マルチメディア通信と分散処理 42－10
（１９８９．５．１９）

WIDE資源管理機構

嵯峨 和幸　石河 裕子　海野 英俊　佐藤 智満
慶應義塾大学

村井 純
東京大学

Abstract

大規模広域分散環境上に置かれている資源を効率良く使用するためには、それらの資源すべてを統一的に管理しなければならない。各資源を管理するためには、それを識別するための名前が必要になる。現状の名前管理機構は、ローカルエリアネットワークを基盤に作られているものであるため、大規模広域分散環境上で利用することはできない。ここでは、広域分散環境上に階層的な名前空間を作り、この空間を用いて、分散された資源を統一的に管理する機構を提案する。更に、ユーザやアプリケーションプログラムと、この名前空間のインタフェイスとなる名前サーバ機能を提案する。

Resource Management Function of WIDE

Kazuyuki Saga　Yuko Ishikawa　Hidetoshi Unno　Tomomitsu Sato
Keio University

Jun Murai
University of Tokyo

Abstract

Design of resouce name space and name server functions for WIDE (Widely Integrated Distributed Environmnt) are discussed. Experimental implementation of name server and application are introduced for evaluation of the design.

1 はじめに

WIDE(Widely Integrated Distributed Environment)プロジェクト[2][1]では、慶應義塾大学、東京工業大学、東京大学、大阪大学をはじめとするいくつかの組織間を接続し、大規模広域分散環境の構築を行っている。大規模広域分散環境上に分散された資源を共有するためには、あらゆる資源に識別子をつけなければならない。この識別子が複数の資源を指していると、どれをアクセスすべきであるか決定することができなくなる。従って、各資源にはユニークな識別子をつけなければならない。

実際に資源をアクセスするユーザやアプリケーションプログラムは、わかりやすく、シンプルな表現を用いてその資源をアクセスできなければならない。この表現が、資源につけられる"名前"である。この名前は、資源につけられた識別子と多対1のマッピングをもっている。識別子は、各資源に一つしかつけられないが、名前は、一つの資源に複数つけることができる。

一般に、各資源の名前は接続された各々の組織で独立して決められる。つまり、各管理母体毎に名前空間を持ち、その名前空間の中で資源にユニークな名前をつけている。従って、この名前空間を利用して、複数の組織にまたがる分散環境上におかれた各資源をアクセスすることは殆んど不可能である。例えば、同姓同名の人間がいるように、同じ名前の資源が異なる組織上に複数存在してしまうことは明らかである。そこで、これらの独立した名前空間をすべて含む論理的な名前空間が必要となるのである。また、アプリケーションプログラムが、この論理的名前空間を利用するためのインタフェースがなければならない。論理的な名前空間は仮想的に一つの空間を作りあげているが、実際には、複数の空間をつなぎあわせているものである。従って、アプリケーションプログラムがこの仮想的な空間を透過的にアクセスできるようなインタフェースが必要である。

本論文では、以下のような方針で、WIDE分散環境上に構造を持った論理的名前空間であるWIDE名前空間を作り、この名前空間をアクセスするためのインタフェースの役割を果たす名前サーバの試作を行なった。

- WIDE分散環境での資源というものを明確にし、その上で各資源をどのようなでもオペレーティングシステムその概念が使えるよう、一般的かつ普遍的に定義し分類していく。
- 名前付けされる資源の性質を考慮して、"WIDE資源の名前構造"を作り、名前をどのようにつけるかを決める。
- ユーザインタフェースとしてのオペレーションの種類、方法を定義し、ユーザの要求を適切に処置できるようにする。

また、このWIDE名前空間上で、名前サーバを利用する応用システムとして、WIDE版のphoneであるwphoneの試作を行なった。そして、このwphoneを通して、今回作成したWIDE名前空間、及び名前サーバの性能の評価を行なった。

2 WIDE名前空間

ここでは、WIDE分散環境上で用いられる名前空間の構造について述べる。WIDE分散環境は複数の組織の集合であるため、各組織の名前空間を包含するWIDE名前空間が必要となる。この名前空間は、ユーザ及び管理者が、効率的かつ容易にアクセスできる構造を持たなければならない。

