大阪大学基礎工学部生物工学科のネットワークと管理

大阪大学 基礎工学部 生物工学科
大家 隆弘
alex@bpe.osaka-u.ac.jp

1 はじめに

私がいる研究室に初めてのワークステーションがきてからもうすぐ1年，ネットワークを構築しはじめて8ケ月がたとうとしています．この間，いろいろなネットワーク関連の設定と管理をすることがありました．

この記事では，生物工学科ネットワークを例にとって，ネットワークの現状とネットワーク上でのホスト情報の管理を効率良く行なうためのシステムである NIS(Network Information Service) と BIND についての設定や管理にまつわる話をしようと思います．

2 生物工学科ネットワークの構成

まず，生物工学科ネットワークの概略を紹介します．生物工学科のネットワーク構成図を Fig. 1 に示します．今のところ，生物工学科ネットワークにはホストマシンが 10 数台しかなく，ネットワークとしては比較的小規模な部類に入ります．

生物工学科ネットワークの各マシンは単一のイーサーに接続され，ネットワークの構成の仕方としては一番簡単な方法をとっています．また，広域ネットワークとの接続は **aoi**(133.1.60.1) をゲートウェイにし，基礎工学部基幹イーサーに接続することで行なっています．

生物工学科では，ネットワーク上のホスト管理を **aoi**(133.1.60.1) が集中的に行なっています．具体的には，**aoi**(133.1.60.1) は，

- 生物工学科 NIS ドメインのマスターサーバー
- 生物工学科 BIND ドメインのプライマリィ・マスターサーバー

になっています．

3 ネットワークの管理

ネットワークを管理していくうえで一番はじめにやらなければならないことは，ホスト情報の管理です．ネットワークが効率良くまた順調に動作するためには，ネットワーク上にどのようなホストマシンが接続されているかを常時把握しておくことが大切です．

ネットワークに接続されているホストの情報がないとネットワーク間での通信すらできないのですから，このホスト管理は正しくきちんとやらなければなりません．ホスト情報の管理は

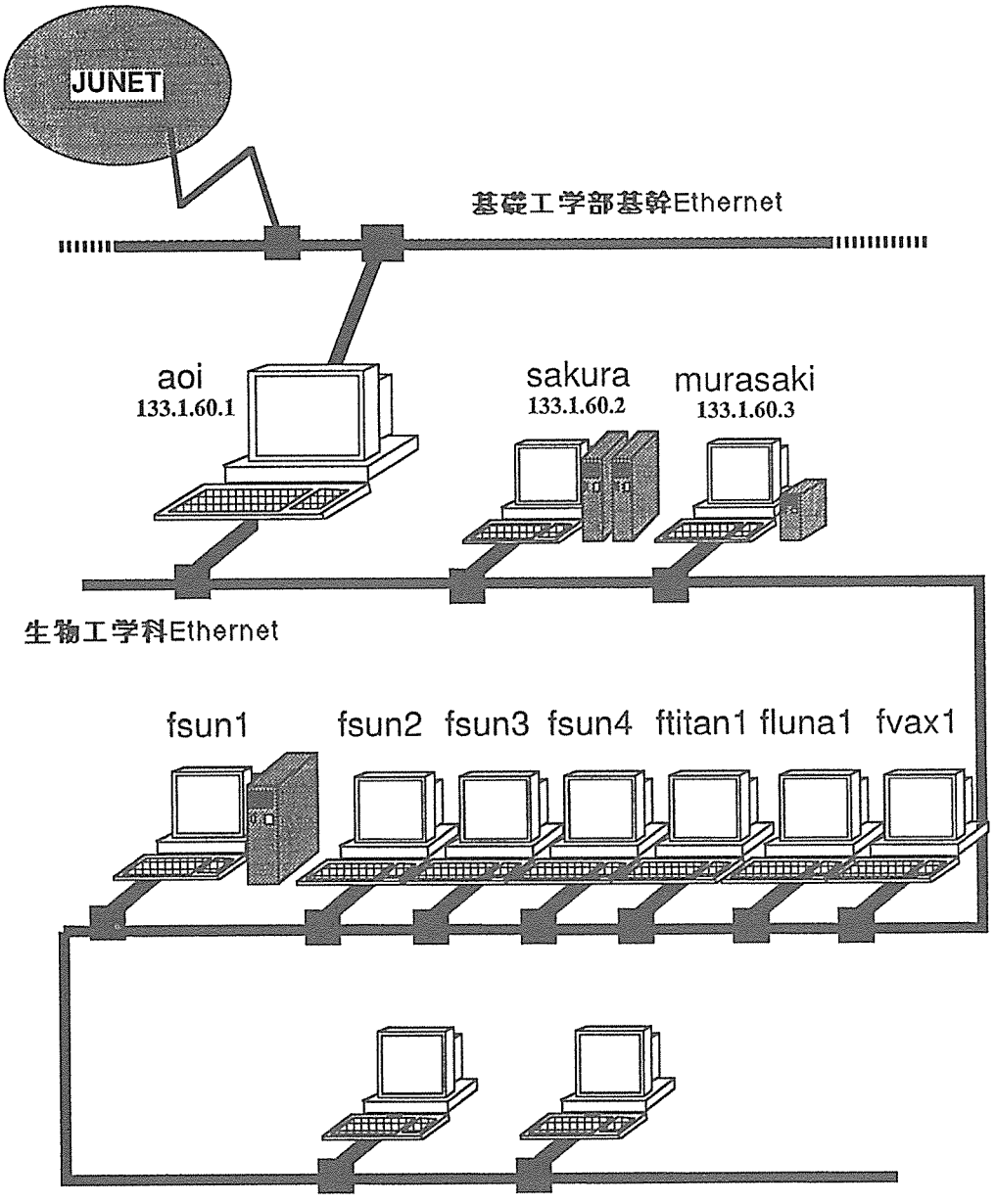

Fig.1 生物工学科のネットワーク

ネットワークが比較的小規模な場合には少ない労力で行なうことができますが，ネットワークが拡大し接続されるホストマシンが増えてくるとホスト情報の管理は非常に複雑で煩わしい作業になります．

