第 14 部

ネットワーク運用技術

第 1 章

概要

ネットワークを安定に運用することは、ユーザにとっては重要な要素であるが、実際に運用する側にとっては大変な作業である。本パートは WIDE Internet の運用を通じて得られた経験を元に、運用のヒントやコツ、失敗例などを収録し、今後のネットワークの運用に携わる人々への参考に供したい。

それぞれの記述は、整理された記述であることは重視せず、たとえ解説が 1 行であってもなるべく多くのトピックスを収録することを目標にした。そのため、管理運用ガイドといった形態になっていないことをお許し頂きたい。また、記述の中には実験的なものも含まれており、安定運用に向けた最終的なマニュアルではないことに特に留意して頂くようお願いしたい。

また、今回の記述に含めることができなかった部分や、新しい経験や失敗に関する情報は、WIDE Technical Note としてまとめていきたい。

なお、記述中 `domain` のようにタイプライタ文字で記された部分はその文字パターンが常に指定されることを表し、*domain* のように斜字で示されているものは例であることを表している。

このメモは WIDE Project の諸氏の協力によって形作られている。さらに、東京大学の高田氏、九州芸工大の藤村氏、東京工業大学の籠谷氏のコメントと協力に感謝する。

第 2 章

Datalink Connections

WIDE Internet では様々な種類の媒体やデータリンクプロトコルが使用されている。キャンパスネットワークとは異なり、広域ネットワークで主役となるのは専用線やパケット交換網、公衆回線などである。これらを IP ネットワークの一部として使う際のいろいろな事項について紹介する。なお、WIDE Internet の接続に使用する専用回線を発注する前には、必ず WIDE Project 電気通信事業者対応担当に相談するようにして欲しい。

2.1 低速専用回線を利用した場合の機器

低速専用回線（9600bps 程度）を利用して遠隔地とのネットワーク接続を行う時、NTT あるいは TTnet のようなキャリアから専用回線を購入する必要がある。一般的には 9600bps の符号品目の回線を使用するが、最近容易に入手できるエラーコレクションモデムを用いて 3.4kHz 音声品目の回線を利用する場合が多い。これは符号品目の回線を利用した場合に比べて若干 RTT が大きくなるが、ファイル転送などを行った際のパフォーマンスが 9600bps 以上得られるからである。モデムの関係で 2 線式の 3.4kHz 音声品目回線を利用している。

現在、表 2.1 のようなモデムが WIDE Internet では用いられている。

表 2.1: WIDE で用いられているモデム一覧

ベンダ	機種	プロトコル	RTT	FTP
Telebit	T-2000	PEP	800ms	1.0kB/s
Telebit	T-2500	V.32/V.42bis	310ms	1.0kB/s
Micrcom	QX/V.32c	V.32 MNP class 10	300ms	1.0kB/s

接続に必要な機材は、モデムとモジュラーケーブル、RS232C ケーブルと SLIP が接続可能な計算機である。WNOC 東京には端末サーバである ANNEX-II に接続し、SLIP ホストとして使用している。

2.2　高速専用回線を利用した場合の機器

64kbps またはそれ以上の速度で接続することが必要になった場合、高速ディジタル専用回線を用いて接続を行なう。

回線は NTT のスーパーディジタルを利用するのが一般的であるが、東京圏ならば TTnet, 関西圏の OMP, 北九州圏の QTnet などのキャリアを利用することもできる。また特に長距離の場合、加入者線の部分は NTT などを使い、長距離部分には日本高速通信、第二電々、日本テレコムなどのサービスを使うこともできる。これらを使う場合はアレンジメントがやや面倒であるが、全面的に NTT を使う場合に比べて安価に運用できる可能性がある。無論 POI との関係によっては経済的でない場合もあるので、注意が必要である。

高速ディジタル専用回線は、日本では NTT 仕様である Y インタフェースと I シリーズ規格に基づく I インタフェースの 2 つがあり、速度によっては両者が選択可能である。I インタフェースを直接収容する機器はまだあまり出回っていないので、WIDE では専ら Y インタフェースを利用している。

Y インタフェースは一部の国産の機器には直接接続することができるが、RS449 や V.35 をインタフェースに持つ多くの機器に接続する場合、変換器（CSU）が必要である。

表 2.2: WIDE で用いられている CSU

ベンダ	機器名	速度	インタフェース
理経	Datashuttle 64	64K	V.35 / RS449
理経	Datashuttle 154	192K ∼ 1.5M	V.35 / RS449
Motorola	6201DTH	64K	V.35

NTT などのキャリアが設置するのは DSU までであって、DSU と CSU を接続するケーブル、CSU、CSU とルータを接続するケーブル等はユーザ側で用意する必要がある。