2.1 名前をつける資源

WIDE分散環境を利用するユーザの最大の目的は、この環境上に分散されている資源にアクセスすることである。ユーザがある特定の資源を利用するためには、その資源を選択しなければならない。そのためには、WIDE分散環境上に置かれている資源一つ一つに名前をつけなければならない。

この名前のついている資源のことを"WIDEのオブジェクト"と呼ぶ(以後、単に"オブジェクト"と呼ぶ)。

WIDE分散環境で利用されるオブジェクトには、大きく分類して以下の3種類がある。以後これを"型"と呼ぶ。

- ドメイン
  管理的・地域的境界で区切られる、ホストや人の集まり。全体空間内では、定義域の役割を果たす。普通は、階層構造をなしている。また、組織・ネットワークとも互いに密接にかかわっている。
- ユーザ
  環境を実際に利用する人間自身。ユーザは送られてきたメイルを受け取り、talkやphoneをかけたりかけられたりする。
- サービス
  ネットワーク上で、クライアントの要求に

[1]本研究は、電気通信基金普及財団昭和63年度研究助成金並びにWIDEプロジェクト共同研究各会社の助成と協力によって行われている。

対して必要な情報を提供する。またはその要求に対して、必要なサービスを提供する。

このようにWIDE分散環境のオブジェクトは、実体のはっきりしたものと、実体のない抽象的なものがある。ユーザ型のオブジェクトは、実際のユーザをさすものであり、これは実体のはっきりしたものである。サービス型のオブジェクトとは、サービスを提供するプロセスとデバイスの組であり、これは実体のない抽象的なものである。例えば、プリンタサービスなどがある。これは、プリンタデーモンとプリンタデバイスからなるオブジェクトである。また、ドメイン型のオブジェクトも、実体のないものである。

各オブジェクトは、以下のような情報を持っている。

- ヘッダ部
- データ部
- サービスエントリ部

ヘッダ部は、オブジェクトを制御するための情報が含まれている。オブジェクトの名前、型、所有者、アクセス権、認証、タイム等の情報が含まれる。

データ部は、各型毎に異なっている。例えば、ドメイン型のオブジェクトは、このデータ部にそのドメインに属しているオブジェクトに関する情報を持っている。オブジェクトに関する情報とは、そのオブジェクトの名前、オブジェクトの型、オブジェクトの位置の組である。

サービスエントリ部とは、そのオブジェクトの提供するサービスのエントリ部名を持つ部分である。例えば、ユーザ型のオブジェクトのサービスエントリ部には、メールサービス、phoneサービス、fingerサービス等のエントリ名を持っている。また、ドメイン型のオブジェクトのサービスエントリ部には、データ部をアクセスして、自分の管理しているオブジェクトの名前を知らせるサービス等のサービスエントリが含まれる。

2.2 階層的な名前構造

WIDE分散環境におけるオブジェクトを統一的な名前で表現し、かつ、管理するために、名前空間に図1のような階層構造(木構造)を持たせる。この木構造に現れる各レベルの名前を名前空間のノードと呼び、特に最下位のノードをリーフノードと呼ぶ。リーフノードを除くすべてのノードは、名前をつける資源のところで定義したドメイン型のオブジェクトの名前しか現れない。リーフノードは3種類の型のオブジェクトすべてが現れる。

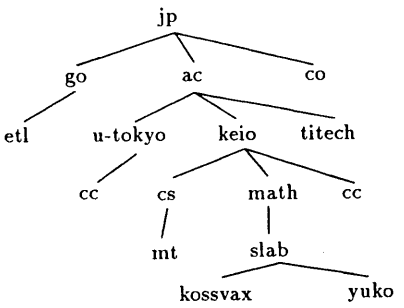


図 1: WIDE名前空間

名前空間に階層構造を導入することによって、WIDE分散環境の膨大な資源の管理を中央で一括して行なうのではなく、各ノード毎に分散することができ、名前の一意性の保証が容易になる。名前空間全体を見渡して一意性を保証する必要はなく、同一の親を持つノードの中でのみ一意性を保証すればよい。これを再帰的に繰り返すことにより、名前空間全体でその名前の一意性が保証される。また、分散管理を行なうことにより、ネットワーク全体の拡張(新しいノードの追加等)に伴う登録や再構成が容易になる。そして、WIDEの環境ではこの拡張により、ノードの数は急激に増えると思われるので、階層構造は固定階層にせず、自由に、必要に応じて拡張できるように可変階層とする。