ホスト管理を効率よく行なうためにNISとBINDと呼ばれるソフトウェアがあります．生物工学科ドメインにはこの両方がインストールされていますので，このそれぞれのソフトウェアについて生物工学科の例をみながら話を進めていきます．(以下で述べられる技術的な話のある部分は，生物工学科での設定に依存していますのでご注意ください)．

ネットワークはしばしばドメインと呼ばれる単位で区分けされ，ドメインを単位としてネットワークの管理が行なわれます．この意味で，生物工学科ネットワークは1つのドメインと考えることができますので，以下，生物工学科ネットワークを他のネットワークと区別する意味で生物工学科ドメインと呼ぶことにします．

NISとBINDの話をする前にまずネットワーク上でマシンが他のマシンを識別するしくみについて簡単にふれておきましょう．

3.1 ホスト名とIP-アドレス

インターネットワーク上では，マシン同士はIP-アドレスを呼ばれるもので相手のマシンを識別します．このIP-アドレスは4オクテッド(8ビット×4桁)の数値からなるもので，インターネットワーク上で各マシンに固有な値が登録されてきます．接続されているネットワークの全体では，IP-アドレスは各マシンにユニークな値が対応付けられますから，IP-アドレスを指定すればネットワーク上のマシンを特定することができます．

例えば，生物工学科のマシンsakuraには133.1.60.2というIP-アドレスが，murasakiには133.1.60.3というIP-アドレスが設定されており，これと同じIP-アドレスを持つ他のマシンはインターネット上にはありません．このようにIP-アドレスとマシンは1対1対応になっているので，IP-アドレスですべてのマシンを特定することができます．(厳密には，IP-アドレスはネットワーク・インターフェースに登録される数値なので，ゲートウェイなどのネットワーク・インターフェースを2つ以上もつマシンではインターフェースの数だけIP-アドレスが対応付けられます)．

インターネット上で通信されるデータ(パケット)には送信先や送信元の情報がIP-アドレスで書かれていて，インターネット上の各マシンはパケットのIP-アドレス情報をみて交信しています．このため，インターネット上ではIP-アドレスがマシンを識別する手掛かりとなります．

しかし，これと同じようにインターネットワーク上の利用者がIP-アドレスでマシンを識別することができるでしょうか？これは可能で，例えば，マシンのIP-アドレスを知っていれば，

```
% telnet 133.1.60.2
```

```
Trying 133.1.60.2 ...
Connected to 133.1.60.2.
Escape character is '^]'.

SunOS UNIX (sakura)

login:  username
Password:  password
....
```

とすることで，そのマシンに login することができます．まぁ，マシンの IP-アドレスさえ知っていればだいたいのことはできます．

でも，マシンが数十台接続されているネットワークを考えたとき，そのすべてのマシンの IP-アドレスを利用者が覚えていられるでしょうか？ネットワークの管理者ならばある程度はできるかもしれませんが，一般の利用者にとってはまず無理です．なにかもっと覚えやすい名前をマシンにつけておいて，その名前でマシンを識別するほうが便利です．上の例も，

```
% telnet sakura
```

の様に意味のある (?) 名前でマシンを識別するほうが利用者にとってはるかに覚えやすくまた扱いやすいでしょう．(名前でマシンを指定する効用は他にもあるが，ここでは触れません)．

ネットワーク上でマシンを識別するのにホスト名などの名前を用いた時には，まずそのマシン内で名前からインターネット上で使うことのできる IP-アドレスへ変換されます．この変換を行なうため，IP-アドレスとマシン名の対応づけをするデータベースが各々のマシンに用意されています．IP-アドレスからホスト名に変換するのにもこのデータベースが使われます．

UNIX マシンでは，IP-アドレスとホスト名 (マシン名) の対応付けは/etc/hosts というファイルでおこなわれています．

例えば，sakura(133.1.60.2) の/etc/hosts では，

```
# SPARCstation
133.1.60.1 aoi
133.1.60.2 sakura
133.1.60.3 murasaki
# Macintosh-II
133.1.60.7 akashi
```

というようにホスト名とIP-アドレスの対応付けを行なっています.

後述するネームサーバーがインストールされていないネットワークでは, これらのホスト情報のデータベースは (/etc/hosts) はマシン毎に設定する必要があります. またマシン間でこのデータベース (/etc/hosts) のデータは無矛盾であることが必要です.

4 まずは NIS

マシン間でホスト情報データベースの無矛盾性を保つためには, 新しいマシンがネットワークに接続される毎に, そのマシンをすべてのマシンに登録してやらなければなりません.

小規模なネットワークでは一人の管理者がすべてを把握してネットワークの管理をしていくことができないことはありません. また, 新しいマシンをネットワークに接続したときに, 従来のホストに新しいホストを登録するのもそんなに苦にはならないでしょう.

しかし, ホストの数が増えてくるとそうも言っていられません. 例えば, 100 台のマシンが接続されているネットワークに 101 台目のマシンを接続するときには, 100 台のマシンのホスト・データーベースを一つ一つ書き直してやらなければなりません. (それが終わっても, ほかにやらなければならないことが山とあるのです). 個々のマシン毎に設定を行なうなんてことはやってられません.

