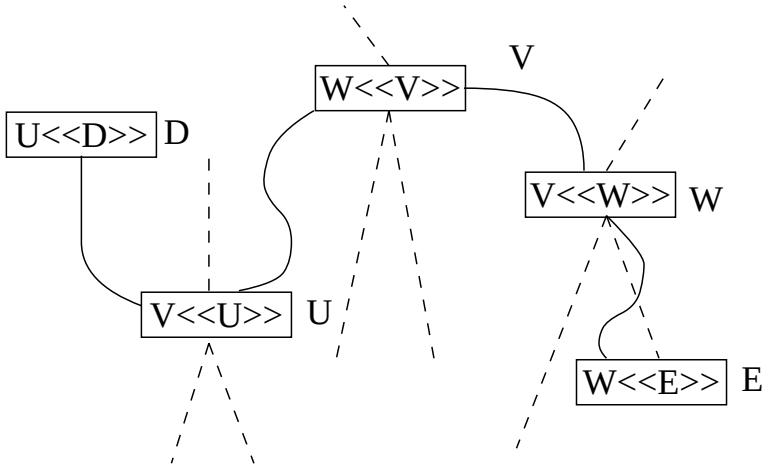

図 5.4: Non-hierarchical Certification Path

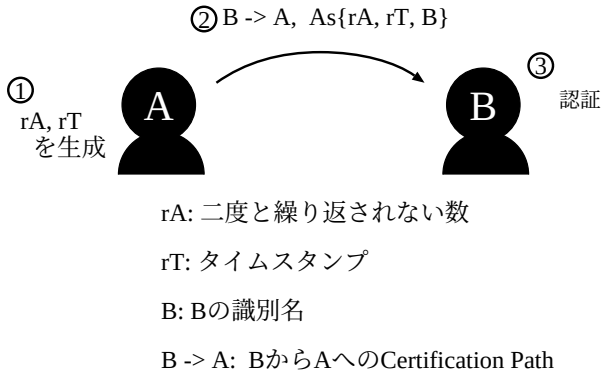

図 5.5: 例：一方向認証

1. user:A は rA, rT を生成する。
2. user:A は、rA, rT, A を userA の秘密鍵で暗号化し、B→A と共に user:B に送る。
3. userB は、CA:Z の公開鍵で $Z << Y >>$を復号化し、CA:Y の公開鍵を取り出す。その要領で順に CA:V, CA:W, CA:X の公開鍵を取り出し、最後に user:A の公開鍵を取り出す。。その公開鍵で As{rA, rT, B} を復号化し、rA, rT, A を確認することで userA を認証する。

5.3 OSISEC

OSISEC は、OSI 上位層を実装したソフトウェアパッケージである ISODE を用いて X.509 を実装している。DAP(Directory Access Protocol) などのディレクトリの実装は、

QUIPU に依存している。

5.3.1 実装状況

具体的に実装している X.509 の機能としては、以下のものがある。

- 暗号鍵の発行
- Certificate の発行
- Cross Certificate の発行
- Certification Path の確立
- Certificate の廃棄

また、OSISEC は上記の X.509 の機能を実現するために、表 5.1 のセキュリティアーキテクチャを実装している。

表 5.1: Generic OSI Security Architecture

Module	Service/Mechanisms/Facilities
Security Service	Data Confidentiality Data Integrity Non-Repudiation of Origgin
Security Mechanisms	Encipherment Integrity Checks Digital Sigunature Verification
Security Facilities	Public Key Cryptography Hash Function

表 5.1で、OSISEC は公開鍵暗号法として RSA を実装し、ディジタル署名を用いることによって、データ整合性のチェックをなど実現している。また、暗号の効率化のためにハッシュ関数を提供しており、MD2[165], MD4[166], MD5[167] が実装されている。これらはすべて C 言語のソースコードで提供されており、自由に参照することができる。

5.3.2 アプリケーションへの応用

1. Directory Service
   OSISEC を QUIPU に応用し、以下のような Strong Authentication を可能にしている。

- サービス、エントリ、属性、そして個々のオブジェクトに対する任意のアクセスコントロール
- Certificate にベースを置いた、DIT のなかの組織同士のアクセスコントロール
- 検索やリストなどのアクセスコントロール
- DSA の referal や chain のアクセスコントロール
- ユーザアカウント

2. MHS (DOCSEC)
   OSISEC を PP(MHS) に応用している。そのパッケージは DOCSEC[168] という。具体的な機能は、メッセージのボディパート部を暗号化、復号化するフィルターを提供し、MHS の暗号化メイルを実現している。また、以下のような応用も検討しているようである。