また、CSU の低速側とルータのポートが同一物理レベルインタフェースであれば問題ないが、異なる場合にはレベル変換器が必要である。レベル変換器には DTE/DCE の方向が固定されているものがあるので、発注の際には十分注意する必要がある。また、無電源タイプも市販されているが、RS232C 側を長くすることができない（30cm 以内が望ましい）ので、物理的な配線の制約がきついことに注意する必要がある。またコネクタの極性にも十分注意されたい。V.35 インタフェースを持つ CSU を RS449 のポートに接続する場合、V.35 DCE/RS449 DTE というケーブルが必要であるが、RS449 側は female であるから短い 37pin の male — male ケーブルが別途必要になる。

2.3 Datalink Protocols

2.3.1 SLIP

SLIP は 1984 年に Rick Adams によって作成されたシンプルなシリアル回線上で IP パケットを交換するプロトコルである。4.3BSD では標準でサポートされ、1988 年にそのプロトコル規格が出版された [9]。SLIP は Internet での標準プロトコルではないが、4.3BSD の普及に伴って広く使われている。

SLIP は `ifconfig` ではなく `slattach` というプログラムによってそのインタフェースをアクティブにする。SLIP はデータリンク層同士でのネゴシエーションなどの高級な機能はないので、相手がダウンしているかどうかはデータリンク層では判断できない。

SunOS 4.0/4.1 の場合、stream 対応のバージョンが入手できるので、それをインストールしなければならない。この場合、`slattach` ではなく `sliplogin` という別なプログラムによってリンクをアクティブにすることに注意されたい。

SLIP が UNIX カーネルにインストールされていない場合、カーネルを再生成する必要がある。細かい手順に関しては SLIP の配布キット中のメモを参照のこと。

SLIP はシリアル回線として 8bit 透過を仮定している。つまりモデムと DTE との間で flow control を行う場合、Xon/Xoff ではなく RTS/CTS による必要があることに注意されたい。

2.3.2 CSLIP

CSLIP は SLIP などのように低速度の回線上で効率良く TCP/IP パケットを交換できるように考案されたプロトコルであり [10]、そのαバージョンがリリースされている。CSLIP は通常 40 octet ある IP header + TCP header を、データリンク層同士に状態を持つことによって、セッション毎に差分を送ることによって数 octet に圧縮してしまう技術である。

CSLIP は通常はインストールされていないので、ユーザ側で作業をしなければならない。このとき、4.3BSD UNIX と SunOS 4.0/4.1 では作業の内容が違うので、配布キットのメモを参照すること。

製品として CSLIP がサポートされている Telebit 社の NetBlazer との接続は確認されていないが、Sony NWS3460 (NewsOS 4.0) と Sun4/260 (SunOS 4.0.3) との間では、一応動作していることが確認されている。`ping` による RTT は CSLIP が ICMP に関しては圧縮の対象にしていないため改善は見られないが、`telnet` などのインタラクティブなコマンドを使用した場合、実用上耐えられるものにまで改善されている。しかしながら、`ftp` などで大きなデータを転送した場合、頻繁に再送が発生し、十分なパフォーマンスが得られていない。この原因については現在調査中である。

2.3.3 Sunlink/IR

64kbps などの高速の同期回線上で IP パケットを交換するソフトウェアで、Sun 上で動作するものとしては SunMicrosystems の Sunlink/IR が知られている。Sunlink/IR は有償であり同社と別途ライセンス契約を結ばなければならない。Sunlink/IR のシリアル側のプロトコルは、Xerox 社の Synchronous Point-to-Point Protocol [8] に準拠しており、SLIP とは異なってデータリンク層同士でネゴシエーションをするため、リンクが不通の時には対応する `if` が “RUNNING” にならない。

Sunlink/IR は `ttya`, `ttyb` などの通常の tty ポートに 19.2kbps 程度の低速の同期回線を接続することができるが、SparcStation-1 を用いている場合、実用上は 64 kbps で通信を行うことが確認されている。

より安定した運用のためには、VME タイプの Sun であれば、MCP あるいは HSI といったシリアル回線専用のインタフェースカードが市販されている。前者は RS449 ポートと RS232C ポートをそれぞれ２個づつ有し、ボード全体として 512kbps 分の回線を収容することができる。後者は RS449 ポート２個が実装されており、それぞれ 1.5Mbps の回線まで収容することができる。それぞれのボードを購入した場合、ボードの基本的なドライバがテープで供給されており、マシンのアーキテクチャに合ったものをカーネルへインストールする必要がある。このテープと Sunlink/IR のテープは別個のものであることに注意されたい。