そこで, ネットワーク上のホスト情報を集中的に管理をするためのシステムとして NIS(Network Information Service) があります. (NIS は以前は YP(Yellow Pages) と呼ばれていましたが, 登録商標の都合上名前が変わって NIS となりました. でも, みんな YP と呼んでいます:-). NIS はホストの管理ためだけでなく, ユーザーの情報やメイルアドレスの設定などネットワーク上で共通な情報や一貫性を保っていなければならない設定などを集中的に管理するために Sun が開発したシステムです.

ここでは NIS で管理されるデータベースの内, ホスト情報データベースに関する部分に注目します.

4.1 NIS の動作

まず, NIS は NIS ドメインと呼ばれる単位で管理さします. NIS ドメインというのは, 同一の NIS データベースで管理されているマシンの集合と思えばよいでしょう. 生物工学科では, 学科全体を 1 つの NIS ドメインにしています.

ネットワーク上に複数の NIS ドメインが存在している場合を考えると, マシンがどの NIS ドメインに属しているのかを指定しておく必要があります. でないと, 間違った NIS データベースを利用して, マシンが (人間も) パニックに陥ってしまいます (これは, 非常に恐ろしい事態になる). マシンがどの NIS ドメインに属しているのかは起動時に domainname コマンドでマシンに登録されます (/etc/rc.local の最初の部分). また, 現在どの NIS ドメインに属しているのかは,

```
% domainname
bpe.osaka-u.ac.jp
%
```

とすればわかります．NISドメインの名前には好きな名前が設定できます．また，もし'noname'という答えが返ってきたら，そのマシンはNISデータベースを使用していないことを意味します．

NISドメイン内のマシンは，

- マスターサーバー
- スレーブサーバー
- クライアント

の3種類に分けられます．マスターサーバーはNISデータベースの管理を行なうマシンです(NISドメイン内に1台のみ存在する)．スレーブサーバーはマスターサーバーの持っているNISデータベースのコピーを持っているマシンです(NISドメイン内に複数台存在可)．スレーブサーバーの数はNISドメインの規模やマスターサーバーの負荷の状況によって調節します．それ以外のマシンはクライアントで，NISデータベースはもちろん，そのコピーすら持っていません．必要に応じて，ホスト情報などNISデータベースで管理されている情報をサーバーに問合せ，サーバーにその情報を教えてもらいます．

ここで重要なのは，NISドメイン内ではNISデータベースの管理はすべてNISのマスターサーバーで行なえるということです．即ち，スレーブサーバーの持っているNISデータベースは，マスターサーバーから自動的にコピーされるか，又は，マスターサーバーから簡単なコマンドですべてのスレーブサーバーに送り付けることができるので，間違いなくマスターマシンのデータベースと同じ内容を保つことができます(無矛盾性)．また，新しいマシンをネットワークに接続したときには，そのマシンでNISを使用する設定を行ない，NISのマスターサーバーでホスト・データベース(/etc/hosts)を書き換えてやって，後は，

```
% cd /var/yp
% make
....
```

とすれば，新しいホストの情報はすべてのマシンで共有することができるのです．

さて，NISの動作を簡単に説明しましょう．NISで重要な役割を担っているプログラムにはypservとypbindの2つのデーモンがあります．ypservはサーバーで起動される(/etc/rc.local)デーモンです．このデーモンはNISドメインのマシンからの色々な問合せに対して自分の管理しているNISデータベースの情報を提供します．ypbindはNISドメイン内のすべてのマ

シンで起動される (/etc/rc.local) デーモンです．このデーモンは起動するとNISドメインのマシンでサーバーとなっているマシンを探し，そのマシンを記憶します．(これは，ypwhichコマンドで見ることができます)．そして，自己のマシン内で発生したいろいろなNISデータベース関連の問合せに応じてサーバーのypservに問合せ，情報を収集する働きをします(サーバークライアント方式)．まとめると，各マシンで走っているデーモンは，

- マスターサーバー (ypserv, ypbind)
- スレーブサーバー (ypserv, ypbind)
- クライアント (ypbind)

となります．

生物工学科のNISマスターサーバーは **aoi**(133.1.60.1) で，スレーブサーバーは存在しません．また，他のすべてのマシンはNISドメイン **bpe.osaka-u.ac.jp** のクライアントとなっています．このとき，NISクライアントである **murasaki**(133.1.60.3) で

```
% telnet sakura
```

を実行すると，

- telnet は gethostbyname() ライブラリィ関数をコールして **sakura** のホスト情報を問合せる．
- gethostbyname() は ypbind に **sakura** のホスト情報を問合せる．
- ypbind はネットワークを介してypservの走っているマシン **aoi** に **sakura** のホスト情報を問合せる．
- **aoi** の ypserv は 自分の持っているホストに関するNISデータベース (hosts.byaddr, hosts.byname) に **sakura** のホスト情報 (**sakura** のIP-アドレスが 133.1.60.2, etc) があるかを調べ，あればそれを **murasaki** の ypbind に通知する．
- **murasaki** の ypbind は gethostbyname() に情報を与える．
- gethostbyname() の戻り値によって telnet は **sakura** のIP-アドレスが 133.1.60.2 であることを知る．

```
Trying 133.1.60.2 ...
Connected to 133.1.60.2.
```

となるわけです．

このようにNISのクライアントのマシンは，NISデータベースで管理されている情報が必要になる度に，サーバーマシンに問合せを行なっています．これは，少々無駄なように思えるかも知れませんが，このおかげで数十台のマシンのホスト情報のデータベースを書き換えなくてよいことを考えれば十分許せるでしょう．