3 名前表現について

ここでは、WIDE分散環境上の名前表現について述べる。

名前空間内で名前と実体を一意に結び付けようとした時、適切な名前表現が必要となる。名前表現は名前空間内で、ある一つのオブジェクトを識別するために用いられる。そのためには、名前表現が、紛らわしくなく実用的なものである必要がある。

また名前表現が、実体にアクセスしようとする要求者やその要求者との経路などの情報に基づき、変わってくる場合もある。そこで階層的な名前構造をもとに、認識しやすい名前表現を考える。

3.1 既存の名前表現

現在UNIX上では、同一のコマンドでありながら引数の形式をいろいろとれる場合がある。例えば、以下のようなコマンドがある。

```
% rcp user@host:remote-file
local-file

% rlogin host -l user

% talk user@host

% talk host!user
```

このように、コマンドによって引数の形式が異なり、そしてさらに同じコマンドでも、複数の種類の引数が可能となっている。そこで、コマンドにおける名前表現が、統一されていないことが問題点としてあげられる。

3.2 WIDE分散環境における名前表現

WIDE分散環境上の名前空間は、階層的な論理的木構造を持っている。そこで、この構造をそのまま名前表現に用いる。この方法を用いると、名前空間の構造と名前表現のマッピングが容易であり、たいへんシンプルである。

名前の表現形式は、UNIXのファイル名の表現形式と同じである。つまり、WIDE名前空間のドメイン型オブジェクトは、UNIXのファイルシステムでのディレクトリにあたり、それ以外の型のオブジェクトは、UNIXのファイルシステムでのファイルにあたる。WIDE名前空間での最上位ノードは、/jpという表現を用いる。そして、各ノードの区切りには、"/"を用い、リーフノードの後には、この区切りを用いないことにする。例えば、jpというドメインの下のacというドメインは、

```
/jp/ac
```

という名前表現になる。各オブジェクトはそれぞれ型を持っているが、名前表現にはこれは現れない。これもUNIXのファイルシステムの名前表現と同様である。

名前表現の最後に型を示すサフィックスを入れて、

```
/jp/ac/keio/yuko.user
```

とする方法も考えられる。また、リーフノードと、その上のノードとの間に型を表すノードを入れて、

```
/jp/ac/keio/user/yuko
```

とする方法も考えられる。前者の方法は、名前表現に意味を持たせることになる。WIDE名前空間では、各オブジェクト自身が型を持ち、それに名前をつけている。そして、名前表現はあるオブジェクトを指す文字列であり、それ自身が意味を持つものではない。したがって、名前表現に意味を持たせることは、冗長である。後者の方法を用いた場合、名前空間の構造との一貫性が失われてしまう。名前空間の構造をこの表現形式に合わせるとすると、必ず各ノードの下に型を表すノードが必要になる。そして、その下に各型のオブジェクトがくることになる。そうなると、それをそのまま名前表現に用いると、上の例は、

```
/jp/domain/ac/domain/keio/user/yuko
```

となってしまう。名前構造は元のままで名前表現だけを変えると、この間のマッピングが複雑になり、簡潔さがなくなってしまう。したがって、名前表現に型を示す文字列を入れず、

```
/jp/ac/keio/yuko
```

という表現形式を用いることにした。

この表現形式を用いると、各オブジェクトの名前は、あるドメイン、すなわちノードの中で一意性を持てば、名前空間全体で一意性を保証したことになる。但し、ノードの中で一意性を保たなければいけないため、あるノード上に"yuko"というユーザ型のオブジェクトと、"yuko"というホスト型のオブジェクトが共存することはできない。

このように表現することにより、表現された名前が特定の実体を示すと同時に、ある実体を表現する時に使われるアプリケーションなどによらず同じ名前を用いることができる。さらにこの木構造を上位レベルから表現することにより、名前のコンプリーションなども行なうことができる。

