第 17 部

OSI アプリケーション

第 1 章

1993 年度の活動概要

1.1 ISODE WG と ISODE パッケージ

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プロダクトの開発にも使用されている。また、TCP/IP スイーツから OSI スイーツへの移行手段の例と捉えることができる。OSI の規格は定められて来たが、その実装例はあまり多くない。特に、公開されたものは希少であるため、OSI プロトコルの実装の一例としての意味も大きい。

現在の ISODE パッケージは、ISODE Consortium によって製品化の作業が行われており、OSI ディレクトリサービスを提供する QUIPU、MHS (メッセージハンドリングシステム) を提供する PP、ネットワーク管理のための SNMP、ファイル転送と管理の FTAM が含まれている。最新のバージョンは IC 1.1 (ISODE 10.0) である。IC 版 ISODE パッケージの元となったフリーソフトウェア版の ISODE 8.0 には、これ以外に、仮想端末、ネットワーク管理モジュール、セキュリティ機能を提供する OSISEC パッケージも含まれているので、順次 IC 版 ISODE パッケージに反映されるであろう。

IC 版の ISODE パッケージは ISODE Consortium 参加組織にのみ配布されている。WIDE プロジェクトは、最新の OSI 技術を元に研究をするために、1993 年度から ISODE Consortium に参加し、IC 版の ISODE パッケージを使用しているが、ISODE Consortium に参加していない組織にその成果を還元するため、ISODE 8.0 も使用している。

1.2 研究・実験内容

1993 年度は、主に OSI ディレクトリサービスの研究を引き続いて行なった。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で多国語が同時に使用できるよう、国際化の仕様を検討した。
- firewall 環境での OSI ディレクトリサービスの利用のため、Relay DSA の使用方法の検討を行った。
- 階層的情報提供システムの最初の試みとして、NIS サーバから OSI ディレクトリサービスを使用する機構を研究し、実装した。
- ネットワーク管理システム (IPANeMa) の情報を OSI ディレクトリサービスで管理する機構を検討した。
- OSI ディレクトリサービスでより幅広い範囲のデータを扱う事ができるよう、動画像 (MPEG) ファイルやプレインテキストファイルの属性を扱う機構を実装した。
- ユーザに容易に OSI ディレクトリサービスの情報を提供できるよう、WWW から OSI ディレクトリサービスを使用する機構の実験を始めた。

第 2 章

OSI ディレクトリサービス (X.500) の国際化

2.1 はじめに

ISODE WG では OSI ディレクトリサービス (X.500)[112] の全国実験を行っている。現在の X.500 では、一部の試験的な場合 [214] を除くと日本語は使用できない。また、多国語を扱う枠組は用意されていない。ディレクトリ・サービスを全世界的に利用できる様にするためには、多国語の扱いなどの国際化が必要である。

そこで ISODE WG では X.500 の国際化、特に多国語化における問題の検討、多国語化の方法、QUIPU[215] に対する設計の検討を行った。

2.2 国際化の対象と目標

2.2.1 ディレクトリを構成する要素

OSI ディレクトリ・サービスは様々な情報の参照や検索機構の提供と、分散した管理を行うためのサービスである。それぞれの情報はエントリと呼ぶオブジェクトに含まれる。エントリは 1 つ以上の属性を持ち、属性は属性名と 1 つ以上の属性値から構成される。属性値は、属姓名によって異なり、データ型は属性構文によって定義される。エントリ、属性、属性値の関係を図 2.1に示す。

それぞれのエントリは相対識別名 (RDN、Relative Distinguised Name) を名前として持ち、相対識別名は属性型と属性値の組 (AVA、Attribute Value Assersion) の集合である。複数の AVA も使用可能である。

ルートから相対識別名を順番に並べたものを識別名 (DN、Distinguised Name) と言い、エントリの名前を表す。

エントリは、仮想的なルートから始まる木構造を構成する階層的な名前を持つ。エントリが構成する木構造をディレクトリ情報ツリー (DIT、Directory Information Tree) と呼ぶ。DIT の例を図 2.2に示す。

ディレクトリの情報は DSA(Directory System Agent) が保持、管理をしている。ユーザはディレクトリの情報を DUA(Directory User Agent) を通して DSA にアクセスする。DSA は、DUA や他の DSA といった要求者から問い合わせを受けたエントリを保持していれば、その情報を応答する。保持していない場合はエラーとして、保持している DSA の情報を参照先の情報 (referral) と共に要求者に返すか、保持している DSA に問い合わ

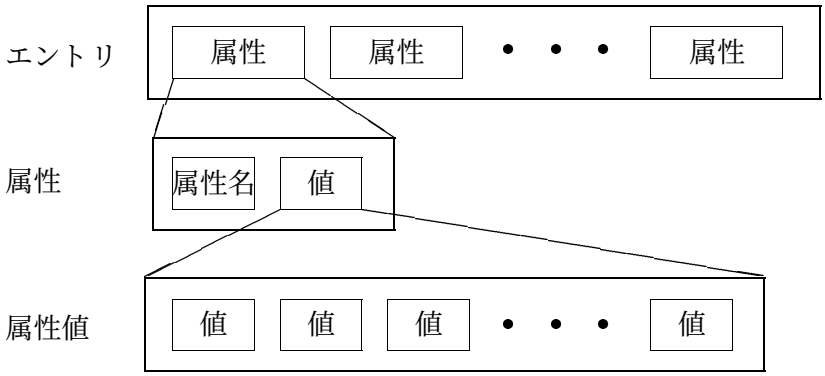

図 2.1: エントリと属性

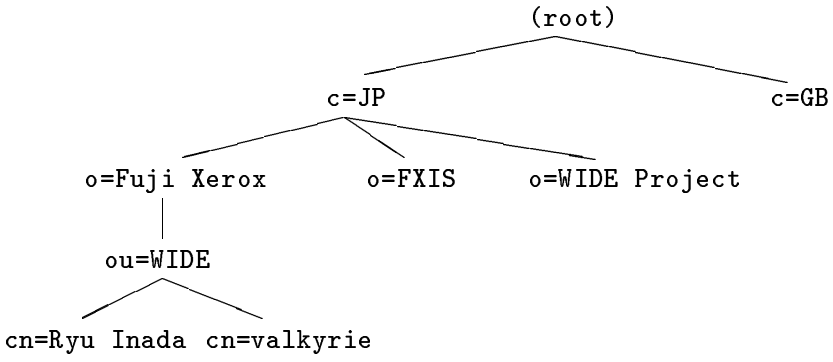

図 2.2: DIT の例

せの連鎖 (chaining) を行う。このように DSA は協調して DUA にディレクトリ・サービスを提供する。

2.2.2　国際化の目標

国際化の目標は次の通りである。

- DSA や DUA で多国語を扱うことを可能とする。すなわち、以下の項目について多国語を使用できるようにする。

  属性名 属性を識別するデータ型

  属性値 属性の値

  識別名 エントリを指定する名前

- 既存の Pilot Project の DSA との相互運用ができなくてはならない。

- ISO/IEC[1]や ITU-TS[2] の標準をなるべく尊重する。

この他に、属性の比較に関する地域化の方法などについては見送っている。

2.2.3　属性名

属性名はオブジェクト識別子として定義されていて、属性名はユーザが指定する文字列としては、プロトコル上は存在しない。ユーザが属性名を指定する文字列からオブジェクト識別子への対応は、DUA がローカルにユーザに提供するものである。このため、属性名の文字列は、locale やユーザの好みに従った変更を、DUA の設定として自由に行うことができる。

各国や言語に従った、属性名を表す文字列とオブジェクト識別子の対応の情報を、ディレクトリ上に格納して利用するのも一つの方法である。

2.2.4　属性値

属性値に複数の言語を格納できなければならない。多くの属性は複数の値を格納することができるため、従来の属性値と多国語を含んだ属性値の両方を格納できる。表 2.1に標準的な属性で、複数値を持てない属性を示す。

国際化されていない属性と、国際化された属性の両方を持たせることで、エントリの持つデータ量は増えてしまう。しかし、国際化のサポートの有無といった DUA の都合で、値を取捨選択した表示が可能となり、柔軟な取り扱いができるようになる。

---

[1] International Organization for Standards および International Electrotechnical Committee

[2] International Telecommunication Union, Telecommunication Standardization Section、旧 CCITT

表 2.1: 1 つの値だけを持つ属性

属性型	意味
AliasedObjectName	別名の指す識別名
countryName	ISO3166[216] の 2 文字の国コード
preferredDeliveryMethod	好ましい配送方法
presentationAddress	アプリケーションのアクセス・ポイント

2.2.5 識別名

識別名は属性名と属性値の組で構成される。属性名や属性値の多国語化を行えば、識別名も原理的には多国語の使用が可能となる。しかし、識別名はエントリの名前を表すため、単に多国語化を許すと次の様な問題を起こす可能性がある。

- 国際化をサポートしていない DUA では名前の指定が困難または不可能であるため、相互運用性や利便性の低下を招く。
- 識別名に多国語の使用を可能にすると、多くの場合は自国語を使用することが予想される。この結果、自国内でしか通用しない名前となってしまう。

この問題については、次のような対処方法が考えられる。

1. 識別名に多国語を使用可能とする。上述の問題点には運用による対処に任せる。
2. 識別名に多国語は使用しない。但し、別名のエントリは例外で多国語を使用可能とする。
3. 識別名に多国語は使用しない。

相互運用性を重視する点からは、当面は、最後の多国語を使用しない方針とする。また、識別名に多国語を使用しているかどうかを検査する機能は、実験の幅を広げる意味から組み込まないことにする。

2.2.6 識別名のテキスト表現形式

IETF の OSI-DS WG では、識別名のテキスト表現方法が 2 つ提案されている。これは、メッセージ中での識別名の参照や、人間が入力する時に利用することを目的としたものである。

一方は識別名と一対一に対応した厳密な名前を表す DN 形式 [128] で、他方は曖昧で識別名に直接対応しないが、よりユーザ指向な形式である UFN(User Friendly Name) 形式 [217] である。

DN 形式では ASCII 文字集合以外の使用を考慮していないので、識別名に多国語を使用しない方針と合致する。

UFN 形式で指定された名前は、最終的な DN 形式を得るための検索に使用する属性値である。このため、識別名に多国語が使用できなくても、一般的な属性値に多国語が利用できれば、多国語の UFN は使用可能になる。

2.3　多国語化の方法

多国語のサポートを実装するためには、以下の文字コードについての検討が必要である。

通信コード	DSA や DUA 間の通信で使用するコード
ファイル・コード	DSA や DUA で使用するファイルに使用するコード
表示コード	主に DUA でユーザへの表示に使用するコード
内部形式	DSA や DUA の内部処理に使用する形式

相互運用で問題となるのは通信コードである。

2.3.1　ディレクトリと文字コード

現在、広く使用されている OSI ディレクトリサービスは、1988 年版の標準を元に拡張を加えたもので、QUIPU もその一つである。残念ながら 1988 年版では、多国語の使用について十分な考慮が行われているとは言い難い部分がある。

ディレクトリでは、テキストの情報を表す属性値において、多くの場合CaseIgnoreString の属性構文が使用されている。この属性構文は図 2.3の様に定義されている。

```
caseIgnoreString ATTRIBUTE-SYNTAX ::=
     SYNTAX  CHOICE { T61String, PrintableString }
     ID      { atrirbuteSyntax 4 }
```

図 2.3: caseIgnoreString 属性構文

これは、PrintableString と TelexString から、一方の文字のデータ型を選んで使用するデータ型であることと、属性構文を表すオブジェクト識別子を定義している。ASN.1[218] で定義されている文字の型を表 2.2に示す。

この様に PrintableString は ASCII 文字集合のサブセットである。また、TelexString は ASCII 文字集合と似た文字集合である。これら 2 つの文字のデータ型はいずれも、ISO2022[219] の符号拡張を認めたものではない。ISO2022 の符号拡張を認める文字のデータ型としては、GraphicString と GeneralString が該当するが、これらは図 2.3にある様に使用できない。