5 BIND

NIS によってネットワーク上のマシンの管理は，1台の NIS マスターサーバーで管理できるようになったのですが，NIS の制限として NIS ドメインの大きさはブロードキャストパケットの届く範囲のネットワークより広くすることができません．(生物工学科ネットワークは class-B subnet でネットワークを構成しているので，生物工学科ドメイン以外のマシンは，NIS ドメイン bpe.osaka-u.ac.jp のクライアントにはなれません)．

それでは，生物工学科ドメイン外のマシンをネットワークを通じてアクセスしたい場合には，生物工学科ドメインの NIS マスターサーバーの aoi のホスト情報・データベースにそのマシンの登録しておいてやらないと，生物工学科ドメイン内からそのマシンをを認識することができません．

例えば，大型計算機センターの ccsun01 に生物工学科内のユーザーがログインしようとしても，aoi にそれが登録してなければ，

```
% telnet ccsun01
ccsun01:  unknown host
telnet>
```

となって，“そんなホスト知らないよ” と通知されるだけです．“それじぁ，生物工学科ドメイン外のすべてのマシンを NIS のマスターサーバーに登録すればいいじゃないか” なんて言わないでくださいね．それは，NIS の必要が生じたのと同じ状況になってしまいます．

この問題に答えるのが，以下に話題にする BIND です．

BIND で用いられている用語は NIS で用いられている用語と重複しているため，よく BIND と NIS の用語を混同してしまいますが，同じ用語でも意味は少しずつ違っています．以下では用語使用の簡便さのため，出てくる用語は BIND のものとし，NIS の用語を使うときにはその旨を明記します．

5.1 Domain Name System

インターネットの世界では，ネットワークが巨大になってくるに従って如何に効率よくネットワークの管理を行なうかが要求されるようになってきました．その回答が，DNS(Domain Name System) です．

DNS では，“インターネット全体をマシンの属する組織に応じてドメイン (domain) とよばれる階層的な名前空間に対応付け，これに基づく管理上の領域 (zone) を設けて各個に資源の管理を行なうようにする” というのが基本精神です．言い換えると，“インターネット上の管理者は自分のドメインの日々の管理さえちゃんとやっていれば，すべてがうまくいくようなネットワークを作りましょう” ということです．

DNS には

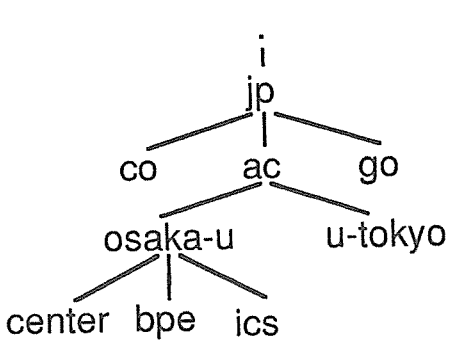

Fig.2 ドメインの名前空間の例

- ドメインの名前空間

の概念があり，また

- 資源レコード，
- ネームサーバー，
- レゾルバ

の３つの部分から構成されています．

5.1.1 ドメインの名前空間

DNS では，ドメインの名前空間の概念が定義されています．この名前空間は木構造をなしており各リーフにはラベルが付けられています．Fig. 2にドメインの名前空間の例を示します．ドメインは１つのノードを意味し，木構造の根 (.) はルートドメインを表しています．Fig. 2の例ではルートドメインの下に jp ドメインがあり，その下に go, ac, co ドメインがあります．さらに ac ドメインの下には，osaka-u, u-tokyo ドメインがあり，osaka-u ドメインの下には，center, ics, bpe ドメインがあります．

これら各々のノードのドメインは階層化されたドメイン名によって識別されます．即ち，Fig. 2の例では，ノード bpe のドメイン名は

```
bpe.osaka-u.ac.jp.
```

ノード center のドメイン名は

```
center.osaka-u.ac.jp.
```

のようにそのノードを含めた上位ノードのラベルを '.' で区切って左から並べた名前になります (右端に '.' があることに注意)．ほかのノードについても，ドメイン名の名前の付け方は同様に行なわれます．

話は変わりますが，ドメインの指定の仕方は2種類あります．1つは今話題にした名前空間でのドメイン名です．もう1つは，IP-アドレスのオクテッドを逆に並べたものに 'in-addr.arpa.' を付けて，

```
60.1.133.in-addr.arpa.
```

のように指定する方法です．

従って，DNS におけるマシンを指定の仕方は，

- sakura.bpe.osaka-u.ac.jp.
- 2.60.1.133.in-addr.arpa.

のように2種類あります．前者はマシンの名前によるもので，後者は IP-アドレスによる指定の仕方です．DNS におけるホスト管理はこの2者の対応を付けることになります．

5.1.2 資源レコード

資源レコードとは，DNS で管理される情報の形式のことです．この資源レコードにはいろいろな属性があり，その中にホスト名の情報や IP-アドレスの情報が含まれています．DNS では，ドメイン内やドメイン間で情報のやり取りに，この資源レコードが用いられます．

5.1.3 ネームサーバー

ネットワーク経由の色々な問合せに対して，自分の持っている資源レコードの情報を提供する機能を持ちます．(NIS の ypserv にあたる)．

5.1.4 レゾルバ

色々な問合せに対して，ネットワーク経由でネームサーバーから情報を得るためのプログラムです．通常，ライブラリ関数の形で提供されます．(NIS の ypbind にあたる)．

5.2 BIND の機能

BIND サーバーのプログラムは，ネームサーバーとレゾルバから構成されています．

ネームサーバーは named (`/usr/etc/in.named`) というデーモンプログラムで，SunOS 4.0.X では，マシンの起動時にデーモンとして常駐プロセスになります (`/etc/rc.local`)．レゾルバはライブラリィの形で提供されています．SunOS 4.0.X に提供されているレゾルバは `/lib/libresolv.a` というライブラリィです．