   - セキュアなテレマティクドキュメントの MHS による交換
   - ODA でないデータの MHS による交換

5.4　動作実験

OSISEC を用いた証明書発行および確認の動作実験を行なった。その動作の流れを図 5.6に示す。

以下、発行局を CA、その識別名を DN_{ca}、ユーザを U、その識別名を DN_u、CA の公開鍵を KP_{ca}、秘密鍵を KS_{ca}、ユーザの公開鍵を KP_u、秘密鍵を KS_u、発行された証明書を $CERT_u$と表す。U は CA である場合もある。

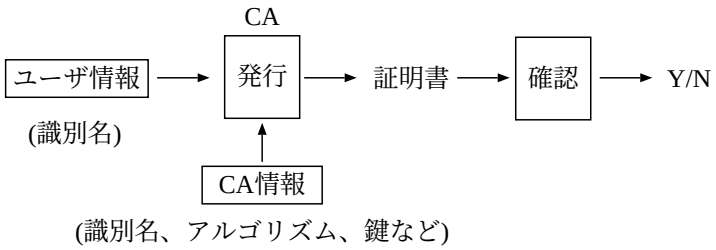

図 5.6: 動作の流れ

5.4.1 CA の情報

CA を運用する上で必要な情報は、tailor file で指定されている。ここで指定される情報には、CA の管理者名、管理者のパスワード、 DN_{ca}、KP_{ca}、KS_{ca}、 KS_u、 KP_uがある。

5.4.2 証明書発行

CA が証明書を発行する処理は、以下のステップからなっている。

1. 入力···証明書発行で必要な情報を入力する。

2. 符号化···1 の入力を基本符合化規則にしたがって符号化する。

3. 圧縮···2 の出力をハッシュ関数によって圧縮する。

4. 署名···3 の出力を CA の秘密鍵 KS_{ca}によって暗号化する。

5. 発行···生成（計算）された証明書を、各ユーザへ返送する。

入力

CA が証明書を発行する時、CA が生成する情報には次のものがある。

- シリアル番号 S_{no}
- 発行日 V_f、有効期限 V_e
- KS_u、KP_u

また、その他 CA が指定するものとして、使用される暗号アルゴリズム Alg_{pk}、そのパラメータ Alg_{pkpara}、圧縮アルゴリズム Alg_{md}、ASN.1 アルゴリズム Alg_{asn}、アルゴリズムのバージョン Alg_{ver}がある。
圧縮アルゴリズムは MD2, MD4, MD5 から選ぶことが出来る。
証明書を発行する時必要な証明書発行入力情報 Xは、次の情報から構成される。

$$X = (DN_{ca}, DN_u, Alg_{pk}, Alg_{pkpara}, Alg_{asn}, Alg_{ver}, S_{no}, V_f, V_e, Alg_{pkpara}, KP_u)$$

実際に、実験で用いたデータを図 5.7に示す。

```
DN_ca: c=JP@o=Fujitsu Laboratories Ltd.@ou=Fujitsu Certification Authority
DN_u: c=JP@o=Fujitsu Laboratories Ltd.@ou=Fujitsu Certification
          Authority@cn=Yasutsugu Kuroda12
Alg_md: md2WithRsa
Alg_asn: ASN{050000}
Alg_ver: 0
S_no: 730956002
V_f: 930301032002Z
V_e: 940301032002Z
A_pk: rsa
Alg_pkpara: 512
KP_u:3047024057b36eb144716f0d9b44bcfc3d599417a6b1fa302f47f6e8aa9c3977bbd4
     b502f6bee9f4aac3687b8dacd876d77950f11ebb8ad9fb9e58c3abfc0372180555cf
     0203010001
```

図 5.7: 証明書発行入力データ

符号化

入力された証明書入力情報は、ISODE で提供されている ASN.1 コンパイラを使って、基本符号化規則 (BER) によって符号化される[3]。BER の関数を BER(), その出力結果を Y とすれば、

$$Y = BER(X)$$

と表される。

例えば、OSISEC の ASN.1 コンパイラは、図 5.7の証明書入力情報 Y を 359 オクテットの大きさに符号化する。

圧縮

情報の改竄を防止するため、5.4.2で符号化されたデータ Yは、ハッシュ関数により 32 オクテットに圧縮される。 OSISEC では、MD2, MD4, MD5 の３種のハッシュ関数が提供されている。ハッシュ関数を H(), その出力を Z とすれば、$Z = H(X)$ と表される。