なお、表 2.2にある ISO2375[220] の登録番号に対応する文字集合を表 2.3に示す。

一方、1992 年版では図 2.4の様に、DirectorString をデータ型として定義している。

表 2.2: ASN.1 で定義済の文字のデータ型

データ型	文字の種類
NumericString	スペース '0' から '9' までの数字
PrintableString	NumericString の文字 'A' から 'Z' までと 'a' から 'z' までのアルファベット シングル・クォート '、括弧 (と)、正符号+、カンマ , ハイフン -、ピリオド .、スラッシュ /、コロン :、等号 =、疑問符 ?
TelexString	スペース、デリート ISO2375 の登録番号 87、102、103、106、107 の文字集合
VideotexString	スペース、デリート ISO2375 の登録番号 1、72、73、102、108、128、129 の文字集合
IA5String	スペース、デリート ISO2375 の登録番号 1、2 の文字集合
GraphicString	スペース すべての図形文字集合
GeneralString	スペース、デリート すべての図形文字集合

表 2.3: ASN.1 で参照している ISO2375 の登録番号の文字集合

登録番号	文字集合種別	終端文字	内容
1	制御 0	4/0	ISO646 の C0
2	94 文字	4/0	ISO646 の IRV
72	94 文字	6/4	CCITT ビデオテックス第 3 追加モザイク
73	制御 1	4/1	CCITT ビデオテックス属性制御
87	不明	不明	不明
102	94 文字	7/5	T.61 Telex 基本図形文字
103	94 文字	7/5	T.61 Telex 補助図形文字
106	制御 0	4/5	T.61 Telex 基本制御機能
107	制御 1	4/8	T.61 Telex 補助制御機能
108	その他	4/1	NAPLPS ANSI X3.110-1983、CSA T500 1983
128	94 文字	7/12	T.101 データ構文 III 補助図形文字
129	96 文字	7/13	T.101 データ構文 III 補助モザイク文字

```
DirectoryString { maxSize } ::=
     CHOICE {
             T61String (SIZE (1 .. maxSize)),
             PrintableString (SIZE (1 .. maxSize)),
             UNIVERSAL STRING (SIZE (1 .. maxSize))
     }
```

図 2.4: DirectoryString の定義

UNIVERSAL STRING は ISO10646[221] の文字集合であり、多国語を使用可能にはなっている。しかし、次の理由により 1992 年版は、将来の目標とせざるを得ない。

- ISODE の開発環境としてサポートされていない。特に ASN.1 も変更されているため、利用可能な開発環境も存在しない。
- 今のところ 1992 年版をサポートした DSA が殆んど稼働していない。
- プロトコルが大幅に変更がされている。そのため既存の DSA との通信には 1988 年版のプロトコルを使用することになる。

また、ISO10646 の文字集合自体も、広く安定して使用されているとは言えないのが現状である。

2.3.2　暫定的な方法 – T61IX

1988 年版に対する修正には次の方法が考えられる。

1. caseIgnoreString 属性構文に GeneralString を追加する。
2. PrintableString を使用して、MIME と同様な符号化を行う。
3. TelexString に ISO2022 の符号化を使用する。

第 1 の方法は、ディレクトリのリモート・オペレーションを変更することになる。このため、既存の DSA ではプレゼンテーション層の符号化処理でエラーになり相互運用が不可能である。

第 2 の方法は、符号化という面では最も安全な方法であり、DUA だけの変更で可能ではある。しかし、標準規格で決められている属性の最大長には、表 2.4に示す様に比較的小さいものもある。

MIME の B エンコーディングでは符号化の結果が長い文字列になる傾向が大きいので、多国語を使用した場合に理不尽な長さの制限が加わることが予想される。また、基本的に 8 ビット透過可能なところを、7 ビットへの符号化を行うのは技術的に望ましい事ではない。

[3] ここで言うパートは住所を数行に分けて書くときの各行に相当し、最大で 6 つまで使用できる。

表 2.4: 主要な属性の最大長

属性名	長さ (文字数)
commonName	64
organizationName	64
organizationalUnitName	64
postalAddress の各パート[3]	30
postOfficeBox	40

そこで、第 3 の方法である、1988 年版に最小限の修正を行って ISO2022[219] の符号拡張の方法による多国語化を行うことを検討した。

ASN.1 では TelexString は厳密には符号拡張は認められていない。符号拡張を使用する文字のデータ型は、GraphicString や GeneralString である。

しかし、TelexString の文字集合を定義している本来の T.61[222] では、付録で ISO2022 に準拠した符号拡張を認めている。そこで、TelexString を T.61 の付録にある符号拡張を含めた文字集合を使用可能と解釈し、ISO2022 の符号拡張を行った文字を使用する。この方法を T61IX(T61 with Interim eXtension) と呼ぶことにする。

TelexString は ISO2022 の符号拡張の方式から見ると、次の様な指示と呼出を行ったものである。

1. 8 ビット環境
2. G0 にテレテックス基本図形文字集合を指示して GL に呼び出し
3. G2 にテレテックス補助図形文字集合を指示して GR に呼び出し
4. 他の文字集合の指示や呼び出しはなし

最後の文字集合の指示や呼び出しを使用可能に拡張したのが T61IX となるわけである。

ISO2022 の使用による利点には以下の様な項目がある。

- Mule[223] に見られる様に多国語化に実績がある。
- 既存の流用可能なツールも多い。

2.4 T61IX と相互運用

ここでは国際化の目標の一つである、従来の DSA や DUA との相互運用について検討を行う。

2.4.1　国際化サポートの判別

アクセスする相手方の DUA や DSA が国際化されているかどうかを区別して、国際化 DUA や DSA がアクセスの方法を変更することも考えられるが、次の問題点がある。

- 接続時のパラメータに区別するための情報を追加するには、サービスの定義の変更が必要である。

- DSA のエントリに含まれる情報、具体的には`supportedApplicationContext` を元に区別を行うことも考えられる。処理が複雑になる上、オペレーションを連鎖する場合の処理の扱いが難しい。

この様に、明示的に区別する情報の追加は難しいが、後者の方法はさらに検討する余地があるであろう。

2.4.2　混在時の問題

国際化された DSA や DUA と従来の DSA や DUA が混在した状況では、以下の様な状況が発生する。

1. 国際化 DUA が保持している、多国語の属性値を持つエントリへの問い合わせが、未国際化 DSA から連鎖される。この場合、未国際化 DSA を通して多国語の属性値を要求者に返さなければならない。

2. 国際化 DSA の情報を、未国際化 DSA がスレーブ (他の DSA の複製) として保持して、それを他の国際化 DUA や DSA がアクセスする。

この状況を図 2.5に示す。

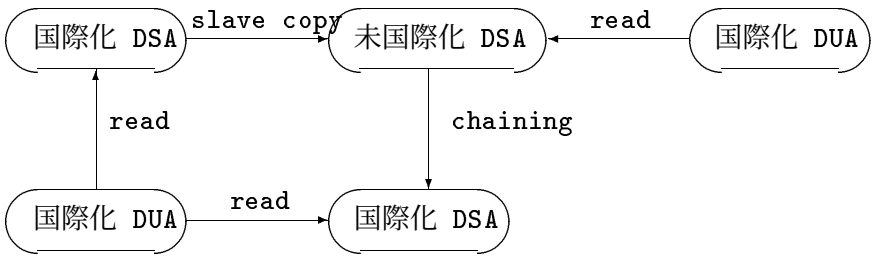

図 2.5: 国際化と未国際化の DUA や DSA の混在

以上の様な理由で、国際化されていない DSA に対して透過的な方法を用いることができれば、それが望ましいと言える。

2.4.3 TelexString と透過性

T61IX の方式を使用する場合には、本来の TelexString が一般にバイト単位で透過的に文字情報を受け渡すかどうかが問題となる。

ISO2022 を 8 ビット環境で使用する場合は、TelexString の GR に呼び出されているテレテックス補助図形文字集合の取り扱いに注意が必要である。ISO8859-1[224] と異なり、アクセントやウムラウトなどのアルファベットの装飾文字は、12/1 から 12/15 まで独立したコード・ポイントを割り当てられて、続く 1 文字を装飾の対象としている。単独の装飾文字は、スペースを続けることで表す。

表 2.5: テレテックス補助文字集合の修飾文字

コード	文字の名称
12/1	grave accent
12/2	acute accent
12/3	circumflex accent
12/4	tilde
12/5	macron
12/6	breve
12/7	dot
12/8	diaeresis または umlaut mark
12/10	ring
12/11	cedilla
12/13	double acute accent
12/14	ogonek
12/15	caron

次のような状況では、文字の値の検査が厳しい DSA ではエラーが発生して、DUA が結果を得られなかったり、スレーブのデータの受け取りに失敗し続けたりする危険性があり得る。

1. GR に文字集合を呼び出した。
2. 装飾文字と同じコード・ポイントの文字が出現する。
3. 続く文字が被修飾文字ではない。

ISO2022 を 7 ビット環境で使用する場合は、TelexString の GL に呼び出されているテレテックス基本図形文字集合には未使用のコード・ポイントが存在し、通常の ASCII に含まれる記号を全ては含んでいないことに注意が必要である。

含まれていない文字の一部はテレテックス補助図形文字集合、つまり GR に含まれているが、それでも全ての ASCII 文字集合をカバーしていない。

表 2.6: TelexString の未使用コード・ポイント

コード	文字の名称	備考
2/3	number sign	テレテックス補助文字集合の 10/6 に用意
2/4	dollar sign	テレテックス補助文字集合の 10/4 に用意
5/12	reverse solidus (backslash)	
5/14	circumflex accent	テレテックス補助文字集合の 12/3 に修飾文字
6/0	grave accent	テレテックス補助文字集合の 12/1 に修飾文字
7/11	left curly bracket	
7/13	right curly bracket	
7/14	tilde	

このため、装飾文字の場合と同様な問題が起きる可能性が考えられる。

しかし、この問題については、現在の ISODE の実装では TelexString の値の範囲については厳格な検査は行っておらず、8 ビットの透過性が確保されていることもあるので、実際の運用を通して検討を続けたい。

2.5　ISO2022 の使い方

2.5.1　基本的な方針

ISO2022 の符号拡張法には様々な選択肢があり、十分に検討して決める必要がある。これは次のような事項が含まれる。

1. 7 ビット環境と 8 ビット環境の選択
2. 文字集合を指示する G0 から G3 の使い方
3. ロッキング・シフトやシングル・シフトの使い方
4. 文字集合の指示と呼び出しに関する取り決め
5. 以上の事項を当事者間の合意とするか、明示的にアナウンサで指定するかの選択

透過性の問題を考慮して、当面は次の様に ISO2022 を使用することとする。

- 8 ビット環境
- 初期状態では G0 にテレテックス基本図形文字集合を指示して GL に呼び出し
- 初期状態では G2 にテレテックス補助図形文字集合を指示して GR に呼び出し
- 文字集合の指示は同時に呼び出しを実行

- G0 に様々な言語をエスケープ・シーケンスで指示

また、ISO2375 に登録されている文字集合には、ほぼ同じ内容で一部が異なる文字集合が登録されている場合がある。これらを厳密に別の文字集合として取り扱うと、全く同一の文字が表示されるがマッチしないといった状況が起きる。一方、字形の類似から、同じ文字と単純に判断しても問題となる場合もあり、現状の ISO2022 では登録された文字集合毎に対応せざるを得ないであろう。

以降では、この様な問題について実用上必要であったり、レビジョンの異なる文字集合の扱いなどについて以降で述べる。

2.5.2 制御文字の前の文字集合の切替え

ISO2022-JP[225] では制御文字の前で G0 に ASCII を指示する様にしている。これは、状態の保持を減少させて文字列の処理を容易にする効果がある。いずれを選択しても既存のディレクトリ・サービスとの相互運用性に直接の影響はない。

新たに実装する場合の容易さを考えて、制御文字の前では G0 に ASCII を指示することにする。

2.5.3 改行文字の前の文字集合の切替え

ISO2022-JP2[226] では改行文字の前で G0 に ASCII を指示する様にし、G2 に指示した文字集合の情報を初期状態に戻している。これは、状態の保持を減少させて文字列の処理を容易にする効果がある。いずれを選択しても既存のディレクトリサービスとの相互運用性に直接の影響はない。また、メールのメッセージなどと異なり、改行文字が出現する頻度も少ない。