5.2.1 ネームサーバー

ネームサーバーの種類には，

- マスターサーバー
- キャッシュのみのサーバー
- リモートサーバー

- スレーブサーバー

の4種類があります．(これは，あるBINDドメインからみたときのわけかたで，異なるドメインからみたときには，同じサーバーでも違った種類になります)．

マスターサーバーはそのドメインについてのすべての情報を保守しています．また，このマスターサーバーは，同一ドメイン内に複数個を持つことができます．マスターサーバーには，

- プライマリィ・マスターサーバー
- セコンダリィ・マスターサーバー

の2種類があります．

プライマリィ・マスターサーバーはドメイン内に1つだけ存在し，そのドメイン内のデータを管理しているネームサーバーです．ネットワーク管理者はこのマスターサーバーに対してドメインの情報を設定することになります．プライマリィ・マスターサーバーのほかのマスターサーバーはすべてセコンダリィ・マスターサーバーです．セコンダリィ・マスターサーバーは，定期的にプライマリィ・マスターサーバーのデータベースからドメインの情報をコピーし自己のデータベースを更新します．(この点では，プライマリィ・マスターサーバーとセコンダリィ・マスターサーバーの関係は，NISのマスターサーバーとスレーブサーバーの関係とほぼ同様に考えることができます)．

セコンダリィ・マスターサーバーの数はドメインの大きさやドメイン情報の問合せの頻度によって調節します．通常，ドメイン情報へのアクセスビリティから，セコンダリィ・マスターサーバーを1つ作っておきます．(生物工学科ドメイン内にセコンダリィ・マスターサーバーはありません．理由として，

- ドメイン自体がまだ小さくドメイン内からの情報アクセスが余り多くない．
- 外からのアクセスに関しては，どのみちプライマリィ・マスターサーバーの aoi をアクセスすることになる，
- saturn.center.osaka-u.ac.jp. と vanilla-ice.gw.osaka-u.ac.jp. が生物工学科ドメインを含めた阪大全体のセコンダリィ・マスターサーバーとなっている．

があげられます)．

キャッシュのみのサーバーはオーソリティでなく，ドメイン情報の問合せを受けると，オーソリティをもつ他のサーバーに情報の問合せを行ないます．このサーバーはNISのクライアントに相当します．このサーバーは，オーソリティを持つサーバーから得た情報を一定期間自己のキャッシュに保持し，同一の問合せに対して頻繁にオーソリティを持つサーバーに問合せを発行しないようになっています．(生物工学科ドメインにはありません)．

5.2.2 レゾルバ

レゾルバの基本機能は，問合せをメッセージ形式にしてネームサーバーに送り，その回答をもらうことです．(その問合せの様式はは，NIS 場合と同じようにサーバークライアント形式となっています)．

レゾルバは，ライブラリィに組み込まれた形でインストールされます．ライブラリィ関数として，

- res_mkquery()... 問合せ用のメッセージを作成する．
- res_send()... 問合せメッセージをネームサーバーに送る．
- res_init()... レゾルバを初期化する．
- res_comp()... ドメイン名を圧縮する．
- res_expand()... res_comp() で圧縮されたドメイン名を展開する．

が提供されています．これらのライブラリィ・サブルーチンを使ってプログラムを書けば，BIND の情報を利用するアプリケーションを作ることができます．

SunOS 4.0.X では shared library を用いているので gethostbyname(), gethostbyaddr() などを使ってコーディングされているプログラムは このライブラリィをインストールするだけで BIND を利用することができます．

5.2.3 資源レコードの転送

BIND ではマスターサーバー間でドメインの情報 (資源レコード) を転送する機能を持っています．この転送は，NIS のものとは異なり BIND ドメイン内のサーバーに対してだけでなく，上位/下位のドメインにあるマスターサーバーと行うことができます．この機能が DNS を実現する要となっています．

BIND では原則として，ドメイン内の情報 (資源レコード) はそのドメインのプライマリィ・マスターサーバーで保守・管理します．この資源レコードの情報を階層化されたネットワークで共有するために，マスターサーバー間での資源レコード転送機能が不可欠です．

一般に，ドメインの階層化の中では，情報 (資源レコード) に対して問合せがあったとき，マスターサーバーはその情報が下位ドメインに関するものであれば下位ドメインのマスターサーバーに，そうでなければ上位ドメインのマスターサーバーにその情報の資源レコードの転送要求を発行し目的の資源レコードを得るように設定されます．これは資源レコードを持っているドメインのマスターサーバーに達するまで行なわれます．このような仕組みを階層化されたドメイン全体で作っておけば，各ドメインでプライマリィ・マスターサーバーが保守・管理している資源レコードを他のドメインのマスターサーバーからアクセスすることができます．

通常，ドメインの階層化の中ではあるドメインのマスターサーバーは下位ドメインの管理している資源レコードのコピーを定期的にとり，下位ドメインのすべてのデータを保持するよう

心掛けています．これは，問合せに対してすぐに情報を提供することができるようにするためです．

5.3 BIND の設定

5.3.1 BIND のインストールする前に 1

BIND はすべてのマシンにインストールする必要はなく，ローカルなドメイン内に 1 つないし 2 つのマシンにインストールすればことがたります．他のマシンは，BIND がインストールしてあるマシンに NIS などを通じて情報をもらえばいいのです．