4 名前管理機構

WIDE名前空間におかれているオブジェクトには、構造的な名前が付けられている。各オブジェクトの名前は、それが属しているドメインの中で管理される。オブジェクトの名前は、WIDE名前空間の中で一意でなければならないが、階層的な名前空間の構造に合わせて名前付けを行なっているため、各ドメイン内で一意性が保証されれば、その名前がWIDE名前空間内で一意であることになる。

ここでは、このオブジェクトの名前を管理するための、名前管理機構について述べる。

4.1 名前管理機能

WIDE名前空間の名前を管理していくためには、名前の登録、追加、削除の機能が必要になる。

名前の登録、追加は、新しいオブジェクトが増えたことを、これを管理するドメイン型オブジェクトに対して知らせなければならない。これを知らされたドメイン型オブジェクトは、自分のデータ部の中に、新しいオブジェクトのエントリを加える。このエントリは、

(名前、型、位置)

という形式になっている。ドメイン型オブジェクトに、新しいオブジェクトのエントリを追加すると同時に、新しいオブジェクトを一つ作る。

名前の削除は、そのオブジェクト自身を消すと同時に、そのオブジェクトを管理しているドメイン型オブジェクトのデータ部から、そのエントリを消す。

4.2 名前のリンク

WIDE分散環境では、今までと異なり、組織にまたがった遠隔loginなどが行なわれる。つまり、同一のユーザが、keioのホストとu-tokyoのホストに同時にloginするということが起こる。この時、このユーザをどこで管理し、かつどのような方法で同一のユーザであることを保証するかが問題になってくる。

名前は単にオブジェクトを参照するためだけのものであるので、上記の例に関していえば、ユーザに関する操作(メールを送る、phoneをかけるなど)はすべて、オブジェクトに対して、あるいはオブジェクトからの情報によって処理される。

ここまでは名前とオブジェクトとは1対1の対応で考えてきたが、一つのオブジェクトに複数の名前を持たせることによって、オブジェクトのアクセス方法に柔軟生を持たせることができる。例えば、/jp/ac/keioと/jp/ac/u-tokyoの二つの異なるドメイン上のホストを利用できるユーザを他のユーザがアクセスする場合を考える。/jp/ac/keio上のユーザは、そのユーザを/jp/ac/keio上のユーザであると認識する。また、/jp/ac/u-tokyo上のユーザは、そのユーザを/jp/ac/u-tokyo上のユーザであると認識する。ここで、そのユーザに/jp/ac/keio/yukoという一つの名前しかついていないと、/jp/ac/u-tokyoのユーザは、/jp/ac/keio/yukoということを知らなければ、そのユーザをアクセスできなくなってしまう。そこで、そのユーザが/jp/ac/keio/yukoという名前と、/jp/ac/u-tokyo/yukoという名前を持てばよいのである。

この問題に関しては、すべてのオブジェクトに対してその正式名称を定め、さらに別名を与えることによって解決できる。これは、UNIXのシンボリックリンク[8]の機能と同等の機能である。

5 名前サーバ

複数の名前空間にまたがるWIDE名前空間上に置かれたオブジェクトを有効に利用するために、これらのオブジェクトをアクセスする機構が必要となる。この機能を提供するのが名前サーバである。ここでは、WIDE名前空間上に構築したWIDE名前サーバの構造、及び機能について述べる。

5.1 名前サーバの管理する情報

名前サーバとは、WIDE名前空間上に分散されているオブジェクトのアクセス方法を、それを利用するアプリケーションプログラムに提供するものである。そのために、名前サーバは、各オブジェクトのアクセス方法に関する情報を持っている。各オブジェクトは、そのオブジェクトの提供するサービスのための口を持っている。これは、現在のポートのようなものである。ここでは、この口のことを"オブジェクトのアクセスポイント"と呼ぶ。アプリケーションプログラムは、このアクセスポイントにサービスを要求し、サービスを受けるのである。したがって、アプリケーションプログラムは、まずオブジェクトのアクセスポイントを知らなければならない。これを教えるのが名前サーバである。このために、名前サーバは管理しているオブジェクト毎に、

(名前、型、アクセスポイント)