新たに実装する場合の容易さを考えて、改行文字の前では G0 に ASCII を指示することにし、G2 に指示した文字集合の情報を初期状態に戻すことにする。

2.5.4 94 文字複数バイト文字のエスケープ・シーケンス

JIS C 6226-1978[227]、GB 2312-1980[228]、JIS X 0208-1983[229] を G0 に指示する場合は、表 2.7に示す 2 種類のエスケープ・シーケンスの使用が ISO2022 では許されている。これらは、複数バイト文字集合を G0 だけに指示可能であった 1975 年版の ISO2022

表 2.7: 94 文字複数バイト文字を G0 に指示するエスケープ・シーケンス

文字集合	ISO2022-1986	ISO2022-1975
JIS C 6226-1978	ESC 2/4 2/8 4/0	ESC 2/4 4/0
GB 2312-1980	ESC 2/4 2/8 4/1	ESC 2/4 4/1
JIS X 0208-1983	ESC 2/4 2/8 4/2	ESC 2/4 4/2

に基づいて、既に登録済のエスケープ・シーケンスである。X Window System のコンパウンド・テキストでは 1986 年版を、ISO2022-JP や ISO2022-JP2 では 1975 年版を基本にしている。ISO2022-1986 で、これら ISO2022-1975 のエスケープ・シーケンスが使用不可能となったわけではなく、両方を認めている。

これらは、どちらも受け付ける様にし、出力時は指定された一方で行うこととする。DSA の EDB ファイル (3.3.2を参照) に使用するコードを指定可能に、DUA ではユーザが選択可能とする必要がある。

2.5.5　TelexString と ASCII

現在の ISODE の実装や DSA の運用で、文字集合の扱いについては`TelexString` で一部のヨーロッパ圏の特殊文字が使用できること以外は殆んど意識されておらず、単純にコード・ポイントで比較している。また、テレテックス基本図形文字集合になく、テレテックス補助図形文字集合にある文字を指定すると、DUA では後者にマップする様な実装となっている。

この点から `TelexString` と ASCII で同じ文字の名称で定義された文字は同一視して扱う必要がある。

2.5.6　JIS X 0201-1976 と ASCII

JIS X 0201-1976[230] の左半面 (以降では JIS Roman と表記) と ASCII では表 2.8に示す文字が異なっている。

表 2.8: JIS Roman と ASCII で異なる文字

コード	ASCII	JIS Roman
5/12	backslash	円記号
7/12	vertical line	縦線
7/14	tilde	オーバライン

JIS Roman の実際の使われ方は ASCII と区別されていないのが一般的である。次の様な方法も一案であるが、この方法を実際に使用するかどうかは検討を加える必要がある。

- JIS Roman には表 2.8の文字だけが含まれているとし、他の文字は互換のエンコーディングであると解釈する。
- 文字を受け取って、内部形式にするときは次の手順に従う。
  - ASCII のエンコーディングが使用されていた場合は、全て ASCII の文字集合と解釈する。
  - JIS Roman のエンコーディングが使用されていた場合は表 2.8の文字については JIS Roman の文字集合として扱い、他は全て ASCII の文字集合と解釈する。

- DUA で文字を表示するときは、出力先の能力やユーザの好みによって次から選んだ方式で出力を行う。
  - ASCII だけの表示ができるときは、全て ASCII にエンコードして出力する。
  - JIS Roman だけの表示できるときは、全て JIS Roman にエンコードして出力する。
  - 両方の表示ができるときは、内部形式の ASCII や JIS Roman をそのままエンコードして出力する。
- DUA で文字を表示するとき以外で出力するときは、内部形式の ASCII や JIS Roman をそのままエンコードして出力する。

JIS X 0201-1976 の右半面 (いわゆる半角片仮名) は、ISO2022 では 1 つの文字集合として扱われる。半角仮名の使用は、運用で避けることにするか、強制的に JIS X 0208 にマップするかは今後の検討が必要である。

2.5.7 JIS 文字集合のバージョン

JIS の文字集合にはバージョンの異なるものが存在する。これらを表 2.9に示す。

表 2.9: JIS 文字集合を G0 に指示するエスケープ・シーケンス

文字集合	エスケープ・シーケンス
JIS C 6226-1978	`ESC 2/4 2/8 4/0`
JIS X 0208-1983	`ESC 2/4 2/8 4/2`
JIS X 0208-1990 [231]	`ESC 2/6 4/0 ESC 2/4 2/8 4/2`

これらの違いについては、以下の様な対処の方法が考えられる。

1. 異なる文字集合として別々に扱う。
2. サポートする文字集合を特定のものに決める。
3. 出力先のエンコーディングの違いだけと判断し、全部同じ文字集合と見なす。
4. JIS Roman と同様な方法で同一視する。

4の場合と同様な方法は次のようになる。

- JIS X 0208-1983 を基本的なベースとする。
- 文字を受け取って内部形式に変換する時は次の手順に従う。

  - JIS C 6226-1978 のエンコーディングが使用されていた場合で、JIS X 0208-1983 でコード・ポイントが移動した文字は JIS C 6226-1978 の文字として扱う。それ以外の場合は JIS X 0208-1983 の文字集合として扱う。
  - JIS X 0208-1983 の場合は、そのまま JIS X 0208-1983 の文字集合として内部形式にする。
  - JIS X 0208-1990 の場合は、追加された文字とコード・ポイントが移動した文字については JIS X0208-1990 の文字集合として内部形式にする。書体の変更された文字については JIS X0208-1983 の文字集合として扱う。

- DUA で文字を表示するときは、出力先の能力やユーザの好みによって次から選んだ方式で出力を行う。
  - どれか特定の文字集合を選択した場合は、その文字集合のエンコーディングを使用して出力する。但し、存在しない文字をそのまま出力するか、代替文字で出力するかの選択も可能とすべきであろう。
  - いずれも受け付ける場合は、内部形式をそのままエンコーディングして出力する。
- DUA で文字を表示するとき以外で出力するときは、内部形式をそのままエンコーディングして出力する。

しかし、JIS Roman の場合と異なり、かなり手間のかかる処理が必要となる割に、どこまで区別する利点が得られるかは疑問である。そこで、出力先のエンコーディングの違いだけと判断して、全部同じ文字集合と見なす 3 を採用することにする。

2.6　設計

2.6.1　符号化 (コード 変換) の組み込み

QUIPU に符号化 (コード 変換) を追加する場合には、次の 2 つの方法が考えられる。

- `caseIgnoreString` 属性構文の処理を変更
- Presentation Stream のライブラリを変更

前者は OSI ディレクトリサービス (QUIPU) のライブラリの変更であり、後者は ISODE パッケージの基本的なライブラリに対する変更である。前者は OSI ディレクトリサービスだけの変更であり、当面の実装方法としては容易に作業できることが期待できる。後者の方式は OSI ディレクトリサービス以外の国際化にも有用であることが予想され、最終的に望ましい方式であると思われる。

また、内部形式以外のコードを含めて、符号化の方式は実行時に設定を可能にすべきである。これは、多国語の符号化の方法の検証を行うためと、DUA でユーザの指定による切り替えを可能とするために必要である。

2.6.2 ファイル・コード

ファイルのコードは DSA のデータベースと、一部の DUA で `modify` オペレーションを実行する時に使用される。
ISODE では次の文字は、C 言語の特殊文字を表すのに似た、バックスラッシュに始まるシーケンスで表現している。

1. `&`、`#`、`$`、`%`、`@`とスペース文字
2. 8 ビットの文字を含む印字不可能な文字

DSA だけであれば保守時に編集できるコードであれば何でも良いが、DUA で使用されることを考慮して、ISO2022 の 7 ビット環境にエンコードされた形式の使用をサポートする。

2.6.3 表示コード

表示のコードはファイル・コードと異なり、元の内部形式に戻す必要はない。ISODE でもファイルへの出力と、DUA の出力で関数そのものは同一であり、表示用を表すパラメータによって出力を変更している。
実際は、DUA のユーザー・インターフェイスに依存して決めることとなる。なお、多国語の表示は次の方法による。

- コマンドを実行した結果を Mule[223] 上で表示
- exterm 上の tty の動作
- X Window System の国際化 Athena Widget の利用

2.6.4 内部コード

内部コードは多国語を同時に処理できなければならない。現在のところ、内部コードには Mule version 1.0 の形式の使用を予定している。当面は Mule のコードを流用するが、次の理由から最終的には独自のコードに置き換える予定である。

- Mule が Emacs Version 19 をベースにする過程で、Mule の内部コードそのものが変わる可能性もある。
- GNU GENERAL PUBLIC LICENSE の制約がある。

2.7 今後の予定

多国語化に関する詳細な設計を行い、実装と運用を通して評価を行う予定である。実装は DSA として QUIPU、基本的な DUA として dish、X Window System をベースとし

た DUA として pod を対象とする。運用については、JPNIC や APNIC の情報を提供する試みとも協力して行っていきたい。

　また、多国語化以外にも国際化や地域化に関する重要な問題が存在するので、引続き調査・研究を行いたい。

第 3 章

Relay DSA の実験

3.1 はじめに

OSI ディレクトリサービス [112] の全国実験では、DSA(Directory System Agent) は Internet からアクセスできるホストに置くことを前提としている。しかし、より広範に DSA を展開する上では大きな制約である。そこで、Internet と直接通信が不可能な DSA を利用する方法の一つである、Relay DSA を使用する実験を行った。この結果と問題点について述べる。

3.2 背景

現在の OSI ディレクトリサービスの実験では Internet から IP で到達可能なホストに DSA を設置している。さらに広範囲な運用にあたっては、IP で到達不可能な場合も考慮する必要がある。これは X.500 に限らず、次の様な理由でネットワーク層で直接通信できない場合のネットワークのアプリケーションで、一般に存在する問題である。

- ネットワーク層が IP に限らない、異なるプロトコルの場合
- Firewall を介して接続している場合

これらは、異なるプロトコルのシステムに移行、または共存する場合にも関連する問題である。一般的な解決の方法に次の 2 つがある。

- トランスポート層での中継
- アプリケーション自身による、アプリケーション層での中継

前者は socks[233] などで用いられている方法であり、後者は Internet のメールや NetNews などで広く用いられている方法である。

ISODE[234] では Transport Service Bridge によって、トランスポート層を介したアプリケーションの接続を行うための機能が用意されている。また、QUIPU[215] では Relay DSA という、アプリケーションのレベルによる、直接通信できない DSA に対する解決方法を用意している。

今年度、富士ゼロックス情報システム (以降は特に断りのない限り FXIS と省略) の DSA を接続する過程で使用した Relay DSA の、動作の確認と運用上の検証を以下で述べる。

以降ではネットワーク層で Internet から通信ができないネットワークを便宜上プライベート・ネットワークと呼ぶことにする。

3.3　エントリ情報の取得

3.3.1　Knowledge Information

DSA は、DUA(Directory User Agent) や他の DSA からの問い合わせに答えて、エントリの情報を提供する。DSA は保持していないエントリの情報を要求されると、エントリを保持する DSA の参照情報と共にエラーを DUA に返すか、他の DSA に問い合わせを行ってエントリの情報を結果として DUA に返す。いずれにしろ、DSA は問い合わせを受けたエントリを保持する DSA を知る必要がある。ここでエントリとエントリを保持する DSA に関係する情報が必要である。この情報を Knowledge Information と呼ぶ。

1988 年版の標準では Knowledge Information についての規定は存在しないため、QUIPU では独自の拡張を行っていて、1992 年版の標準にも影響を与えている。以下は QUIPU の場合を述べている。

3.3.2　EDB

QUIPU では DIT 上で同一の親を持つエントリを単位にして管理している。この単位を EDB(Entry Data Block) と呼び、同一のファイルに格納している。その情報元から、EDB は次の 3 つに分類される。

- DSA が情報の正当な提供元であるマスタ
- 他の DSA からコピーされた情報であるスレイブ
- 他の DSA にアクセスした結果残っているキャッシュ

マスタの EDB は BIND[235] における primary server として持つ情報に、スレイブの EDB は secondary server として持つ情報に相当する。

3.3.3　masterDSA と slaveDSA

QUIPU が保持するリーフでないエントリは `masterDSA` 属性を必ず持たなければならない。また、`slaveDSA` 属性を持っていても良い。この 2 つの属性は、DIT 中でエントリの直下にあるエントリを保持する DSA を指定している。