そこでまず，どのマシンに BIND をインストールするかを決めます．

生物工学科ドメインでは，aoi(133.1.60.2) はドメインの NIS マスターサーバーになっています．aoi は生物工学科ドメインのゲートウェイになっているのであまり負荷はかけたくないのですが，ホスト情報や他のいろんな情報を管理しているということで aoi に BIND をインストールすることにしました．

aoi の OS は SunOS 4.0.3 なので，この OS にあった BIND のソースを入手しました．インストールした BIND のバージョンは 4.8.2 です．他の OS についてはよくわかりませんが，BIND を新しくインストールするときにはその旨を

```
inet@center.osaka-u.ac.jp
```

にメイルを出してお伺いをたて，最新の情報を入手したほうがいいと思います．(大阪大学インターネット管理者の皆さんがいろいろと相談にのってくれます．でも，BIND をインストールするような人は，すでにこのメイルリストに登録されているかも知れない:-)

5.3.2 BIND のインストールする前に 2

BIND の設定をするためには，その前にいろいろと決めておかなければならない事柄があります．これらの事柄の中には，他のネットワークとの相互関係をしめすものが含まれていますので，BIND をインストールするときには上位のドメインの管理者又は管理部門に相談するのがよいでしょう．大阪大学内なら，やはり

```
inet@center.osaka-u.ac.jp
```

にメイルを出すか，情報処理教育センターの山口先生に連絡をとって相談するのがよいでしょう．

5.3.3 BIND の設定ファイル

aoi での BIND の設定をみてみましょう．

BIND の設定ファイルには，

- /etc/named.boot
- /etc/resolv.conf

```
;
;       $Header: /etc/ns/RCS/named.boot,v 1.2 90/09/07 14:18:57 alex Exp Locker: alex $
;
directory       /etc/ns
;
; type          domain                  source file/host        backup file
;
cache                 .                                         root.cache
primary         bpe.osaka-u.ac.jp       bpe.zone
primary         60.1.133.in-addr.arpa   133.1.60.rev
primary         0.0.127.in-addr.arpa    localhost
forwarders      133.1.192.4
```

Fig.3 /etc/named.boot

があります. /etc/named.boot は named が起動時に参照するファイルで, /etc/resolv.conf はレゾルバが情報の問合せの際に参照するファイルです.

/etc/named.boot (Fig. 3)

/etc/named.boot には, named が管理する資源レコードのデータファイルが指定されています. (リストの ';' はコメント行を意味しています). まず, directory で資源レコードのデータベースがおかれているディレクトリィ(/etc/ns) が設定されています.

次に, cache によってルートサーバーの情報をネームサーバーのキャッシュにあらかじめ登録します (これはすべてのサーバーで登録しなければならない).

primary の行は, このサーバーが第2フィールドで指定されたドメイン (bpe.osaka-u.ac.jp, 60.1.133.in-addr.arpa, 0.0.127.in-addr.arpa) のプライマリィ・マスターサーバーであることを意味し, 第3フィールドにその資源レコードのデータファイルを指定します.

forwarders の行は, 資源レコードの問合せを依託するサーバーの IP-アドレスを指定します.

/etc/resolv.conf (Fig. 4)

domain で, レゾルバの使用するデファルトのドメイン名 (bpe.osaka-u.ac.jp) を指定します. これは レゾルバに対して sakura のような最後に '.' のついていない名前で問合せが来たときに, レゾルバはここで指定されたドメイン名を使って,

sakura.bpe.osaka-u.ac.jp.

という名前にしてからサーバーに問合せを行ないます.

nameserver は, レゾルバが問合せを行なうサーバーの IP-アドレスを指定します (通常, 3つまで有効). なぜ IP-アドレスなのかは説明するまでもないでしょう.

5.3.4 資源レコード・データベースの記法

BIND のデータファイルでは,

$INCLUDE *filename opt_domain*

```
domain          bpe.osaka-u.ac.jp
nameserver      133.1.60.1
nameserver      133.1.12.7
nameserver      133.1.192.4
```

Fig.4 /etc/resolv.conf

$ORIGIN *domain*

domain opt_ttl opt_class type rr_data

の3つの書式が使えます.

$INCLUDE は, 指定されたデータファイルを読み込む機能です.

$ORIGIN では資源レコードの設定の際のカレントのドメイン名を指定します. 例えば,

```
$ORIGIN                         bpe.osaka-u.ac.jp.
```

としていれば, 次に $ORIGIN が指定されるまで, 相対的に指定された名前 (最後に '.' がついていない名前) はすべてこのドメインの下にあると思います. 即ち,

```
$ORIGIN                         bpe.osaka-u.ac.jp.
murasaki
```

の murasaki は,

```
murasaki.bpe.osaka-u.ac.jp.
```

を表します. また, ドメインの指定に IN-ADDR.ARPA が使われたときには,

```
$ORIGIN                         60.1.133.in-addr.arpa.
```

と指定されているときには,

```
3
```

は,

```
3.60.1.133.in-addr.arpa.
```

を表します.

domain opt_ttl opt_class type rr_data

の書式は具体的に資源レコードを記述するためのものです. *opt_ttl* は, ネームサーバー中でのデータの生存時間を秒単位で指定するフィールドです. 省略すると SOA レコードの最小値が用いられます. *opt_class* は, データのクラスを指定するフィールドです. 現在はサポートされているクラスは, インターネットに関するデータのクラス IN のみです.

type には,

- SOA... ゾーンの開始

  name [*ttl*] [*class*] SOA *origin person* (*serial refresh retry expire minimum*)
- NS... ネームサーバーの指定

  domain [*ttl*] [*class*] NS *server*
- A... アドレスのデータ

  host [*ttl*] [*class*] A *IP-address*
- CNAME... 別名の指定

  nickname [*ttl*] [*class*] CNAME *host*
- HINFO... ホスト情報の記述

  host [*ttl*] [*class*] HINFO *hardware software*
- MX... メイル転送先の指定

  host [*ttl*] [*class*] MX *preference host*
- PTR... ドメイン名のポインタ

  host [*ttl*] [*class*] PTR *name*

などがあります. (このほかにもいろいろあります).