の組を持つ。これが、名前サーバの管理している情報である。

ここでWIDEの環境はIPレベルで通信可能であるため、通信手段としてはWIDE-IPCを定義する。それを用いて名前サーバ同士は互いに通信をおこない情報を交換する。すなわちここでいうオブジェクトのアクセスポイントとは、各オブジェクトが通信するためのIPCポートを意味する。

5.2 名前サーバの構造

WIDE名前空間は、先に述べた通り階層構造を持っている(図1)。また、WIDE名前空間上での名前の管理は、各ドメイン毎に行なう。この構造を有効に用いて名前空間の管理を行なうために、名前サーバは各階層のノード上に置く。したがって、名前サーバも名前空間と同じ階層構造を持つことになる(図2)。

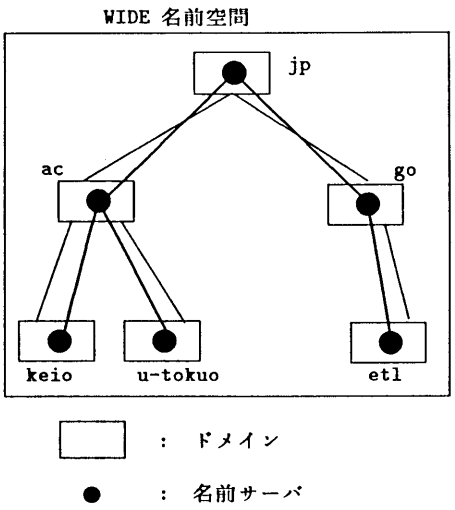


図 2: 名前サーバの構造

各ドメイン上には、少なくとも一つ以上の名前サーバが存在することになる。名前サーバは自分自身が置かれているドメイン内に存在するすべてのオブジェクトに関する情報を持ち、それに関しての責任を負う。オブジェクトに関する情報は、そのオブジェクトの名前と、そのオブジェクトの型、及びそのオブジェクトのアクセスポイントである。それぞれの名前サーバは、クライアントから要求された名前の指すオブジェクトを探し、そのオブジェクトの型とそれへのアクセスポイントを返す。

5.3 名前サーバの検索方法

WIDE分散環境上に置かれた資源を利用するアプリケーションプログラムは、この名前サーバを用いて、目的のオブジェクトへのアクセスポイントを得る。そのために、名前サーバに対して、目的のオブジェクトの名前を渡す。名前を受け取った名前サーバは、その名前を自身で管理している場合、そのオブジェクトへのアクセスポイントを返す。それ以外の場合は、そのオブジェクトを管理している名前サーバにその名前を渡し、そこからオブジェクトへのアクセスポイントを獲得し、それをアプリケーションプログラムに返す。この時、どの名前サーバに依頼するかは、構造的に付けられたオブジェクトの名前から判断することができる。実際には、各名前サーバは、自身の上位の名前サーバと下位の名前サーバの存在しか知らない。したがって、直接あるオブジェクトを管理する名前サーバに要求を出すことはできない。そこで、自分が管理していないオブジェクトを要求された場合、これを上位、または下位の名前サーバに依頼することになる。この要求を受け取った名前サーバも、同じ動作をする。そして、最終的にそのオブジェクトを管理する名前サーバに要求がいく。この名前サーバは、目的のオブジェクトの型とそれへのアクセスポイントを返す。

例えば、図1の名前空間上で、/jp/ac/keio の名前サーバが、アプリケーションプログラムから/jp/ac/keio/mathという名前を要求された場合には、これは自身で管理しているオブジェクトであるので、そのオブジェクトへのアクセスポイントを返す。/jp/ac/u-tokyo/ccという名前を受け取った場合には、上位の/jp/acの名前サーバに依頼する。/jp/acの名前サーバは、目的オブジェクトを管理する名前サーバが、自分の下位の名前サーバであることを判断し、この要求を/jp/ac/u-tokyoのサーバに渡す。/jp/ac/u-tokyoの名前サーバは、目的オブジェクトを自身で管理しているので、目的オブジェクトである/jp/ac/u-tokyo/ccの型とそれへのアクセスポイントを最初に要求を出した/jp/ac/keioの名前サーバに返す。/jp/ac/keioの名前サーバはそれを受け取り、その結果をアプリケーションプログラムに返す。