`masterDSA` 属性はマスタの EDB を持つ DSA を、`slaveDSA` 属性はスレイブの EDB を持つ DSA を指定する。EDB を単位にエントリを管理しているので、これでエントリの情報を保持する DSA を判断できることになる。

3.3.4 エントリの情報の取得

DUA または他の DSA からエントリの情報を要求されると、DSA は次の様にしてエントリの情報の所在を調べて処理を行う。

1. エントリの情報を保持していれば、それを返す。

2. エントリの情報を保持していない場合は、エントリの識別名から親のエントリを調べる。親のエントリも保持していない場合は、保持するエントリに達するまで繰り返す。

3. エントリが見つからなかった場合は、quiputailor ファイル[1]に指定された parent で指定された DSA を問い合わせの候補とする。

4. エントリが見つかった場合は、その masterDSA 属性とslaveDSA 属性に指定された DSA を問い合わせる DSA の候補とする。

5. 問い合わせる候補の DSA がわかると、その結果を元に連鎖の処理を行うか、これらの DSA の情報を参照情報と共にエラーとして返す。これは連鎖の処理を DSA が許可されているかや、問い合わせ元のエンティティによって判断する。

6. 連鎖を行う場合は候補の各 DSA に対して問い合わせを行い、結果が得られるか、すべての候補の DSA からエラーが返るまで各 DSA に問い合わせを行う。

parent で指定する DSA は、BIND におけるルート・サーバとして設定するネームサーバといった見方もできる。

3.4 異なるネットワークと Relay DSA

3.4.1 OSI コミュニティ

ISODE はトランスポート層以下に TCP/IP、X.25、CLNP などの様々なサービスを使用するが、相互運用を可能にするためには、これらの異なるネットワークを統一したプレゼンテーション・アドレスで表現できなければならない。実際は、トランスポート層より上の層は OSI に基づいた単一の実装であるため、トランスポート層以下のアドレスの表現が問題となる。

ISODE では TCP/IP、X.25、CLNP などの様々なサービスのアドレスを RFC1277[236] にしたがって表している。この方式では、同一のトランスポート層やネットワーク層を使用し、相互に通信が可能なホストには共通な識別部分を割り当てる。この識別部分を OSI コミュニティと呼び、プライベート・ネットワークに対して新しい OSI コミュニティを作成することも可能である。

[1] QUIPU の動作の設定ファイル

OSI のアプリケーションではアクセスする先をプレゼンテーション・アドレスによって表す[2]。プレゼンテーション・アドレスには、下位の層の RFC1277 に従ったアドレスが含まれているので、プレゼンテーション・アドレスを比較してアクセスする先と直接通信できるかどうかを判断できる。

3.4.2 Relay DSA

プライベート・ネットワーク上の DSA は、3.3.4の手順に従うと Internet 上の DSA と接続できない。しかし、Internet とプライベート・ネットワークの両方に接続しているホストで稼働している DSA が存在しているなら、そこにオペレーションを連鎖して処理することができる。QUIPU の DSA には、連鎖する先の DSA を `relayDSA` 属性として持たせることができる様になっている。

3.3.4の手順で、問い合わせる候補の DSA が決まった時点から後の処理を、さらに詳しく以下に示す。

1. 連鎖を行う場合は、候補の DSA のプレゼンテーション・アドレスを調べる。OSI コミュニティが同一なら、問い合わせを行う。

2. OSI コミュニティが同一でなく、`relayDSA` 属性を DSA 自身が持っていた場合は、`relayDSA` 属性の値の DSA に対して問い合わせを行う。

3. `relayDSA` 属性を DSA 自身が持たないか、問い合わせに失敗した場合は、さらに候補の DSA が他に存在すれば同様に問い合わせが可能かどうか確認する。可能であれば、問い合わせを行う。

4. 問い合わせた DSA から結果が得られたら、これを返し、すべての候補の DSA への問い合わせに失敗した場合はエラーを返す。

`relayDSA` 属性による指定は、通常とは異なる DSA を利用する点で BIND の`forwarders` の指定と似た設定であると言える。

3.5 設定と動作の確認

3.5.1 基本的な実験環境

FXIS は富士ゼロックスと共通の社内 TCP/IP ネットワークを構成していて、firewall を介して WIDE Internet と接続している。Internet に対するプライベート・ネットワークである社内 TCP/IP ネットワークで富士ゼロックスは `129.249` の IP アドレスを、FXIS は `131.221` の IP アドレスを使用し、相互に通信可能である。

図 3.1に関連する DIT を示す。

[2]TCP/IP では IP アドレスと、TCP または UDP のポートの組に相当する。

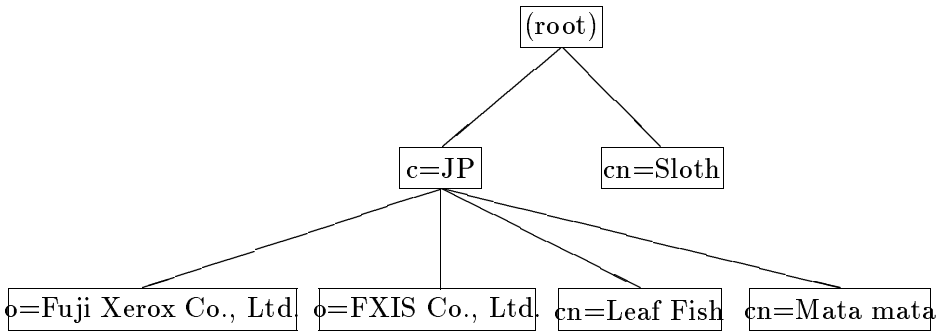

図 3.1: 関連する Directory のオブジェクト

cn=Sloth は Internet 上の DSA である。DIT 中の c=JP より下位にあるエントリのマスタの EDB を保持している。この様にルートの直下にエントリを持つ DSA を Level 0 の DSA と呼ぶ。

c=JP@cn=Mata mata は firewall 上で動作している DSA である。Internet とプライベート・ネットワークの両方にアクセスできる。DIT 中のc=JP@o=FujiXeroxTCP Co., Ltd. より下位にあるエントリのマスタの EDB を保持している。この様に、国を表すエントリの直下に位置する DSA を Level 1 の DSA と呼ぶ。

c=JP@cn=Leaf Fish はプライベート・ネットワークで動作している DSA であり、Internet に直接アクセスできない。DIT 中のc=JP@o=FXIS Co., Ltd. より下位にあるエントリのマスタの EDB を保持している。これも Level 1 の DSA である。

接続関係から見た DSA の配置を図 3.2に示す。

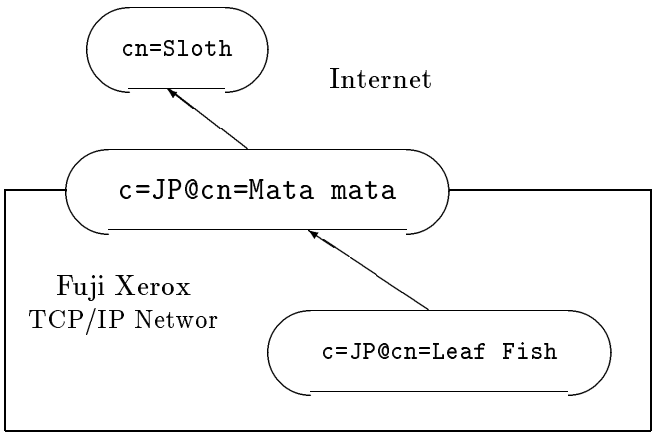

図 3.2: DSA の配置

3.5.2 プライベート・ネットワークでの DSA の立ち上げ

c=JP@cn=Leaf Fish を立ち上げるため、プライベート・ネットワークに対応した OSI

コミュニティを設定した。これは、富士ゼロックスの TELEX 番号を元に [234] に従って作成した。Internet の OSI コミュニティは ISODE の配布に含まれる、一般に使用されているものである。Internet とプライベート・ネットワークに対応する OSI コミュニティを表 3.1に示す。

表 3.1: 設定した OSI コミュニティ

OSI コミュニティ	対応するネットワーク・アドレス
Internet	TELEX+00728722+RFC-1006+03+
FujiXeroxTCP	TELEX+07247983+RFC-1006+01+

c=JP@cn=Leaf Fish はプライベート・ネットワークだけの DIT で動作していたものの設定を変更した。

3.5.3　プライベート・ネットワークから Internet のアクセス

Relay DSA を使用するために、次の様な設定を行った。

- c=JP@cn=Leaf Fish の DSA のエントリに relayDSA 属性を次の様に追加した。

```
relayDSA=c=JP@cn=Mata mata
```

- quiputailor ファイルの dspchaining を on に設定して、他の DSA からのオペレーションを連鎖して処理することを許可した。

この結果、プライベート・ネットワークの DUA を c=JP@cn=Leaf Fish に接続して、Internet の DSA の持つ情報をアクセスできることを確認できた。

3.5.4　Internet からプライベート・ネットワークのアクセス

3.5.3の設定のままでは、c=JP@cn=Mata mata 以外の Internet 上の DSA から、DIT で c=JP@o=FXIS Co., Ltd. より下位のエントリにアクセスできない。これは relayDSA 属性を設定していない場合に、プライベート・ネットワークから Internet にアクセスできないのと全く同じ理由である。すなわち、Internet 上の DSA は c=jp@o=FXIS Co., Ltd. より下位のエントリの DSA の情報を取得できるがネットワーク的に到達できないので接続できない。また、連鎖する先の DSA を取得することもできない。

そこで、relayDSA 属性で指定した DSA の設定を変更し、c=JP@o=FXIS Co., Ltd. のスレーブの EDB を保持する様にした。この結果、Internet 上の DSA は c=jp@o=FXIS Co., Ltd. の情報を c=JP@cn=Mata mata から取得できる様になる。

次善の回避策であるが、これらの変更によって Internet 上の DSA が、プライベート・ネットワーク上の DSA に保持されているエントリの情報を取得可能となった。

3.6 問題点

3.6.1 連鎖によるオペレーションの制限

連鎖による場合はセキュリティの関係から、エントリの変更を伴うオペレーション Add, Remove, ModifyEntry, ModifyRDN を許可していない。Relay DSA では必ず relayDSA 属性で指定された DSA による連鎖を介してオペレーションを行うため、これらのオペレーションを実行できない場合が発生する。例えば、Internet 上の DUA からプライベート・ネットワークの DSA が保持するエントリの情報の修正したり、逆にプライベート・ネットワーク上の DUA から Internet 上の DSA が保持するエントリの情報の修正をしたりすることはできない。

多くの場合には問題とはならないが、c=jp@o=FXIS Co., Ltd. のエントリの変更を、Internet にアクセス可能な DUA から行わなければならなかった。これは、プライベート・ネットワークにある DSA の管理者は、Internet にアクセスできなければならないことを意味している。

3.6.2 DSA の情報の更新

現在の X.500 の実験運用では、DSA が情報を保持するエントリよりも、DSA のエントリを DIT 上では高い位置に配置している[3]。例えば c=JP@o=FujiXeroxTCP Co., Ltd. より下位のエントリの情報を保持する DSA は c=JP@cn=Mata mata である。このようにすることで、エントリの情報を保持する DSA に確実にアクセスできるようにしている。

DSA のエントリがある階層と EDB ファイルの変更のアクセス権の関係から、DSA のエントリの情報の変更は次の様な段階を経て達成される。

1. QUIPU の DSA は、EDB とは別に自身のエントリの情報を保持する。そして、DSA の情報の問い合わせには、これを使って答える。

2. DSA に対する修正のオペレーションは、直接 DSA に対して実行される。他の DSA にアクセスしていた場合は、参照先の情報と共にエラーを DUA に返す。

3. DSA は Modify のオペレーションを自身で保持するエントリの情報に対して行い、DUA に対して応答する。

4. マスタの EDB を保持している DSA は、EDB に含まれている QUIPU の DSA に対して定期的に Read オペレーションを行い、この結果を EDB に反映する。

ところが、プライベート・ネットワーク上の DSA は、これを保持する DSA から直接アクセスできないため、上述の手順による DSA の情報の修正は不可能となる。すなわち、c=JP@cn=Leaf Fish に cn=Sloth は接続できないため、c=JP@cn=Leaf Fish のエントリの情報は全く更新されない。結果として EDB にある DSA のエントリの情報と、DSA 自身が保持する DSA の情報は不一致のままであった。この状態を DUA からのアクセスで