2.3.4 Proteon PP

Proteon 社の製品である P4100+ または P4200 は、Sun をルータとして流用するのとは異なり、専用ルータとして IP のみならず XNS や AppleTalk などの多くのネットワーク層プロトコルに対応することができる。

P4100+ は４スロット分インタフェースカードを装備することができ、64kbps まで、2Mbps まで、X.25、Ethernet をそれぞれ１ポートずつ収容可能なカードがある。P4200 は８スロット分インタフェースカードを装備することができ、64Kbps を４回線、2Mbps を２回線、X.25 を２回線、Ethernet を１つ収容できるインタフェースがあり、２スロット分を使う FDDI インタフェースもある。現在のソフトウェアの Revision は 9.1 であり、RIP/EGP の他 OSPF-2 を経路情報交換プロトコルとしてサポートしている。

2.3.5 cisco PP

AGS+, AGS, MGS, IGS

2.3.6 PPP

Sunlink/IR、Proteon、cisco などのルータをシリアル回線として用いた場合の問題点は、シリアル回線の両端が同じメーカのルータでなければならないということである。つ

まり、一方が Sunlink/IR を使うとすれば、他方も Sunlink/IR を使う必要があり、ベンダ間の相互互換性が強く望まれていた。

PPP [117] [118] は、TCP/IP 系における Point-to-Point プロトコルの標準であり、これが普及したあかつきにはシリアル回線においても Ethernet のように異なったベンダの製品を混在させて使用することができる。

まだプロトコルが完全に決まったわけではなく流動的な部分も多いが、数箇所で実装されている。製品として PPP をサポートしているのは現在 NetBlazer だけであるが、今後規格が FIX するとともに多くの製品でサポートされるものと予想されている。PPP はプロトコル上は TCP/IP のみならず、OSI/CLNP や XNS など異なったプロトコル群のパケットを取り扱うことが考慮されている。

2.4 IP アドレス

WIDE Internet における IP アドレスの使用方法は、ローカルエリアネットワークのそれと若干異なる場合がある。それらについて述べる。

2.4.1 unnumbered リンク

Ethernet のようなローカルエリアネットワークの場合とは異なり、組織外部へのリンクは二つのルータを対向で用いる、いわゆる Point-to-Point の接続形態を取ることが多い。このようなネットワークは、マルチアクセスネットワークではないが、IP アドレスの割当方針からいえば、一つの独立なネットワークアドレスを割り振り、それぞれのルータのインタフェースに異なったホスト番号を与えるのが基本である。

しかしながら、Point-to-Point リンク一つ一つにネットワークアドレスを割り振ることは、ネットワークアドレスを多く必要とするのみならず、経路制御的なオーバヘッドが大きい。従って WIDE Internet では、特に技術的な問題がない限り Point-to-Point ネットワークアドレスを割り振らない、いわゆる “unnumbered” なリンクとして用いることを原則としている。unnumbered リンクとして問題があるのは、現在のところルータとして SunOS 3.5 を利用している場合に限られている。

unnumbered リンクとして Point-to-Point リンクを用いる場合、それぞれの側の IP アドレスとしては、各々のルータの Primary な Ethernet インタフェースに割り振られた IP アドレスを使用するのが通例となっている。

SLIP や Sunlink/IR の場合 [C]SLIP や Sunlink/IR は、リンクを活性化する際に両端のアドレスを指定する必要がある。リンクに IP アドレスを割り振る場合は、ネットワーク部が共通である一対の IP アドレスを指定すれば良いし、unnumbered リンクとして用いる場合、両端のルータの Primary な Ethernet インタフェースに割り振られた IP アドレスを、ネットワーク部が異なっているのを気にせず、そのまま指定すれば良い。

Proteon の場合　P4100+ や P4200 では、SL/n のインタフェースのアドレスを指定する場合には、process 6 でプロトコルを IP（プロトコル番号は 0）を選択しておき、`add address` コマンドによってインタフェースのアドレスを設定することができる。Point-to-Point リンクを unnumbered にするためには、ここで通常の IP アドレスを指定する代わりに、`0.0.0.`p というアドレスを指定すればよい。ただし、p はそのインタフェースの番号である。

cisco の場合　cisco ルータのアドレスを設定する場合、`configure` モードで `interface` コマンドによってインタフェースのアドレスを設定する。unnumbered として運用する場合、`interface` コマンドによってそのリンクが unnumbered である旨宣言し、ルータ自身から発信する IP パケットの発信アドレスは、適当なインタフェースを指定して、そのインタフェースに割り振られている IP アドレスを用いることを指示すればよい。