生物工学科のプライマリィ・マスターサーバー **aoi** の **BIND** データベースの設定例をみながら説明をしましょう.

/etc/bpe.zone (Fig. 5)

このファイルは, ドメイン名やホスト名に対していろいろな情報(マシンの**IP-**アドレス, メイルのルーティング, ホストの情報など) を定義しておくファイルです.

まず, 最初の SOA レコードでゾーンの開始を宣言し, 次に, NS レコードでこのドメインで使用するネームサーバーを定義しています. NS レコード で指定されたネームサーバーの内,

```
vanilla-ice.gw.osaka-u.ac.jp.
saturn.center.osaka-u.ac.jp.
```

は, この **aoi.bpe.osaka-u.ac.jp.** のネームサーバーに目的の情報がないときに使用するネームサーバーの指定です.

それ以降の A, MX, CNAME, HINFO レコードは各マシンの情報を記述しています.

/etc/133.1.60.rev (Fig. 6)

このファイルは, **IP-**アドレスからホスト名を得るためのものです.

このファイルでもまず, 最初の SOA レコードでゾーンの開始を宣言し, 次に, NS レコードでこのドメインで使用するネームサーバーを定義しています. それ以降の PTR レコードは **IP-**アドレスに対してホスト名を定義しています.

```
;
; $Header: /etc/ns/RCS/bpe.zone,v 1.4 91/01/12 04:37:38 alex Exp Locker: alex $
;
$ORIGIN osaka-u.ac.jp.
bpe     IN      SOA     aoi.bpe.osaka-u.ac.jp.  root.bpe.osaka-u.ac.jp.  (
                        1.6     ; Serial
                        3600    ; Refresh
                        300     ; Retry
                        3600000 ; Expire
                        3600    ; Minimum
                        )
                IN      NS      aoi.bpe.osaka-u.ac.jp.
                IN      NS      vanilla-ice.gw.osaka-u.ac.jp.
                IN      NS      saturn.center.osaka-u.ac.jp.
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
;
$ORIGIN bpe.osaka-u.ac.jp.
*               IN      MX 1000 aoi.bpe.osaka-u.ac.jp.
aoi             IN      A       133.1.60.1
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation1    SunOS/4.0.3c
bpegw           IN      CNAME   aoi
sakura          IN      A       133.1.60.2
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation370 SunOS/4.0.3
murasaki        IN      A       133.1.60.3
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation1    SunOS/4.0.3c
troll1          IN      A       133.1.60.5
                IN      HINFO   JccXstation      X11R3
troll2          IN      A       133.1.60.6
                IN      HINFO   JccXstation      X11R3
akashi          IN      A       133.1.60.7
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   Macintosh        MacOS
fsun1           IN      A       133.1.60.33
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation370 SunOS/4.0.3
fsun2           IN      A       133.1.60.34
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation330 SunOS/4.0.3
fsun3           IN      A       133.1.60.35
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation1    SunOS/4.0.3c
fluna1          IN      A       133.1.60.36
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
fx1             IN      A       133.1.60.37
fmac1           IN      A       133.1.60.40
                IN      HINFO   Macintosh        MacOS
fvax1           IN      A       133.1.60.41
fsun4           IN      A       133.1.60.42
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
                IN      HINFO   SPARCstation1    SunOS/4.0.3c
ftitan1         IN      A       133.1.60.43
                IN      MX 10   aoi.bpe.osaka-u.ac.jp.
```

Fig.5 /etc/ns/bpe.zone

```
;
; $Header: /etc/ns/RCS/133.1.60.rev,v 1.4 91/01/12 04:37:48 alex Exp Locker: alex $
;
$ORIGIN 1.133.in-addr.arpa.
60      IN      SOA     aoi.bpe.osaka-u.ac.jp.  alex.bpe.osaka-u.ac.jp.  (
                        1.6     ; Serial
                        3600    ; Refresh
                        300     ; Retry
                        3600000 ; Expire
                        3600    ; Minimum
                        )
                IN      NS      aoi.bpe.osaka-u.ac.jp.
                IN      NS      vanilla-ice.gw.osaka-u.ac.jp.
                IN      NS      saturn.center.osaka-u.ac.jp.
;
$ORIGIN 60.1.133.in-addr.arpa.
1       IN      PTR     aoi.bpe.osaka-u.ac.jp.
2       IN      PTR     sakura.bpe.osaka-u.ac.jp.
3       IN      PTR     murasaki.bpe.osaka-u.ac.jp.
5       IN      PTR     troll1.bpe.osaka-u.ac.jp.
6       IN      PTR     troll2.bpe.osaka-u.ac.jp.
7       IN      PTR     akashi.bpe.osaka-u.ac.jp.
33      IN      PTR     fsun1.bpe.osaka-u.ac.jp.
34      IN      PTR     fsun2.bpe.osaka-u.ac.jp.
35      IN      PTR     fsun3.bpe.osaka-u.ac.jp.
36      IN      PTR     fluna1.bpe.osaka-u.ac.jp.
37      IN      PTR     fx1.bpe.osaka-u.ac.jp.
40      IN      PTR     fmac1.bpe.osaka-u.ac.jp.
41      IN      PTR     fvax1.bpe.osaka-u.ac.jp.
42      IN      PTR     fsun4.bpe.osaka-u.ac.jp.
43      IN      PTR     ftitan1.bpe.osaka-u.ac.jp.
```

Fig.6 /etc/ns/133.1.60.rev

/etc/localhost (Fig. 7)

これは ローカルホスト (そのマシン自身) に対する定義が書かれています.