[3] 一部の国では、DMD というエントリに置くなど、独自の方法を取っている場合もある。

修正することはできないので、DSA の動作を制御した上で、EDB のファイルを直接編集する必要がある。

DSA の情報の変更は、Level 2 の DSA を追加するといったプライベート・ネットワーク内だけに関係した変更でも必要なため、これはディレクトリ・サービスを運用して行く上で大きな制約である。

3.6.3　問題の回避

以上の問題は、現在の QUIPU が DSA 間の通信、DSP(Directory System Protocol) で認証をサポートしていないことも原因の一つであり、QUIPU の課題である。

また、現在の Relay DSA の運用上の問題と見なして、上述の問題が起きないような配置で使用することも考えられる。これは DSA を、そのエントリがあるマスタの EDB を保持する DSA と直接通信できる様に、DIT やネットワーク上の位置を調整することである。具体的に、次の様に使用することになる。

- Level 1 の DSA は必ず Internet へのアクセスを可能にする。
- `relayDSA` 属性を持ったプライベート・ネットワーク上の DSA は、Level 2 以下の DSA で運用する。Level 2 の DSA は、組織 (Organization) よりも DIT で下位に位置する組織単位 (Organization Unit) などを担当する DSA である。

3.7　結論と課題

Relay DSA の機能を使用して、Internet に直接アクセスできない組織への DSA の展開を試みた。Relay DSA の機能によって、部分的に相互運用が可能なことが確認できた。しかし、DSA の機能だけによって Internet に直接アクセスできない組織をディレクトリ・サービスに取り込むことは困難であることが判明した。

今後は次の様な方法を考慮しながら、ネットワークの接続に制限がある場合のより有効な接続の方法の検討と実験を行う予定である。

- Transport Service Bridge による接続
- DSA の認証による、制限のないオペレーションの連鎖
- より有効な Relay DSA の方法の考案

第 4 章

OSI ディレクトリサービスによる NIS とのインターフェイス

4.1 背景

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 OSI ディレクトリサービス (X.500)[112] がある。

ISODE WG では OSI ディレクトリサービスと、実際に広く利用されている NIS のインターフェイスを利用する機構の設計および実装を行った。

4.2 NIS および X.500 の問題点

- NIS および NIS+
  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+では、情報の更新管理をユーザ自身で行うことが可能となった。また、ドメインの階層化によって情報の管理も階層化され、分散管理が可能となった。しかし、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。

  このため、広域分散環境における情報も結局はドメイン内で管理を行わなくてはならない。

- X.500
  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。

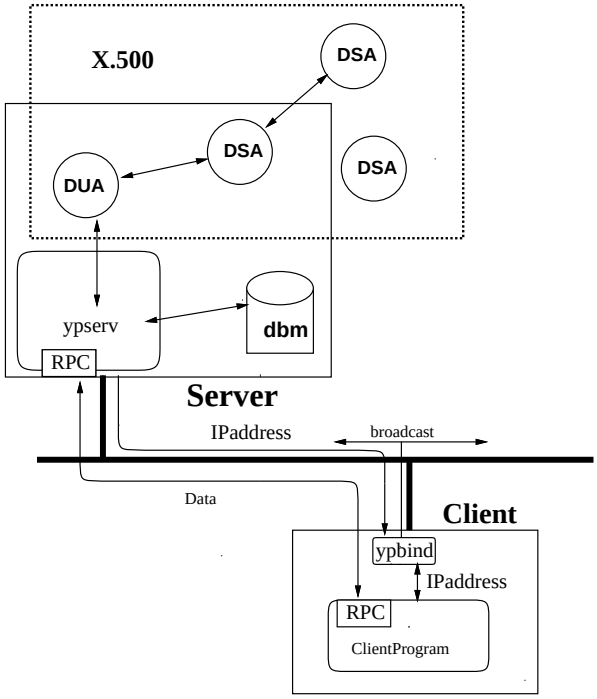

図 4.1: NIS と X.500 の連係

4.3　設計

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 OSI ディレクトリサービス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。

図 4.1 のように、NIS サーバ (ypserv) は通常ローカルの情報 (dbm) から引き出すが、必要があれば、DUA として DSA から情報を引き出し、適切な形にした後クライアントに提供する。

4.3.1　分散的な管理法

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 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。

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、ログインシェルが記述されている。 X.500 での広域分散環境において、パスワード、 gcos フィールド、ログインシェルを管理するものとした。

```
=shigeo:*:9618:96:@c=jp@o=University of Electro Communications@ou=
Computer Science@ou=test@cn=shigeo-nis-test:/home/pdl/shigeo:
/usr/local/bin/bash
```

図 4.2: X.500 で管理されたパスワードエントリの例

```
# by shigeo for NIS
loginshell:        wideAttributeType.4   :CaseExactString
passwd:            wideAttributeType.5   :CaseExactString
```

図 4.3: 新たに定義したオブジェクト

```
# For NIS shigeo
nisuser:  wideObjectClass.10: top : : passwd, loginshell
```

図 4.4: 新たに定義したオブジェクトクラス

パスワードファイルのエントリ中においてユーザー名の先頭に “=” 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、図 4.2 のように /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。

X.500 へは、図 4.3 と図 4.4 のように、NIS のパスワードとログインシェルを記述するための新たなオブジェクトと NIS の管理情報を含んでいることを示すオブジェクトクラスを定義した。

これらを使用して、図 4.5 のように X.500 にユーザ情報を登録する。

NIS はユーザ名の先頭に “=” を持つエントリがあるとその gcos フィールドに記述されている DIT のアクセスをし、必要な情報を検索する。また、gcos フィールドは sn(surname) の属性値を用いる。これにより図 4.6 のエントリがクライアントに渡され、X.500 で管理を行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。

```
objectClass= top & person & newPilotPerson & quipuObject & nisuser
cn= shigeo-nis-test
sn= Shigeo SAKUMA
loginshell= /bin/sh
passwd= pFQGyvwaDy9sU
```

図 4.5: ユーザ情報のエントリ例

```
shigeo:pFQGyvwaDy9sU:9618:96:Shigeo SAKUMA:/home/pdl/shigeo:/bin/sh
```

図 4.6: X.500 の情報によって再構成されたパスワードエントリ

```
"@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 4.7: services の定義例

4.3.2　構造的な管理法

/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。このような情報は世界共通で使用されているもの、ある組織間で使われているものといったようにグループが階層的に管理されていると考えられる。世界的に共通なものは DIT の上位で定義し、ローカルなものほど下位で定義する。情報の検索は下位の層から上の層へ検索を行えば、必要なもの見つかる。また、世界中の計算機に分散している同じ情報が効率よく配置されるはずである。しかし、QUIPU の実装では下位から上位へ向かう検索はない。

4.4　評価

NFS4.0 のソースコードを含む ypserver を SONY CISC NEWS 上に実装した。

OSI ディレクトリサービスのアクセスとして提供されている DAP(Directory Access Protocol) を用いた。X.500 へのアクセスはそれほど速くははないので ypserv 内に dbm で X.500 用のキャッシュも内蔵した。

従来の NIS サーバでの情報のアクセス、X.500 用のキャッシュを用いない NIS サーバでの X.500 を参照する情報のアクセス、X.500 用のキャッシュを用いた NIS サーバでの情報のアクセス応答時間を以下に示す。

これは、従来通りの情報、X.500 を参照する情報をそれぞれランダムに ypmatch コマンドで 500 回検索する時間を time コマンドで 10 回測定し平均をとった。

	従来のサーバ	キャッシュをもたないサーバ	キャッシュをもつサーバ
時間 (秒)	18.2	36.8	22.3

X.500 の DAP そのままでは倍近い時間が情報の参照に費されている。キャッシュを用いることでかなりの時間が軽減されている。キャッシュはキャッシュが参照された後 X.500 が更新されていないかを確かめるアルゴリズムを採用した。したがって X.500 の検索そのものの時間は現れにくいようになっている。

しかし多くの情報が X.500 上で運用された時、その情報の一貫性を考えるならばこのアルゴリズムでは問題が生じるだろう。DAP のアクセス速度と RPC を用いた NIS のプロトコルを考えた時、その情報が RPC のタイムアウトによって渡せない状況を避けるために今回はあえてこのアルゴリズムを採用した。

初期のバージョンでは、X.500 へのアクセスに時間がかかりすぎ、別の NIS サーバーに接続することがまれにあった。

なお、従来の dbm のみによる情報のアクセスに関してはどのサーバーも目だった差は生じなかった。

以上の実装によって以下の利点と問題点が考えられる。

X.500 を NIS に対応したアプリケーションやライブラリから利用することが可能になった。また、NIS はドメインの管理にとらわれない広域分散環境での管理が可能になった。

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、 OSI 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。

問題点として、X.500 のセキュリティがあげられる。信頼できる DUA の認証とアクセス制御の実装をしなくてはならない。X.500 のセキュリティパッケージはあるが実装している DUA, DSA は少ない。

4.5 今後の課題

X.500 の検索速度などをおぎなう目的でキャッシュを実装したが、X.500 と NIS との一貫性のとれた環境の妨げになっている。NIS と X.500 との情報を効率良く配置できるような格納法を検討する必要がある。

X.500 のプロトコルとして、DAP 以外にも LDAP (Lightweight Directory Access Protocol)[237] というものもある。このようなものの積極的な利用も必要だろう。

第 5 章

OSI ディレクトリサービスを使用したネットワーク管理システム IPANeMa

5.1 はじめに

IPANeMa(イパネマ)[238] は、IPA Network Manager の略で、IPA(情報処理振興事業協会）と WIDE プロジェクトが共同で開発しているネットワーク管理システムである。現在、この IPANeMa と OSI ディレクトリサービス [112] を使って、WIDE インターネット管理の実験を行なっているので、それについて述べる。

5.2 IPANeMa はどんな管理システムなのか

IPANeMa は、マルチベンダー. マルチプロトコルで、しかも、大規模なネットワークを管理するという問題を、オブジェクト指向分析手法に従って分析し、分析、設計、実装、という全てのステップをオブジェクト指向によって実施したものである。すなわち、拡張、メンテナンスの容易さを重視したシステムであると言うことができる。このことは、ネットワークの進歩に従って、ネットワーク管理の概念が変化して行けば、それに合わせて変化することができる、ということを意味している。しかし、はたして、本当にネットワークの進化に合わせて IPANeMa も進化して行けるか、それを、今まさに、進歩の真只中にある WIDE インターネットで実験運用することによって検証する。

その前に、現時点における最低限の管理がきちんとできることを確認しなければならない。最低限の管理とは、以下のようなことである。

1. ネットワークリソースの構成図（ネットワークマップ）が書けること。ネットワークリソースとは、LAN を構成するイーサネットや FDDI、LAN と LAN をつなぐ専用線等のメディア、また、ルーター、ブリッジ、重要なサービスを載せているホスト、等である。

2. リソースから必要な管理情報を収集できること。

3. ネットワークリソースの構成図の情報と、リソースから収集した管理情報を関連付けて保持し、それに基づいた基本的な管理処理 (display) ができること。

4. 大規模なネットワークを管理する時、いくつかの管理領域ごとに、ネットワークリソースの構成図が作成され、管理情報の収集が行なわれ、何らかの管理処理が行なわれることになるが、この時、管理領域どうしの関係が明らかでなければならない。つまり、管理領域の認識ができ、その関係を明らかに示すことができなければならない。

5. 管理領域ごとに、管理システムが存在する時、それらは、各管理領域の情報を、互いにやり取りすることができなければならない。

IPANeMa はオブジェクト指向の管理システムである。それはどういうことなのかを、これから説明する。まず、より、わかりやすいユーザーインタフェースである、と言うことができる。なぜならば、全ての情報と機能をオブジェクトとして保持しているからである。

世界に実際に存在する全てのモノ（オブジェクト）は、属性と機能を持っていて、モノどうしは互いに関連し合っている、と考えるのがオブジェクト指向である。この世界で何かが起こる時、それは、全て、あるモノからモノへ、関連の糸をたどって、次々とそれぞれのモノの機能が呼び出されて実行されることによって、ある結果に達するのである。

たとえば、車が目の前にあると想像してみてほしい。車はたくさんの部品から成っている。それぞれの部品は、さらに細かい部品から成っている。この車の、「走る」という機能を実行するとどうなるか。まず、キーを回してエンジンを始動する。そしてアクセルを踏むと、エンジンから車輪に力が伝わって「走る」ことができる。エンジンという部品から車輪という部品に力が伝わるまでには、他のもっと細かい部品達の機能がフルに使われているのである。

IPANeMa がオブジェクト指向であるということは、現実のモノ達、すなわち、ネットワークリソースが、属性と機能を持ったオブジェクトとして、IPANeMa というシステム上に存在している、ということである。

ユーザーは、現実にルーターやホストに対するのと同じ感覚で、それらを反映したオブジェクトに対することができる。管理という目的を果たす、という視点で現実のネットワークを眺めた時、この現実を正確に反映するように、オブジェクトの設計ができていれば、それは、より自然でわかりやすいユーザーインタフェースを提供するための大きな要素となる。逆に、どんなにすぐれたグラフィックインタフェースがあっても、もともとの問題を正確に把握したオブジェクト構成になっていなければ、わかりやすいユーザーインタフェースとはならない。

実験の中で、本当に使いやすいインタフェースかどうか検証することは、オブジェクト達が現実を正確に反映しているかどうか検証することでもある。

IPANeMa がオブジェクト指向であることの、もうひとつの利点は、変更、拡張のしやすさである。たとえば、車の場合、スタイリングを変えたければ、外装部品を取り替える。もっとパワフルに走るようにしたければ、エンジンを取り替える。このように、部品の交換や追加によって変更、拡張ができるのである。ただし、オブジェクトの設計がうまくできていれば、である。

どのような変更、拡張が行なわれるか、ある程度予想して設計されていなければならない。つまり、変更、拡張が行なわれない部分、共通の概念がうまく抽出されて、それ

が、オブジェクトライブラリとして充実すれば、それ以降の変更、拡張は容易にできる。

先に述べた、1から 5の最低限の管理ができるだけでは、管理システムとして充分ではない。会社や組織の管理ポリシーに合わせてカスタマイズできなければならない。さらに、新しいタイプのサービスや、機器が現れた場合は、その管理ができるように拡張できなければならない。

現在の IPANeMa が提供しているオブジェクトライブラリが、共通の概念をうまく抽出しているかどうか、今後の変更、拡張が容易にできるかどうか、WIDE インターネットで実験運用しながら検証し、本当に充実したライブラリとしていかなければならない。

5.3　IPANeMa のシステム構成

図 5.1 に、IPANeMa のシステム構成を示す。IPANeMa として、提供されるソフトウェアは、Managing AP と、IPANeMa オリジナルのエージェントである。管理すべきネットワーク上のあるホストを、Managing Node として、そこに、Managing AP を載せる。管理されるべきリソースには、何らかのエージェントが載っていなければならない。

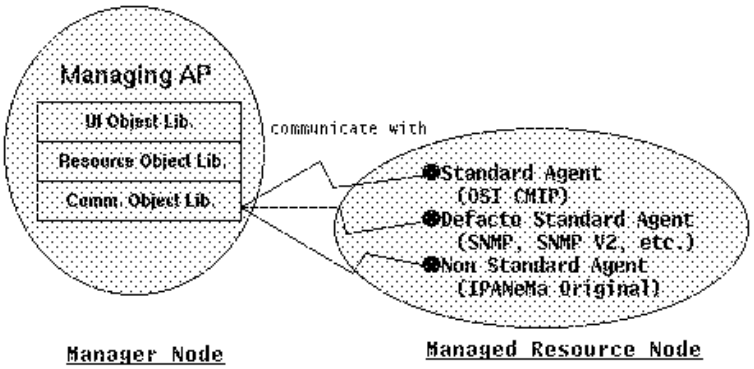

図 5.1: IPANeMa のシステム構成

Managing AP は、３つのオブジェクトライブラリ（ユーザーインタフェースオブジェクトライブラリ、リソースオブジェクトライブラリ、通信オブジェクトライブラリ）から成っている。これらのライブラリのオブジェクト構成は、オブジェクト指向分析の結果、得られたものである。

リソースオブジェクトは、通信オブジェクトを使ってリソース上の管理情報を得ることができる。この時、通信オブジェクトは、リソース上の管理情報構造を知っていて、リソースオブジェクトが必要とする管理情報構造とマッピングした上で、リソース上の何らかのエージェントと通信する。リソースオブジェクトは、ネットワーク上のリソースを反映したオブジェクトで、物理的位置情報と論理的位置情報を持ち、通信サービスプ

ログラムを部品オブジェクトとして包含している。さらに、通信オブジェクトを使って、必要な情報を収集する、という機能を持っている。

以下に、現在の IPANeMa が扱っているリソースオブジェクトの継承木を示す。