ところで、ルータ間で経路情報交換をやりとりする場合、cisco ではこのような設定をした場合、RIP パケットは Point-to-Point アドレスであるにも関わらず、IP パケットの目的アドレスには、アドレスを流用した Ethernet インタフェースに設定されているブロードキャストアドレスが用いられる。従ってこのままの設定では、折角送っている RIP パケットが相手側ルータによって捨てられてしまう問題点がある。これを回避するためには、

> unnumbered リンクのブロードキャストアドレスを、相手ルータの Ethernet インタフェースのブロードキャストアドレスと同じものに設定する

ことが必要になる。

このような場合の `configure` コマンド列の一例を示すと、図 2.1 のようになる。この例では、５行目の設定で Serial 0 が unnumbered であることを宣言し、cisco から Serial 0 に向けて発信される IP パケットの発信元アドレスを Ethernet 0 というインタフェースの IP アドレスを流用することを指示している。そして６行目で Serial 0 のブロードキャストアドレスを相手側の cisco のシリアルインタフェースが流用している Ethernet のブロードキャストアドレスを指定している。

```
interface Ethernet 0                        #1 Ethernet の設定
ip address 133.4.11.5 255.255.255.224       #2 IP アドレスと netmask
ip broadcast-address 133.4.11.31            #3 ブロードキャスト
interface Serial 0                          #4
ip unnumbered Ethernet 0                    #5 Unnumbered
ip broadcast-address 133.229.1.255          #6 相手側のブロードキャスト
bandwidth 64
```

図 2.1: cisco の unnumbered リンクの設定

2.4.2 unnumbered リンクと Subnet

このような IP アドレスの割り当てを行った場合で、Ethernet インタフェース側を RFC950 [14] で規定された subnet として用いている場合には注意が必要である。Point-to-Point リンクのローカル側の IP アドレスは Ethernet インタフェースのそれと同じであるから、同じ netmask を持つ subnet であるように取り扱う必要がある場合がある。これはワークステーションをルータに用い、gated で経路情報交換を行う場合、Point-to-Point リンクの netmask が正しく設定されていないと、gated が正常に動作しない。従って、ptp_start や slattach でリンクを起動した直後、ifconfig によって正しい netmask の設定をするようにしておく。

2.4.3 IP アドレスの使用

WIDE Internet では、133.4.0.0 という class B の IP ネットワークアドレスを、0xfffffffe0 という subnet mask によって subnet に分割して使用している。組織へのリンクに unnumbered ではなくアドレスを割り振る必要がある場合には、その組織で取得したアドレス（またはその subnet アドレス）を使うことを原則とする。

また NOC 設置場所を提供して頂いている組織の場合でも、WIDE Internet のアドレスをその組織のマシンに使わないという原則がある。これは、IP の発信アドレスが WIDE Internet のものになってしまうことを避けるために必要になっている。Ethernet に接続されているルータが専用ルータで直接外部に IP パケットを発信することがない場合やそれに準じる場合にはこの限りではないが、アドレスの運用については WIDE 側担当者に相談されたい。

2.5 インストール

2.5.1 SLIP

SLIP は SunOS4.0 用、SunOS4.1 用などに分かれたパッケージを入手し、その中のファイルを用いてカーネル中に SLIP のドライバをインストールしなければならない。4.3BSD では SLIP は標準でサポートされているので特にカーネルの再構成は不要である。

カーネル再構成の際、修正を必要とするファイルは OS ごとに異なっているので、パッケージ中のメモをよく読む必要がある。SLIP のドライバを適当なディレクトリにコピーする他、conf/files などのファイルの修正が必要であるが、これらもバージョンによって異なる。

SLIP のインタフェースを活性化するためには、slattach あるいは sliplogin というプログラムを、IP アドレスや tty ポート名、回線速度などのパラメータとともに起動しなければならない。特に sliplogin はその標準入力が結合されているポートに対して動作するので、モデムから login して sliplogin を起動する場合以外では、標準入力を当該ポートにリダイレクトしておく。

SLIP で経路情報交換を行う場合には、routed や gated などの経路情報交換プロセスを起動するよりも先にする。またこの段階では、名前サーバが立ち上がっていない場合もあるので、IP アドレスの指定にはシンボリックな形式ではなく、IP アドレスを直接用いるべきである。

2.5.2　CSLIP

CSLIP のインストレーションは SLIP のそれとほとんど同じであるが、4.3BSD でも CSLIP はインストールされていないのでカーネルの再構成が必要になる。手順は SLIP の場合とほぼ同じである。SunOS 4.0 の場合には、SLIP（sliplogin）ソースも必要であるから、予め入手しておく。