これは、オブジェクトの名前構造と、名前サーバの構造が、WIDE名前空間の構造の上に構築されているために、容易に判断できるのである。

5.4 名前サーバの情報キャッシュ

名前サーバは、自身の管理しているオブジェクトに関する情報と、自身の上位名前サーバへのアクセスポイントだけを持っている。したがって、先の述べた通り、自身で管理していないオブジェクトについては、上位もしくは下位の名前サーバに検索を依頼することになる。しかし、頻繁にアクセスされるオブジェクトの検索を常にこの方法で行なっていては、検索のオーバヘッドが大きくなる。そこで、一度検索したオブジェクトに関する情報はキャッシュしておき、次回からの要求にはこれを用いる。

6 応用システムの試作

名前サーバを使用する応用システムとしてWIDE版phoneであるwphoneをUNIX 4.3BSD上で試作した。

phoneはUNIXシステムで使用されている、ユーザ間で会話をするためのアプリケーションであり、次のように使用される。

phone *user@host* [*tty*]

phoneを使用すると、ネットワークを介して*host*にloginしている*user*と会話をすることができる.

現在の計算機システムではユーザの名前空間はホスト毎に独立している。そのために、ネットワーク環境でユーザを特定するためにはホストも指定する必要がある。しかし、WIDE環境ではネットワーク全体を定義域とした名前空間を提供するので、ユーザの名前だけでネットワーク上の特定のユーザを示すことができる。

そこでwphoneは次のように使用される。

wphone *username* [*hostname* [*tty*]]

ただし、実際に会話を行なうためには特定のホストの特定の端末を知る必要がある。

従来のphoneの場合には*tty*が省略された場合はユーザが使用している端末をphoneが見つける[2]。同様にwphoneにもユーザが使用しているホスト/端末を見つける機能が必要になる。

6.1 ユーザオブジェクト

先にも触れたように現在の計算機システムではユーザは計算機毎に独立した存在として扱われている。しかし、実際には同一の人間が複数の計算機を使うことはよく行なわれることである。また、さらに組織を越えて、例えば

/jp/ac/keio/yuko

と

/jp/ac/u-tokyo/yuko

が同一の人間であることもある。

そこで、複数の"名前"が同一のサービスを表すことができるようにする。つまり、前述の例では

/jp/ac/keio/yuko

/jp/ac/u-tokyo/yuko

は両者とも同じサービスを示すことで、同一のユーザであることがわかる。

これらは正式名称としての"名前"（例えば/jp/ac/keio/math/slab/yuko）の別名としてとらえることもできる。

ユーザオブジェクトはユーザに関する以下の様な情報を扱う。

- 正式名称("名前")
- MAIL-BOX
- 現在使用しているホスト/端末

また、よく行なわれていることとして、ユーザは同時に複数のホスト/端末にloginすることがある。その場合wphoneは複数のホスト/端末から適切なものを選択するべきである。そのためには例えば、

- idle time
- 実行中のコマンド
- 端末の状態(アイコン化など)

などの情報が必要になる。

しかしこれらの情報は変化が激しく、ホスト側からユーザオブジェクトに登録を行なうのは負荷が大き過ぎると思われる。また、常に必要な情報ではないので、ユーザオブジェクトが常時正確な情報を確保していても、大部分の情報は参照されずに更新されてしまう可能性がある。

そこで、ユーザオブジェクトには情報が必要になった時に情報の更新を積極的に要求する機能が必要になる。

6.2 wphone

wphoneはまず*username*を名前サーバに問い合わせて、*username*で示されるオブジェクトの情報を得る。そして、そのオブジェクトサーバがユーザオブジェクトであることを確認し、サーバから*username*がloginしているホスト/端末の情報を得る。以降はphoneと同様の動作をする。