```
Top     - CommEntity            - TCPEntity
                                - IPEntity
                                - OSINetworkEntity
                                - OSITransportEntity
                                - NSLevel1-2Entity
                                - NSLevel3Entity
        - Connection            - TCPConnection
                                - TransportConnection
                                - SPPConnection
        - LogicalNetwork        - IPNetwork
                                - NSNetwork
        - Equipment             - Host                  - Router
                                - TransferEQ
        - NetArcEntity
        - MDomain
        - Service               - MailMasterService
                                - NFSService
                                - NISMasterService
                                - NSFileService
                                - NSCHService
                                - NSMailService
                                - NSPrintService
                                - PacketMonitor
        - PhysicalPort
        - PhysicalMedia         - FDDI
                                - Ether
        - PhysicalNetwork       - FDDINet
                                - EtherNet
```

以後、IPANeMa と言えば、Managing AP を指すものとする。

5.4 ネットワークリソースの構成図

管理対象は、WIDE バックボーンと、そこにつながっているいくつかの組織である。これらの組織は、IPANeMa の実験に協力することを表明している組織である。WIDE バックボーンを構成する各 NOC ごと、そして、これらの組織ごとに、IPANeMa を使ってネットワークリソースの構成図を作成する。

例えば、WNOC-TYO(東京 NOC) は 16 個のルーターと 1 本のイーサネットで構成されている。図 5.2 に WNOC-TYO の構成図を示す。イーサネットを意味する線、ルーターを意味するアイコンは、それぞれ、イーサネット、ルーターというオブジェクトを表し

ている。つまり、ネットワークリソースの構成図を作成することは、IPANeMa の中にオブジェクトインスタンス[1] を作成することである。

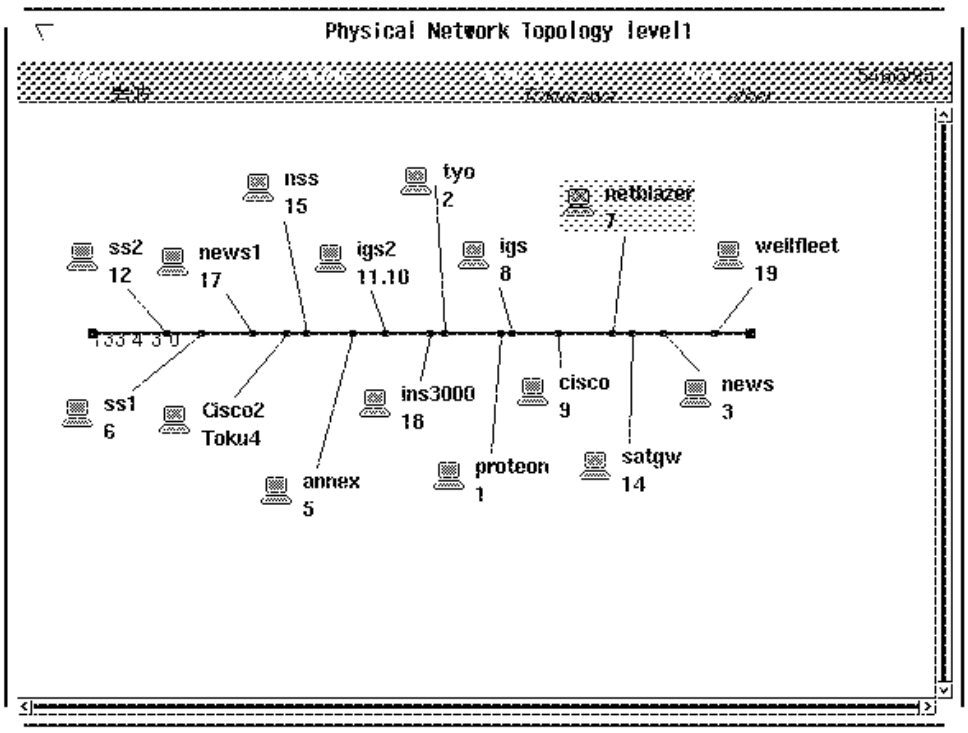

図 5.2: WNOC-TYO の構成

この時、如何にして自動的に構成図を作成するか、あるいは、一度作成された構成図の正しさを如何にしてチェックするか、という問題がある。そのやり方はいろいろ考えられるが、それはネットワークの運用方法に依存する。この部分は、今後、WIDE インターネットの運用方法に合わせて徐々に充実させて行くこととして、現在の IPANeMa は、管理者がイーサネット、ルーターというオブジェクトを、ひとつひとつ登録していくようになっている。

ただし、ルーターを構成する部品は、ルーターと共にシステムが勝手に登録する。さらに、ルーターに標準的な管理通信エージェント[2] が載っていれば、それと通信して、そのルーターにつながっているメディアの情報を見て、自動的にメディアオブジェクトを登録する。

[1] オブジェクトの定義に従った個々の実体。たとえば、「車」という概念に対して、実際のそれぞれの車は「車」のインスタンスということになる。

[2] 標準的な管理通信プロトコルとして、TCP/IP 上の SNMP と、OSI の CMIP がある。

5.5 リソースから管理情報を収集

市場に出ている主なルーターやブリッジ等には、標準で SNMP エージェントが載っている。また、SNMP のフリーソフトウェアも出ている。SNMP は TCP/IP 上の管理通信プロトコルである。そこで、WIDE バックボーンの各ルーター、各組織のルーターやホストには、できるだけ SNMP エージェントを載せることとする。

IPANeMa は、最初にルーターやホストが登録される時、その他必要に応じて、SNMP で通信し、必要な情報を得ることができる。構成図の各アイコンには、“display” メニューがついている。これによって、その属性や部品を見ることができる。部品の部品や属性を見るためには、リストされた部品をセレクトして “display” メニューを選べば良い。

SNMP で通信した結果は、部品や部品の属性という形で表される。図 5.3に “display” の例を示す。

5.6 情報の保持と管理処理

IPANeMa は全ての情報をオブジェクトとして保持している。オブジェクトは属性と機能を持つものであるが、他のオブジェクトとの関係も属性として持っているのである。構成図の作成によって各オブジェクトを登録すると、それは、構成情報の登録であり、しかも、各リソース上の管理情報の登録でもある。なぜなら、登録のときに、通信して管理情報を得ているからであり、また、各オブジェクトはそれぞれ自分が必要なリソース上の管理情報を得るための機能を持っているからでもある。

管理処理とは、収集した管理情報の処理である。最も単純な管理処理は、情報を見せること、 “display” することである。それ以外の管理処理は、各組織の管理ポリシーに大きく依存する。