CSLIP の slattach には SLIP にはないオプションがある。これらを全部 OFF（-c -e -i）にした場合、ヘッダ圧縮を行わないので、対抗側は普通の SLIP でもよい。

2.5.3　T-2500 の設定

SLIP または CSLIP を運用するとき、モデムとして Telebit 社の TrailBlazer T-2500 を使う場合、T-2500 はいくつかのモデム間プロトコルを選択することができるが、UUCP とは異なり IP では RTT が小さいことも要求されるので、PEP ではなく V.32/V.42bis を用いる方がよい。

T-2500 を専用線に対して使用するためには、S101 レジスタを設定する必要がある。一対のモデムの片方を answer 側として S101 の値を 3 にする。もう一方のモデムは originate 側に設定するために S101 を 4 にする。このようにすると、接続が切れている場合、双方のモデムは 20 秒に一回、モデム間の接続を試みるようになる。

DTE 速度はできれば 19.2kbps（S51=5）がよい。モデム間のデータ伝送速度が 19.2kbps には達していないので、38.4kbps にする必然性はあまりない。SLIP は 8bit 透過であることを仮定しているので、モデム－ DTE 間のフロー制御には Xon/Xoff プロトコルを使うことはできない。そのため、CTS/RTS の制御線を用いるハードウェアフロー制御を使わなければならないので、S58=2, S68=2 に設定しておく。ただし、DTE 側が CTS/RTS 制御線を用いるフロー制御をサポートしていない場合には、多少のパケットロスを覚悟でフロー制御なし（S58=0, S68=0）にするか、パケットロスが気になる場合には DTE 速度を 9600bps に落とすことが必要になる。

また、インタフェース速度を固定（S66=1）しておく方が便利であるし、対向側に管理に関して援助を求める場合には、リモートモデムからローカルモデムのパラメータのアクセスが可能なようにしておくとよい（S48=1）。リンク開設当初でなければ、スピーカを ON にしておくとモデム間リンク切断時に騒々しいので、スピーカボリュームを落としておいた方がよい（S61=0∼10）。

また、gated などの定期的なトラフィックがあるプロセスが走っている場合、モデム接続時に発生するパケットによってハンドシェークが妨害されないようにする。さらに、必

要ならば fall back を禁止し、最高速度での接続ができるようにする。

工場設定時のレジスタ集合で変更が必要なものを表 2.3 にまとめておく。これらのパラメータは、設定した後に電源を落としても変化しないように NVRAM に記録しておくべきである（AT&W）。

表 2.3: T-2500 のレジスタ設定

順番	手順	説明
1	`ats51=5&w`	DTE 速度を 19.2Kbps にして記録
2	ここで DTE 側の速度を 19.2Kbps に変更する	
3	`ats50=6`	V.32 (9600bps) を選択
4	`ats45=255s48=1`	リモートアクセス可、8bit 比較
5	`ats54=3s55=3`	Break と Escape 文字を透過
6	`ats58=2s68=2`	CTS/RTS フロー制御
7	`ats61=0s66=1`	スピーカ off、インタフェース速度一定
8	`ats64=1s94=0`	接続中のデータ無視、fall back 禁止
9	`ats95=2s97=1s106=1`	V.42, MNP の順に接続
10	`ats131=1&w`	CD をキャリア検出にして記録
11	`atn?`	レジスタ群の値を確認
12	`ats101=3&w`	Answer 側の場合
	`ats101=4&w`	Originate 側の場合

2.5.4 Sunlink/IR

Sunlink/IR は SunMicrosystems 社の製品で Sun ワークステーションに対するオプショナルソフトウェアとして販売させている。Sunlink/IR は Xerox の SPP に基づいたデータリンクプロトコルをサポートしており、9.6kbps 以上の同期専用回線上で用いられる。

Sunlink/IR のテープには、zs 用の SDLC ドライバが含まれているので、Sun の TTY ポートに接続する場合、このソフトウェアをインストールすればよい。MCP や HSI などの高速インタフェース上で Sunlink/IR をインストールするためには、それぞれのインタフェースカードに附属しているドライバをインストールし、その後に Sunlink/IR をインストールすればよい。TTY ポートを使用する場合、メーカは 19.2Kbps 程度の速度しか保証していないが、あまり CPU に負荷が掛からない Sun4 ならば 1 回線は 64Kbps を接続しても実用上十分に動作することが確認されている。しかしながらこれはメーカの保証の範囲外であるから、より安定な運用のためには、MCP や HSI[1] を購入することが必要になるだろう。

[1] HSI のドライバの初期のものは正常に動作しないことがあるらしい。

第 3 章

Routing

IP ネットワークを運用する上で経路制御は非常に重要な要素の一つであり、その他のサービスにとってネットワーク層がきちんと動作しており、管理者の意図した範囲で IP reachability が保たれることが、そのネットワークを使う上で大切なことである。ここでは経路制御に関するいくつかの事項について扱う。