例えば、5.4.2の Yを符号化したものを、MD2 を用いて圧縮すると、以下のようになる。

$$Z = H(Y) = \text{aae9ce6ebf6c56348feac95cb5bad4e2}$$

[3]実際には、DER 符号化される

署名

OSISEC での署名は、5.4.2で圧縮されたデータを、KS_{ca}で RSA 暗号化 [169] したものである。鍵 KS_{ca}で Z を RSA 暗号化することを $RSA(Z)_{KS_{ca}}$と表す。

例えば、5.4.2の Z を RSA 暗号化すると、以下のような 64 オクテットになる。

$$RSA(Z)_{KS_{ca}} = \begin{array}{l}\text{9f2ee67cd6d2cf1c2d3c58399ee97ee44f85f64ce945023634bde5d245e3b1db} \\ \text{b498b024a2f0be1cca6b1f57f40b7cd9742668110cdedb0b1aed064c2cb05f13}\end{array}$$

発行

OSISEC の証明書は、入力情報に RSA の署名を付加したものである。図 5.8に実際に出力される例を示す。

```
md2WithRsa#{ASN}050000#9f2ee67cd6d2cf1c2d3c58399ee97ee44f85f64ce945023
634bde5d245e3b1dbb498b024a2f0be1cca6b1f57f40b7cd9742668110cdedb0b1aed0
64c2cb05f13#c=JP@o=Fujitsu Laboratories Ltd.@ou=Fujitsu Certification
Authority#c=JP@o=Fujitsu Laboratories Ltd.@ou=Fujitsu Certification Au
thority@cn=Yasutsugu Kuroda12#md2WithRsa#{ASN}050000#0#730956002#93030
1032002Z#940301032002Z#rsa#512#3047024057b36eb144716f0d9b44bcfc3d59941
7a6b1fa302f47f6e8aa9c3977bbd4b502f6bee9f4aac3687b8dacd876d77950f11ebb8
ad9fb9e58c3abfc0372180555cf0203010001#
```

図 5.8: 証明書の例

図 5.8で、#はデリミタで、3 番目の “9f2ee67cd6d....” フィールドが RSA 署名、4 番目の “c=JP@o=Fuji....” フィールドが DN_{ca}、4 番目の “c=JP@o=Fuji...” が DN_u、最後の “3047024057b....” が KP_uである。

5.4.3 証明書確認

証明書を確認する処理は、以下のステップからなっている。

1. 入力･･･証明書確認で必要な情報を入力する。
2. 符号化･･･1 の入力を基本符号化規則にしたがって符号化する。
3. 圧縮･･･2 の出力をハッシュ関数によって圧縮する。
4. 復号化･･･署名の RSA 復号化

5. 4 と 3 の値を比較して、証明書が有効であるか、無効であるかを判定する。

1-3 は 5.4.2の 1-3 と同様である。
3 の出力を Z'する。また、鍵 KP_{ca}で 5.4.2の署名 $RSA(Z)_{KS_{ca}}$を RSA 復号化することを $RSA(RSA(Z)_{KS_{ca}})_{KP_{ca}}$と表す。
証明書の確認は

$$Z' = RSA(RSA(Z)_{KS_{ca}})_{KP_{ca}}$$

で行なわれ、左辺 = 右辺 であれば有効、左辺 $\neq$ 右辺 であれば無効となる。

5.5　評価

5.5.1　証明書の改竄

ここで、実験のために作成した、証明書を確認するコマンドを用いて、図 5.8の証明書の一文字を故意に改竄し、その出力結果を比較する。
図 5.8の証明書の例の、DN_{ca}フィールドの “c =JP@o=Fujitsu...” を “c=JP@o=fujitsu...” と改竄して、計算した例を表 5.2に示す。

表 5.2: 改竄の例

DN_{ca}	ハッシュ関数の出力
c =JP@o=Fujitsu...	750c713f1c51e3c3c6816dc2648a77f2
c =JP@o=fujitsu...	aae9ce6ebf6c56348feac95cb5bad4e2

表 5.2のように、証明書のなかの 1 文字を変えただけでも、出力される結果が劇的に変化することがわかる。従って、その後の RSA の計算も当然異なる値となり、改竄が生じたことが判明する。[4]
情報に改竄が行なわれていないことを保証する性質を完全性という。OSISEC は RSA によるディジタル署名により、完全性を保証している。