IPANeMa では、ルーターやホストの部品として、サービスを登録することができる。サービスとは通信サービスのことであるが、ネットワークモニタリング等の管理サービスも含まれる。いくつかの管理サービスを登録し、そのサービスと通信することによって、その管理機能をコントロールすることができる。

IPANeMa は、オリジナルのモニタリングサービスを提供している。このモニタリングサービスと IPANeMa Managing AP は、オリジナルプロトコルで通信する。

たとえば、あるホストにモニタリングサービスを登録して、そのサービスを “display” すると、図 5.4のようなウィンドウが現われる。 “realtime monitor” ボタンを押すと、そのホスト上のモニタリングサービスの realtime monitor 機能が呼ばれて、モニター結果が図 5.5のように、グラフと表で表示される。

モニタリングサービス以外のサービスについては、WIDE インターネット、または、各組織の管理ポリシーに合わせて、それぞれ、カスタマイズする。

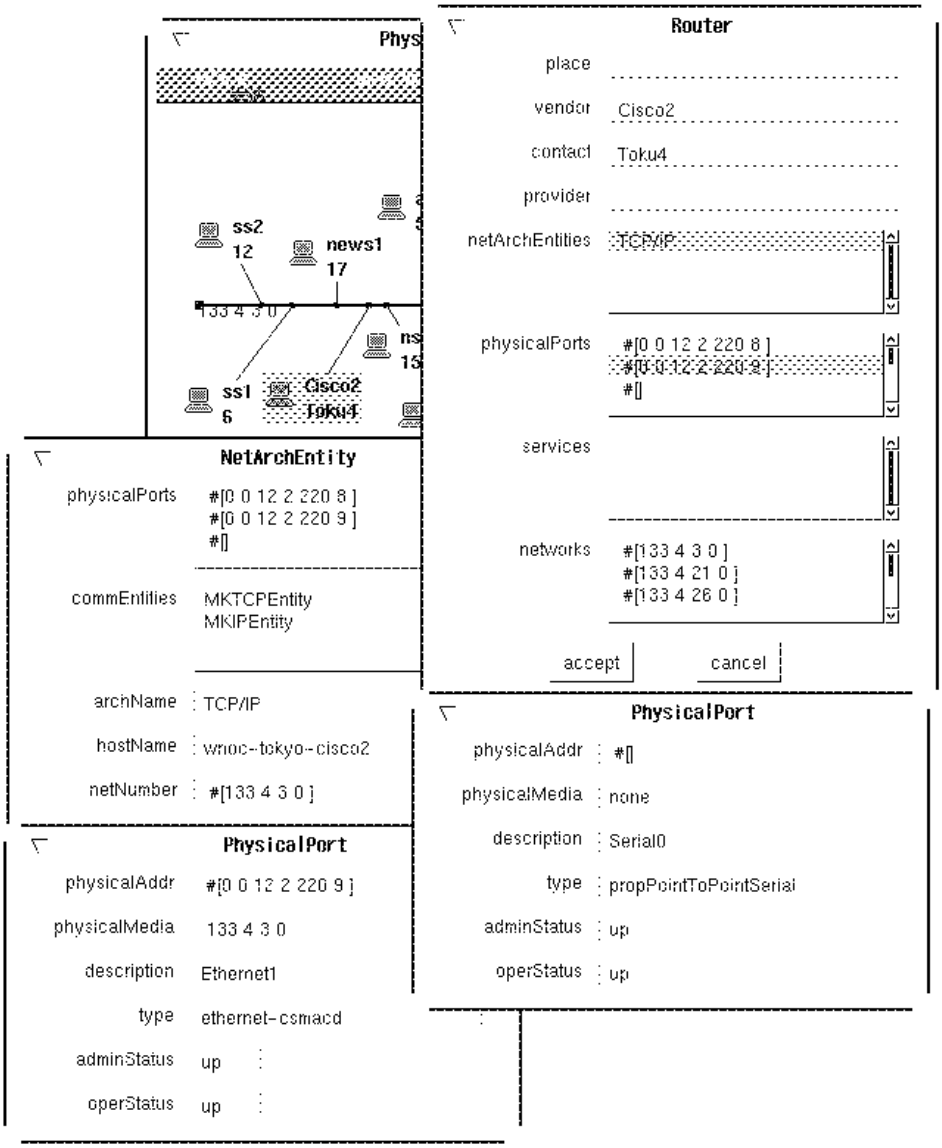

図 5.3: “display” の例

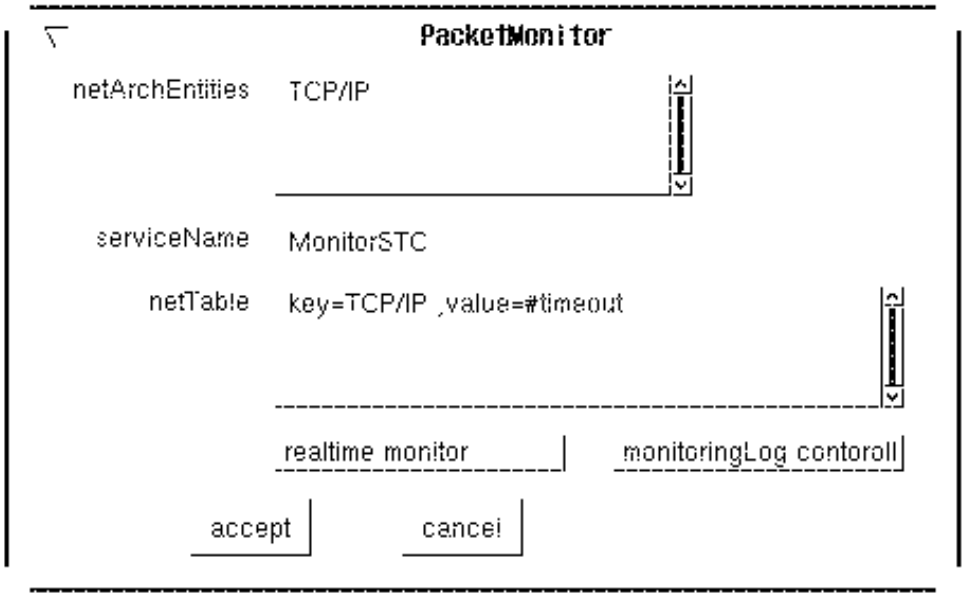

図 5.4: IPANeMa のモニタリングウィンドウ

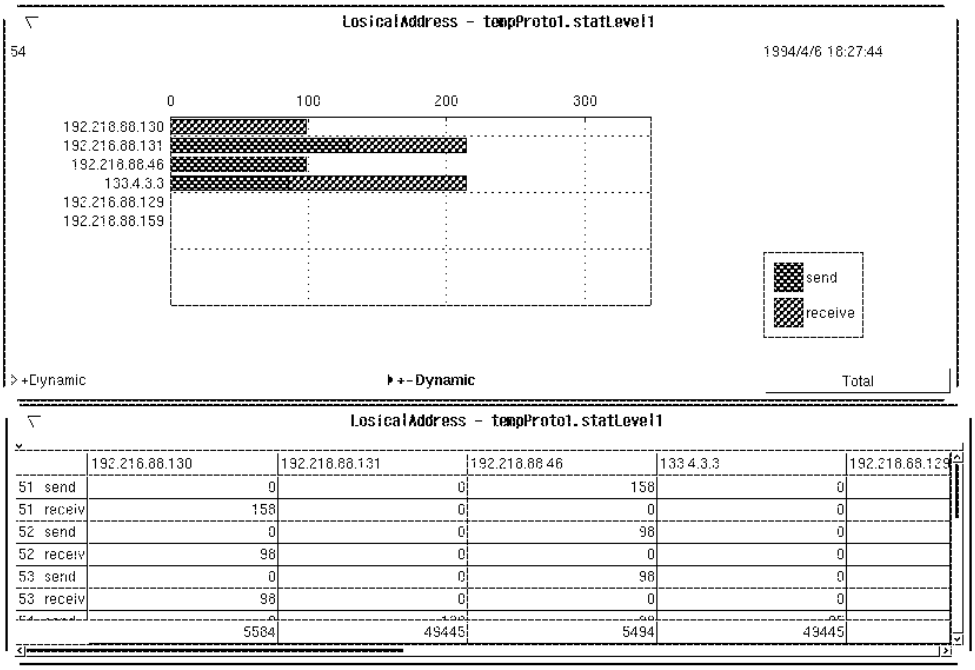

図 5.5: IPANeMa のモニタ表示

5.7　管理領域の認識

現在、世界に広がるインターネット網は、主に、DNS によって管理されている。そこで、原則として、DNS のドメインを、ひとつの管理領域とする。すなわち、WIDE バックボーンは、ひとつの管理領域となり、これにつながる各組織は、それぞれ、ひとつの管理領域となる。

IPANeMa では、管理領域は、いくつかの物理ネットワークによって構成されている。WIDE バックボーンは 9 つの物理ネットワークによって構成されている。それは以下の通りである。

- WNOC-FUK (WIDE Fukuoka Network Operation Center)
- WNOC-HIJ (WIDE Hiroshima Network Operation Center)
- WNOC-KYO (WIDE Kyoto Network Operation Center)
- WNOC-NAR (WIDE Nara Network Operation Center)
- WNOC-OSA (WIDE Osaka Network Operation Center)
- WNOC-SFC (WIDE Fujisawa Network Operation Center)
- WNOC-SND (WIDE Sendai Network Operation Center)
- WNOC-SPK (WIDE Sapporo Network Operation Center)
- WNOC-TYO (WIDE Tokyo Network Operation Center)

WIDE ドメインと各物理ネットワークを登録する。物理ネットワークの登録とは、すなわち、ネットワークリソースの構成図を作成することである。各物理ネットワークどうしの関係、および、他のドメインとの関係を、IPANeMa で示したのが図 5.6 である。

図 5.6 で、大きな楕円は物理ネットワークを、小さな楕円は他の管理領域を意味する。Connect Other PN というウィンドウは、WNOC-TYO と IPA の関係を詳しく表示したものである。WNOC-TYO のあるルーターの、あるポートが、64k の専用線で、IPA のあるルーターの、あるポートにつながっていることを示している。

WIDE の各 NOC には、もっとたくさんの組織がつながっているが、ここでは、今回の IPANeMa 実験に協力している関係組織だけを表示している。全ての組織を見たければ、各 NOC を示すアイコン上で “list all domain” というメニューを選択すると、全ての隣接ドメインをアイコンでリストした、別のウィンドウが表示される。

各管理領域の管理者は、自分の担当領域が、ネットワーク全体の中でどこに位置しているか、認識していなければならない。つまり、他の管理領域とどのようにつながっているか、知っていなければならない。各管理領域ごとに、IPANeMa を使って、それぞれ、物理ネットワークを登録し、他の管理領域との関係を登録する。

管理者は、他の管理領域の情報を必要とする場合がある。この時、IPANeMa は、他の IPANeMa と情報交換できなければならない。

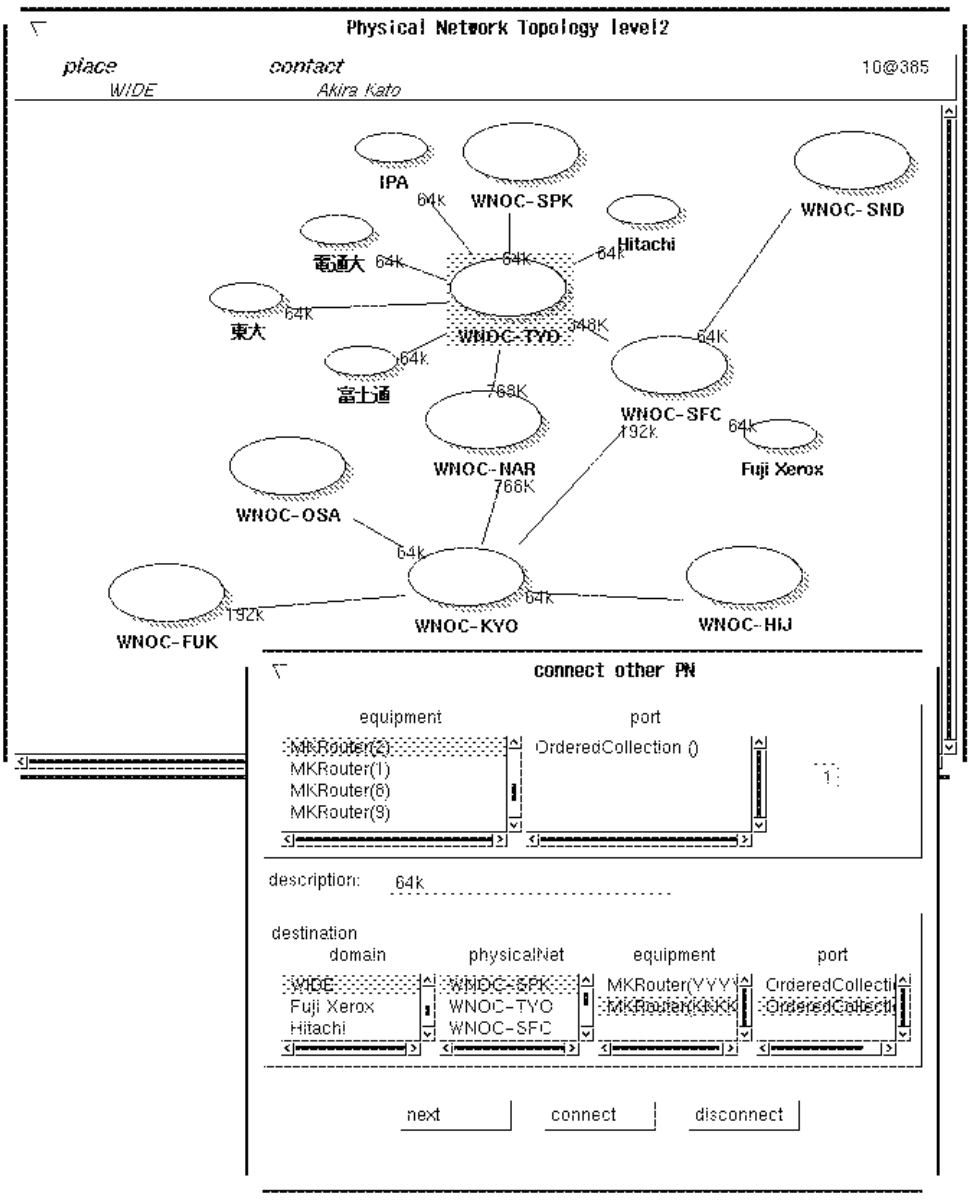

図 5.6: IPANeMa におけるネットワークの関係

5.8　管理システムどうしの情報交換

IPANeMa どうしは、直接、情報交換は行なわない。その代わり、OSI ディレクトリサービス [239] を介して情報交換を行なう。現在、WIDE インターネットでは、ISODE を使った OSI ディレクトリサービスの全国実験を行っている。IPANeMa は、この OSI ディレクトリサービスにアクセスするインタフェースを持っている。各管理領域ごとに、IPANeMa で作成されたオブジェクトインスタンスの集合が、バイナリモードで OSI ディレクトリサービスに格納される。各管理領域の情報を OSI ディレクトリサービスを介してやり取りすることによって、広範囲のネットワークの情報を得ることができる。

IPANeMa としては、管理領域の定義の仕方についての規則等はない。管理者の都合のよいように定義すればよい。しかし、OSI ディレクトリサービスを介して各管理領域の情報をやり取りするためには、管理領域とは何者か、ある程度はっきりさせて、共通の認識を持つようにしなければならない。そこで、原則として、管理領域は、DNS が使用するドメインの概念と一致するように定義することにする。

原則として、DNS に登録されているドメインごとに、管理領域を定義する。その管理領域ごとに作成されたデータベースは、IPANeMaDomain として Directory に登録される。IPANeMaDomain は、属性として、DNS ドメイン名 (複数)、ネットワーク番号 (複数)、隣接ドメイン (複数)、ネットワークプロジェクトであるかどうかのフラッグ、バイナリデータを持つ。たとえば、fxis.fujixerox.co.jp をひとつの IPANeMaDomain として登録すると次のようになる。