3.1 WIDE における Routing

インターネット上の経路制御技術としては多くの研究がなされているが、現在主に利用されているものとしては RIP [119]、EGP [120]、HELLO [121]、BGP [122]、OSPF [109] などがあり、その他研究開発中の多くの経路情報交換プロトコルや経路制御法が発表されている。現在、日本のインターネットは NASA の AS[1] に属しているため、NASA と経路情報交換が可能な IGP (Interior Gateway Protocol) を使用することが必要になる。現在は IGP の中でも UNIX ワークステーションを始めとして多くの機器でサポートされている経路情報交換プロトコルである RIP を使用している。

3.1.1 Default Route

RIP は default route をサポートしている。現在、IP で接続可能なネットワークは、IP Network Address の数で約 2000 もあり、これらのネットワーク一つ一つに対する経路情報を WIDE Internet で配布することはほとんど意味がない。そこで、海外へのリンクを収容しているルータで default route を国内に向けてアナウンスしており、海外のネットワークに対する経路情報はこの default route によって代表するようになっている。

3.1.2 `gated`

ワークステーション上には 4.2BSD UNIX の時代から `routed` という daemon が実装されており、RIP による経路制御に用いられていた。`routed` で提供されている機能は、経路情報を供給するかどうかのスイッチと隣接ゲートウェイを指定する機構だけである。

[1] Autonomous System

広域ネットワークのゲートウェイとして用いるには、指定した相手以外から供給された情報を無視したり、経路情報のアナウンスの制御をすることが必要である場合が多い。

そこで、routed を改良した gated [123] というプログラムが作成され、広く使われている。ワークステーションをルータとして用いる場合、routed ではなく gated を使うことが強く要請されている。gated は経路情報に関する制御ができる他、EGP や HELLO、最近のバージョンでは BGP といった多くの経路情報交換プロトコルをサポートしている。

3.2 経路情報管理上のヒント

3.2.1 Host Route

経路情報は一般にはネットワーク単位、あるいは組織内部ではサブネット単位で取扱いがなされるが、特定のホストに関して経路制御上別個に取り扱う必要がある場合がある。この場合、ホスト部が 0 でない経路情報を交換することによって、そのホストへの経路制御を行うことができる。

ところが、point-to-point ネットワークに独立した IP ネットワークアドレスを割り振った場合、それぞれのゲートウェイが自分自身に割り振られた IP アドレスを host route として広告することが多い。ところがこの場合、同時にその point-to-point ネットワークの net route も広告することが多く、これは無駄である。従って、host route を必要がない場合には広告しないように設定することが望ましい。

3.2.2 Default Route

Default route に関しては、海外のネットワークへの経路情報を表現するように、前述のような取扱いがされている。従って WIDE Internet 参加組織は、経路情報交換に参加している限り、海外向けルータからのホップ数に応じたメトリックを持つ default route を受け取ることになる。また国内のネットワークに関する経路情報は、各ネットワーク毎にやり取りされる。

組織のネットワークが外部へのリンクを WIDE Internet 以外に持たない場合、国内のネットワークに関する経路情報を組織内部に対してアナウンスする必要は特にない。従って、外部ネットワークの経路情報を default route で全て代表させてしまうことが可能である。この場合、その default route をインターネット側に送出しないように特に注意すべきである。

3.2.3 Subnet Default

Default route は経路制御に必要なオーバヘッドを低減する方法として優れているが、組織内部の subnet に対して default route を適用することが問題になることがある。

組織内部のネットワークに障害が発生し、一部の subnet に対する経路情報が到達しないような状況はたびたび発生する。このとき、unreachable になっている subnet にパケットを送った場合、その subnet に対する経路情報が無いため、default route が適用され、組織外部へと運ばれてしまう。

組織外部の最初のゲートウェイでは、subnet に対する認識をしないため、network で経路制御を行うことになり、従ってそのパケットはその組織のゲートウェイへと送り返される。一般に低速である組織外部へのリンクを、パケット一つ毎に TTL が尽きるまで（つまり少なくとも十数回、多ければ百数十回）ピンポンすることになるわけで、他の通信に与える影響は大きい。

このことは、特にネットワークの障害が発生していなくても存在しない subnet へパケットを送ろうとした場合にも発生するので、たとえば組織内部で IP アドレスの変更を行うような場合には要注意である。