/etc/root.cache (Fig. 8)

最初にネームサーバーのキャッシュにロードされるルートドメイン情報です.

5.4 NIS を BIND に対応させる

さて, BIND で設定した情報を効率よく使いたいものです.

```
;
;       $Header: /etc/ns/RCS/localhost,v 1.2 90/09/07 14:18:57 alex Exp Locker: alex $
;
@               IN      SOA     aoi.bpe.osaka-u.ac.jp.  alex.bpe.osaka-u.ac.jp.  (
                                1.2     ; Sirial
                                3600    ; Refresh
                                300     ; Retry
                                3600000 ; Expire
                                3600    ; Minimum
                                )
;;
1               IN      PTR     localhost.
localhost.      IN      A       127.0.0.1
```

Fig.7 /etc/ns/localhost

```
;
;       $Header: /etc/ns/RCS/root.cache,v 1.2 90/09/07 14:19:00 alex Exp Locker: alex $
;

;
;
; Data file for initial cache data for root domain servers.
.               99999999        IN      NS      kogwy.cc.keio.ac.jp.
                99999999        IN      NS      relay.cc.u-tokyo.ac.jp
;
; Prep the cache (hotwire the addresses).
;

kogwy.cc.keio.ac.jp.     99999999        IN      A       131.113.1.1
relay.cc.u-tokyo.ac.jp.  99999999        IN      A       192.41.197.3
```

Fig.8 /etc/ns/root.cache

BIND を設定するまでは，生物工学科のマシンは NIS で管理されドメイン内のマシンはすべて NIS マスターサーバーの aoi にホスト情報を問合せるようになっていました．ですから，aoi の上で走っている ypserv が BIND のレゾルバにホスト情報を問合せるようになれば，すべてのマシンで BIND で管理されているドメイン情報を共有することができます．

SunOS 4.0.X では，NIS データベースを作成するためのメイクファイル /var/yp/Makefile の中で dbm ファイル hosts.byaddr,hosts.byname を作成している makedbm のオプションとして '-b' を追加して make しなおせば，NIS を BIND に対応させることができ，目的が達せられます．

現在，生物工学科ドメイン内でのクライアントのホスト情報の取得は，

- 生物工学科ドメイン内の情報は NIS ,
- 生物工学科ドメイン外の情報は BIND

を使っています．

5.5 BIND のテスト

BIND にはリモートのマシンの情報をみるためのプログラム nslookup がついています．これを使って，バインドのテストをすることができます．試しに，

```
% nslookup
Default Server:  aoi.bpe.osaka-u.ac.jp
Address:  133.1.60.1
> set type=ANY
> ccsun01.center.osaka-u.ac.jp.
Server:  aoi.bpe.osaka-u.ac.jp
Address:  133.1.60.1
```

```
Non-authoritative answer:
ccsun01.center.osaka-u.ac.jp inet address = 133.1.4.1
Authoritative answers can be found from:
JP-GATE.WIDE.AD.JP inet address = 133.4.1.1
ACRUX.IS.S.U-TOKYO.AC.JP inet address = 133.11.7.185
>^D
%
```

どうやら今日もうまく動いているみたいです.

5.6 BIND のデータベースの更新

さて, BIND ドメイン内に新しいマシンが増えたときには, そのドメインのプライマリィ・マスターサーバーにそれを登録します.

aoi の BIND サーバーに新しいマシンを登録するときには,

- まず, 資源レコードのデータベース・ファイル (/etc/ns/133.1.60.rev,bpe.zone) に新しいマシンを登録します. (このとき, 各々のデータベースファイルの SOA レコードのシリアル番号を増やしておくのを忘れないようにします. こうしておかないと, 他のマスターサーバーに変更を通知することができません).
- 次に, 現在走っているネームサーバーに新しいデータベースをロードさせます.

```
% kill -HUP `cat /etc/named.pid`
```

これで BIND ネームサーバーのデーターベースの更新は終わりです. 数時間もすれば, BIND マスターサーバーを通じて他のドメインのマスターサーバーにその内容が送られ, インターネットワーク全体で新しいマシンを認識することができるようになります.

5.7 BIND の利用方法

BIND を設定したことによってホスト情報のデータベースに定義されていないインターネット上のホストマシンを DNS の名前空間で定義された名前でアクセスすることができるようになりました.

例えは, 生物工学科のユーザーが大型計算機センターの ccsun01 にログインするときには, DNS の名前空間で定義されている ccsun01 の名前 ccsun01.center.osaka-u.ac.jp. を使って,

```
% telnet ccsun01.center.osaka-u.ac.jp.
Trying 133.1.4.1 ...
```

```
Connected to ccsun01.center.osaka-u.ac.jp.
```

とすればよくなりました.

現在, インターネットワークのほとんどのドメインではBINDがインストールされていますから, ドメインの名前空間で定義される名前でほとんどのサイトがアクセスできるようになっています.

6 おわりに

本記事では, 生物工学科ネットワークの現状とネットワークの管理のためのソフトウェアであるNISとBIND紹介をしました.

NISとBINDの説明に生物工学科での設定例を参考にしましたので, 紹介しなかった機能も他に沢山あります. NISとBINDについてもっと詳しく知りたい方は, 1990年7月のUNIX MAGAGINEかNISのBINDマニュアル又は解説書をみてください.