```
DN = { c=jp@o=FXIS@ou=IPANeMa-FXIS }
    属性 :  DNS ドメイン名 :    fxis.fujixerox.co.jp
            ネットワーク番号 :  133.214.3.0, 133.214.4.0
            隣接ドメイン :      { c=jp@o=Fuji Xerox@ou=IPANeMa-KSP }
            NetProjectFlag :    off
            バイナリデータ :    IPANeMa-FXIS のオブジェクトインスタンスの集合
```

この IPANeMaDomain は、OSI ディレクトリサービスのツリーの o=FXIS の直下に登録される。OSI ディレクトリサービスのツリーのどこに登録するかは、各組織で決める。以下に、KSP, WIDE をそれぞれひとつの IPANeMaDomain として登録した場合の例を示す。

```
DN = { c=jp@o=Fuji Xerox@ou=IPANeMa-KSP }
    属性 :  DNS ドメイン名 :    ksp.fujixerox.co.jp
            ネットワーク番号 :  133.214.1.0, 133.214.2.0
            隣接ドメイン :      { c=jp@o=FXIS@ou=IPANeMa-FXIS }
                                { c=jp@o=Fuji Xerox@ou=IPANeMa-YBP }
                                { c=jp@o=Fuji Xerox@ou=IPANeMa-Ebina }
                                { c=jp@o=WIDE@ou=IPANeMa-WIDE }
            NetProjectFlag :    off
            バイナリデータ :    IPANeMa-KSP のオブジェクトインスタンスの集合
```

```
DN = { c=jp@o=WIDE@ou=IPANeMa-WIDE }
    属性 :  DNS ドメイン名 :     wide.ad.jp
            ネットワーク番号 :  133.4.1.0, 133.4.2.0
            隣接ドメイン :      { c=jp@o=TISN@ou=IPANeMa-TISN }
                                { c=jp@o=JAIN@ou=IPANeMa-JAIN }
                                { c=jp@o=SINET@ou=IPANeMa-SINET }
                                ................
            NetProjectFlag :    on
            バイナリデータ :    IPANeMa-WIDE のオブジェクトインスタンスの集合
```

WIDE のようなネットワークプロジェクトは、他のネットワークプロジェクト、そして、数十、あるいは、数百の LAN を隣接ドメインとして持っている。

管理者は、DNS ドメイン名や、ネットワーク番号等で、OSI ディレクトリサービスをサーチし、他の管理領域のオブジェクトインスタンスの集合をロードすることができる。たとえば、IPA の管理者が、WIDE バックボーンの情報を知りたいと思ったら、wide.ad.jp という DNS ドメイン名でサーチできる。そうして、WIDE バックボーンのオブジェクトインスタンスの集合をロードすると、図 5.2 や図 5.6 のようなウィンドウを表示させることができる。すなわち、WIDE バックボーンの構成を知ることができる。さらに “display” メニューによって、各機器が現在どういう状況にあるかを知ることもできる。

ネットワーク管理システムとして、OSI ディレクトリサービスをどのように利用するべきかについては、まだ、十分な検討が行なわれていない。実験を進めながら、より深く検討して、もっと充実させなければならない。

5.9 IPANeMa の変更・拡張

IPANeMa は先に述べたように、オブジェクト指向のシステムである。従って、変更・拡張とは、オブジェクト構成の変更、オブジェクトの機能や属性の変更、あるいは追加、そして、新しいオブジェクトの追加、のことである。

IPANeMa の運用実験において、先の 1から 5 の基本的な管理機能に重大な問題が発見されれば、現在のオブジェクト構成を変更することもあり得る。現在のオブジェクト構成について、詳しくは、IPANeMa User's Guide を参照されたい。

現在、予想している変更・拡張は、以下のようなことである。

- 変更 -

1. 各組織ごとに異なる、ネットワーク運用管理のポリシーに従った、通信サービスオブジェクトの機能の変更（カスタマイズ）。通信サービスオブジェクトとは、NISMaster, MailMaster, あるいは、モニターサービス等を表すオブジェクトのことである。

- 拡張 -

1. 新しい種類のネットワークリソース（ATM 交換機、無線 LAN 等）が追加された時、それを認識するために、新しいオブジェクトの定義を追加する。

2. 新しい種類の通信プロトコル（TCP/IP, OSI, XNS 以外）が現われた時、それを認識するために、新しいオブジェクトの定義を追加する。

3. 新しい種類の管理通信プロトコル（SNMP V2 等）が現われた時、それを認識するために、新しいオブジェクトの定義を追加する。

4. 新しい種類の通信サービスが現われた時、それを認識するために、新しいオブジェクトの定義を追加する。

以上のような、変更・拡張のためのインタフェースを提供する予定である。しかし、その前に、現在のオブジェクト構成が十分に実用的であることが、ある程度実証されなければならない。そのために、しばらくの間、運用実験を行なった後、その時点のオブジェクト構成に基づいて、変更・拡張のためのインタフェースを作成する。

5.10　ネットワーク管理標準化アクティビティとの関係

さまざまなネットワーク管理標準化アクティビティがあるが、それらは全て、キャリアー、または、ベンダー主導である。すなわち、リソースが提供する管理情報の体裁の統一と、それを管理システムに与えるための管理通信プロトコルの統一をめざしている。

IPANeMa はこれらの標準化アクティビティのアウトプットのうち、利用可能なものを利用して、実際のネットワークを管理するシステムである。つまり、キャリアーやベンダーによって、標準化された管理エージェントが提供されていれば、それを利用する。その他、正式に標準化されていなくても、利用できるものを利用して、今現在のネットワークを実用的に管理することを最大の課題としている。

標準化が進んで、より高度な管理サービスを提供する機器、あるいはソフトが現われたら、それに対応するオブジェクトの定義を追加する。

5.11　IPANeMa の配布

IPANeMa はフリーソフトウェアである。anonymous ftp で、ソースコードとドキュメントを得ることができる。動作環境や条件についてはドキュメントを参照のこと。

[anonymous ftp サーバー] ftp.mgt.ipa.go.jp:/pub/IPANeMa

第 6 章

その他の実験・研究

6.1 新しいデータ属性の追加の試み

OSI ディレクトリサービスには様々な型の情報を提供する事ができる枠組が用意されている。ISODE パッケージの QUIPU では主にテキスト情報を提供する事に主眼を置いているが、最近になってモノクロやカラーの静止画像、音声データを提供する事ができるようになった。

しかし、それら以外にも OSI ディレクトリサービスで検索したい情報には多様なものがある。そのため、ISODE WG により長い文章が入っているテキストデータを扱えるように、また、名古屋工業大学により、MPEG フォーマットによる動画像データを扱えるように、QUIPU を改造した。実際に行った作業は以下の通りである。

- 新しい属性用の AttributeType, AttributeSyntax, ObjectClass の OID を追加する
- 属性構文型を MPEG:file とする MPEG ファイル用の属性と、Text:file とするテキストファイル用の属性の OID を追加する
- MPEG と Text の属性を扱えるよう、属性構文の解釈部に処理ルーチンを追加する
- MPEG, Text 型の属性を持っているデータを表示するツールを追加する

この作業によって、動画像や長い文章を OSI ディレクトリサービスで扱う事ができるようになった。

しかし、この方法では扱いたい属性型が増える度に OSI ディレクトリサービスのソフトウェアを改造しなければならず、汎用性が乏しい。この問題を解決するためには、属性型の情報やそれを取り扱うツールなどを自動的に拡張する事ができるようにしなければならない。これについては今後さらに研究を行う必要がある。

6.2 WWW による OSI ディレクトリの検索の試み

OSI ディレクトリサービスでは多様な情報を提供する事ができるが、実際にその情報を検索するユーザインタフェースツールはそれほど数多くある訳ではない。一方最近 WWW (World Wide Web) に代表される数種類の「情報検索システム」が登場してきた。特に

WWW のユーザインタフェースツールは、初心者でも比較的簡単に各種の情報を検索することが出来る。

この WWW などから OSI ディレクトリサービスの情報を検索する試みは、すでにアメリカやヨーロッパで行われているが、日本ではまだ行われていない。WIDE ISODE WG では、OSI ディレクトリサービスの様々な応用方法との比較や、国際化された OSI ディレクトリサービスの検証などを行うために、日本で WWW から OSI ディレクトリの情報を検索する事ができるようにしている。

第 7 章

今後の活動

1993 年度の ISODE WG は、OSI ディレクトリサービスにより様々な情報を提供する機構と、その応用方法の研究を行ったが、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 の本格的な研究を行う。

7.1 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の国際化の提案と実装
- ネットワークで共有すべき情報の提供方法 (ディレクトリサービスの本来の使い方) の研究。
- ネットワークのコスト情報収集・提供システムとしてのディレクトリの研究 (最適なアクセス先を決定するための情報提供。ポリシールーティングの経路を決定するための情報提供)
- PEM に関する情報提供システムとしてのディレクトリの研究
- ネットワーク管理情報 (IPANeMa) に関する情報提供システムとしてのディレクトリの研究
- ディレクトリサービスでの JPNIC, APNIC 情報の提供に関する研究
- ディレクトリサービスでのセキュリティ機能の研究
- ディレクトリの名前空間 (ツリー構造) についての検討

7.2 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 での MHS の実験環境の整備
- MHS の国際化の提案と実装
- MHS と MIME とのゲートウェイ機能の検討と実装
- MHS と PEM とのゲートウェイ機能の検討と実装