解決策としては、subnet に対する default route の適用を組織の外部へのゲートウェイで制限することが考えられている。つまり、subnet default と net default を明確に分離して、上記のようなピンポンが発生しないようにすればよい。この機能はすでに Proteon など一部のルータには実装されている。

3.3 Gated 1.9.1.7

BSD 系 UNIX マシンをルータとして使用する場合、RIP を取り扱うために `routed` というプログラムが標準でサポートされている。しかしながら、`routed` では経路情報に関する種々の制御ができないため、WIDE Internet では `gated` [123] と呼ばれる `routed` を改良したプログラムが使用されている。

`gated` は RIP の他に HELLO、EGP、BGP などの経路情報交換プロトコルもサポートされているが、前述のように国内では RIP を使用しているのでこれらのプロトコルは現在は使われていない。

3.3.1 `gated.conf` の書き方

`gated` は `/etc/gated.conf` というファイルの記述に従って動作するので、このファイルの記述が重要である。

アドレス　アドレスはシンボリックな形式でも差し支えない部分もあるが、`gated` が立ち上がる時には名前からアドレスへの変換機構が十分に稼働していない場合もあるので、すべての IP アドレスは dotted decimal notation（例：133.27.48.2）で記した方がよい。

インタフェース　Ethernet のようなマルチアクセスネットワークの場合は、そのインタフェースに割り当てられたアドレスをによってインタフェースを表現する。Point-to-Point

ネットワークの場合には、numbered, unnumbered を問わず相手側のアドレスによってそのインタフェースを表すことに注意が必要である。

経路情報交換プロトコル RIP を ON にするためには、

```
rip yes
```

を指定しなければならない。また EGP や HELLO は通常使用しないので

```
egp no
hello no
```

を指定しておく。

特定のインタフェースのみ RIP を使用する場合には、`RIP yes` を指定しておいて、`noripoutinterface` や `noripfrominterface` で RIP を送信しない、あるいは受信しないインタフェースを指定する。

RIP の個別対応 Ethernet に接続されている複数のゲートウェイに対して異なった経路情報を与える必要がある場合には、個別に RIP を送る様に

```
RIP pointopoint
```

を指定し、`sourceripgateways` に RIP を送るゲートウェイのアドレスを指定しておく。

3.3.2 trustedripgateways

gated は送られてくる RIP はすべて受信し、処理するが、特定のゲートウェイから送られてきた RIP 以外は無視することができる。この場合、`trustedripgateways` に信頼できるゲートウェイのリストを指定しておく。

`trustedripgateways` を使うことにより、指定したゲートウェイ以外から送られてきた RIP によって経路制御が撹乱されることを防ぐことができる。また特にセキュリティを気にしない場合でも、split holizen をサポートしていない古いバージョンの `routed` から供給される経路情報による routing loop の発生を防ぐことができる。

3.3.3 passiveinterfaces

gated は、アクティブなインタフェースリストを最初に作成し、直接接続されているネットワークに対する経路情報を設定する。そして、RIP の交換を開始する。

ところで、point-to-point インタフェースなどの場合、相手側から RIP が来なくなった場合には、
gated はタイムアウトによってそのインタフェースに関する経路情報を落としてしまう。勿論、相手側から経路情報が再び届いた時点で、これらは復活する。

相手側ゲートウェイがダウンした場合には、ブート完了後、相手ゲートウェイから RIP request パケットが送られてくるので、ダウン以前の状態に復帰することが可能である。しかしながら、リンクがダウンした場合には、両方のゲートウェイがインタフェースをタイムアウトしてしまい、経路情報的にリンクが復旧しなくなる。

このような状況を避けるには、インタフェースを passive にし、タイムアウトしないように指示しておけばよい。

3.3.4　経路情報の受信の制御

gated は、受信した RIP のうち特定のネットワークやホストに関する情報を選択することができる。この場合、対象となるネットワークやホスト毎に、インタフェースやゲートウェイを指定して、受信した経路情報を採用するかボツにするかを指定する。

donotlisten（donotlistenhost）は、対象となるネットワーク（ホスト）に関する経路情報を不採用にすることを指定する。その際、intf all を用いれば、RIP を受信したインタフェースに関わらず不採用になるし、intf 部分にインタフェースを指定すれば、そのインタフェースから来た情報のみを不採用にすることができる。

しばしば用いられるのが、自分の point-to-point インタフェースの host route を無視する設定である：

```
donotlisten my-pp-addr intf all proto rip
```

特定のゲートウェイから来た経路情報のみを採用にする場合には listen を指定する。例えば、

```
listen 0.0.0.0 gateway upflow-gw proto rip
```