5.5.2　処理時間

RSA 鍵を生成する時間を表 5.3に。任意のファイル長に対する処理時間と処理速度を表 5.4に。証明書の発行にかかる時間を表 5.5に、証明書の確認にかかる時間を表 5.6に示す。
ただし、測定は組み込みの time コマンドを用いた。また、RSA 鍵を生成する時間、証明書の発行にかかる時間、明書の確認にかかる時間は、各モジュールを 100 回繰り返して平均を取ったものである。

[4]数学的な評価は行なっていない。

表 5.3: 鍵の生成時間

	処理時間 [sec]
p	23.87
q	19.32
他	0.15
合計	43.37

表 5.4: RSA の処理時間

	処理時間 [sec]			処理速度 [Kbps]
ファイル長 [Kbyte]	9612	106496	2598812	
RSA 暗号化	17.8	177.8	4826.7	4.524
RSA 復号化	197.6	1972.1	47890.6	0.2519

5.6 今後の研究題目

現在のインターネットにおけるセキュリティ上の脅威を考える時、パスワード認証に代わる認証方式の研究が急務である。その 1 つの候補が公開鍵暗号をベースとした X.509 である。

WIDE ISODE WG としての、X.509 の実装である OSISEC を使った今後の研究には、「Certification Authority とその運用方式に関する研究」、「X.509 の認証方式の他アプリケーションへの応用に関する研究」の大きな 2 つの柱がある。

1. Certification Authority とその運用方式に関する研究
   現在インターネットでは、暗号化メイル PEM[170] の研究が進んでいる。WIDE インターネット内で暗号化メイルを実用化しようとした時、WIDE を代表する CA の立ち上げが必要不可欠である。

   その準備として、ISODE WG 内で CA を立ち上げ、実験を行なう。また、CA の運用で問題となる、暗号鍵の配送、保存場所などの研究も行なう。

2. X.509 の認証方式のアプリケーションへの応用に関する研究

表 5.5: 証明書発行の処理時間

	処理時間 [sec]
ASN.1 符合化	0.005
MD2	0.030
署名時間	1.070
合計	1.105

表 5.6: 証明書確認の処理時間

	処理時間 [sec]
ASN.1 符合化	0.005
MD2	0.030
署名確認時間	0.110
合計	0.145

現在 ISODE Consortium から、X.509 を MHS に応用したソフトウェア Docsec が提供されている。PP(MHS) の接続実験と並行して Docsec の実験も行なう。また、MHS に限らず他のアプリケーションへの応用も行なう。

第 6 章

今後の活動

1992 年度の ISODE WG は、ディレクトリサービスによる様々な情報管理の試みと、MHS および OSI セキュリティの研究のための調査と基礎実験を行ったが、次年度はさらなるディレクトリサービスの応用の研究と、MHS および OSI セキュリティの本格的な研究、その他の上位層、下位層についての調査・研究を行なう。

6.1 ディレクトリサービス

次年度のディレクトリサービスの研究は、以下のような項目について研究することを予定している。

- ディレクトリサービスでの動的情報の取扱い
- 情報の検索、取得方法の最適化
- 既存のディレクトリサービスの共存と置換
- 新規システム情報の配送機構
- セキュリティ機構

6.2 MHS

今後の研究課題としては、実験用 MHS 網の整備と、OSI 環境下での運用および、TCP/IP 環境との共存などの見地から、以下の項目を予定している。

- WIDE ISODE WG に参加している組織間の接続
- X.500 Directory との連係
- Pilot Project 国際実験への参加
- JP1(日本語化) に対する対応
- SMTP/MIME との間の相互変換機能の調査

6.3　OSI セキュリティ

次年度は、基礎実験を元に、Certification Authority とその運用方式に関する研究と、X.509 の認証方式のアプリケーションへの応用に関する研究を行なう予定である。

6.4　その他の OSI 上位層

その他の OSI 上位層として、ディレクトリサービス、MHS に続いて、重要なアプリケーションであるファイル転送・管理 (FTAM) についての調査と基礎実験を行う予定である。

また、実際のネットワークの構築の際に問題となると思われる相互接続性に関して、各種のアプリケーションにおける共通の問題点、解決方法について考察をする予定である。

6.5　OSI 下位層

UNIX 上での OSI 下位層の使用や、TCP/IP との共存に関しての問題点、解決方法についての研究を行う予定である。

6.6　ゲートウェイ機能

次年度以降、TCP/IP と OSI の共存および、TCP/IP から OSI への移行をしていく上で必須と思われる各種のゲートウェイ機能の実現と運用について研究・実験していく予定である。