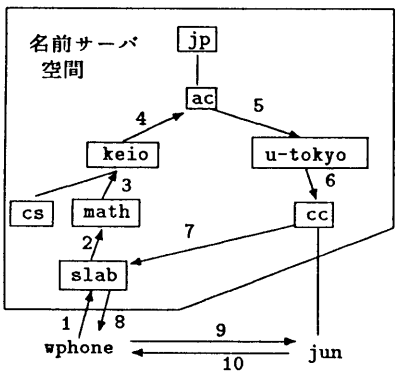
図3は、/jp/ac/keio/math/slab/yukoというユーザがユーザ/jp/ac/u-tokyo/cc/junにwphoneをかけた例である。そしてこの例は、名前サーバが初期状態の時である。次回からの/jp/ac/keio/math/slabの名前サーバに対する/jp/ac/u-tokyo/ccへの要求は、1の次にすぐに/jp/ac/u-tokyo/ccへ要求を出す。そして7ではユーザオブジェクトであるjunの型とアドレスを返し、10でユーザ/jp/ac/u-tokyo/cc/junの持つ情報（login-hostとtty）を返してはじめてwphoneは成立する。

名前空間の概念と、名前空間で提供される各種のサービス、そしてwphoneを使うことでユーザが物理的にどこに存在しているかに関わらず、論理的な"名前"で会話することができる。

7 おわりに

大規模広域分散環境の構築のための実験環境であるWIDE分散環境は、複数の組織の集合であり、各組織は独自の名前空間を持っていて、

[2] utmpを参照するようになっている。



yuko% wphone /jp/ac/u-tokyo/cc/jun

図 3: wphone の例

その中で資源の管理をしている。しかし、その資源管理機構を用いてWIDE分散環境の資源の名前付けをし、それに対して透過的なアクセスを保証することは不可能である。そこで、WIDEの環境で資源の名前付けを行ない、かつ、透過的なアクセスを行なうために、WIDEの環境全体を包含できる論理的な名前空間を階層的な名前構造を用いて定義した。その名前空間の中に現れる資源に、階層構造にしたがった表記方法で名前付けを行ない、それをオブジェクトとした。そして、この名前空間を利用して、オブジェクトにアクセスをするアプリケーションの実例としてWIDE版phoneとしてwphoneを、インタフェースとして名前サーバを試作した。これらはまだ実験段階であり、WIDE-IPCも開発中であり、名前サーバのキャッシュ情報の有効期間、セカンドサーバの存在、サービスの処理の効率、アクセス権、セキュリティ、認証等の問題もある。これらは、WIDE-IPCと密接に関連しており、それによって解決されるべき問題である。また、アプリケーションレベルでのオブジェクトの名前の表記方法の統一という問題も残されている。これらWIDEの環境をより利用し易いものにする資源管理機構の構築が今後の課題である。

謝辞　本研究に関する多くの議論と助言を頂いたWIDE研究会のメンバーに感謝する。

References

[1] 大川 恵子. インターネットネームサーバに関する一考察. Technical Report, 慶應義塾大学大学院理工学研究科電気工学専攻修士課程, February 1985. 昭和 5 9 年度修士論文.

[2] 村井純, 加藤朗, 佐藤智満, 楠本 博之, and 山口 英. 大規模広域分散環境 WIDE の構築. 情報処理学会 マルチメディア通信と分散処理研究会 資料 *DPS-42-8*, May 1989.

[3] 村井純 and 田中啓介. 広域分散環境における資源管理. 情報処理学会 マルチメディア通信と分散処理研究会 報 告 *88-MDP-36-10*, February 1988.

[4] 石井 忠俊. *Phonenet* ネームサーバの設計と実装. Technical Report, 慶應義塾大学理工学部電気工学科, February 1985. 昭和 5 9 年度卒業論文.

[5] *SunOS4.0 Network Programing.* Sun Microsystems,Inc., first edition, May 1988. Chap.1 Network Services.

[6] XEROX CORPORATION. *Clearinghouse Protocol.* XEROX CORPORATION, 1984. Xerox System Integration Standard.

[7] Kevin J.Dunlap. *Name Server Operations Guide for BIND.* University of California Berkeley, April 1986. UNIX System Manager's Manual(SMM).

[8] Marshall Kirk McKusick, William N.Joy, Samuel J.Leffler, and Robert S.Fabry. *A Fast File System for UNIX.* University of California Berkeley, July 1983. UNIX System Manager's Manual(SMM).

[9] The *Hesiod* Name Server. In Stephen P.Dyer, editor, *Usenix Conference Proceedings*, USENIX, February 1988. Winter Conference. Dallas,Texas.