は、*upflow-gw* 以外からは default route を受信しないように指示する。

3.3.5　経路情報広報の制御

gated が賞用される一つの理由は、経路情報広報の制御が可能であるということである。WIDE Internet では、内部のネットワークの経路情報の一部を外部に広報しないようにする必要があったり、SLIP を使っている場合に全部の経路情報を交換するのは負荷が大きいため一部に制限したりしなければならない場面が多い。

gated では、受信した経路情報の広報は announce/noannounce によって指定される。ネットワーク単位ではなくホスト単位の経路情報の制御には announcehost/noannouncehost を使うが、意味は全く同じである。

gated では、あるインタフェースに注目した場合、announce と noanounce を同時に指定するできない。announce が一つでも指定されていた場合、announce が明示されていない経路情報はアナウンスされない。noannounce が指定されている場合、指示がない経路情報はアナウンスされない。

インタフェース *if1* には default route のみアナウンスする場合には

```
announce 0.0.0.0 intf if1 proto rip
```

を指定すればよいし、133.4.0.0 も *if1* にアナウンスしたい場合には

```
announce 133.4.0.0 intf if1 proto rip
```

を追加すればよい。一方、インタフェース *if2* には default をアナウンスしない場合には

```
noannounce 0.0.0.0 intf if2 proto rip
```

を指定する。インタフェースが異なっているので、上の３行の記述はエラーにはならない。

このように、gated でのアナウンスはインタフェース単位に考えていった方がやりやすい。つまり、インタフェース１に対してはどの経路情報をアナウンスして、どれをアナウンスしないのかを考え、それを announce/noannounce を使って表現すればよい。

3.3.6 ケーススタディ

SLIP の場合 SLIP は低速であるから、経路情報を全て取り扱うとすると、1991 年 4 月現在で国内のみで 90 を越えており、RIP のオーバヘッドが無視できなくなる。そこで、Internet 側からは default のみ（あるいは major なもの幾つか）を送ってもらうようすることで、RIP の負荷を軽減することができる。

海外へのリンクがダウンしている場合、WIDE Internet 側からは default route は供給されなくなってしまう。そのため、国内の各ネットワークに対してパケットを届けることができなくなる。それを回避する為には、静的に default route を定義しておく。ただし、くれぐれもこの default route に関する RIP が WIDE Internet 側に逆流することのないように処置されたい。

Default を内部のネットワークに対して定義するためには、defaultgateway を宣言する：

```
defaultgateway defgw RIP metric 4 passive
```

active に設定した場合、RIP によって default を変更することができるが、その経路は RIP でアナウンスされない。static の場合には、指定した metric より小さな metric の default route がある場合にのみその経路が採用され、また default が RIP でアナウンスされる。

Default route を設定する場合には、その経路が WIDE Internet 側に流れることのないように、announce/noannounce を用いて設定しておくことが重要である。

また SLIP がダウンした場合に備えて、片側（Internet 側でない方）では、RIP が来なくてもそのインタフェースをタイムアウトしないように passiveinterfaces に指定しておいた方がよいと思われる。

一例として、192.50.89.0 と 133.4.1.1 が SLIP で接続されている場合の、192.50.89.0 側の /etc/gated.conf を図 3.1 に示しておく。この時の 133.4.1.1 側でこの接続に関して必要な記述は図 3.2 のようになっている。

```
#
rip             yes
egp             no
hello           no
traceflags      rip update
#
trustedripgateways      133.4.1.1       # Neighbour Gateway
passiveinterfaces       133.4.1.1       # Do not timeout
# declare default
defaultgateway          133.4.1.1       rip metric 8 passive
#
# only announce its attached network to Internet
#       (DO NOT ANNOUNCE DEFAULT ROUTE)
announce        192.50.89.0       intf 133.4.1.1 proto rip
```

図 3.1: SLIP を使う場合の gated.conf の一例

```
trustedripgateways      192.50.89.10 # SUTGATE.BITNETJP.AD.JP
.....
announce        0.0.0.0           intf 192.50.89.10 proto rip
```

図 3.2: SLIP をサポートする上流ゲートウェイの gated.conf

ANNEX　ANNEX は端末サーバであるが、SLIP を収容するルータとしても使用することができる。WIDE Internet では、東京の WNOC に一台 ANNEX-II を使って 9.6Kbps あるいは 19.2Kbps の回線を数本収容している。

ANNEX の問題点は、経路情報の取り扱いである。ANNEX 自身が経路表を構成する場合、着信した RIP を使用したり、gateways ファイルの設定を参照したりすることができる。gateways ファイルは ANNEX のブートホストの /usr/spool/erpcd/bfs に置かれており、routed のように経路情報の取り扱いを指示する。その一例は図 3.3 のようになっている。この意味は、192.52.188.0 に対する経路を 192.52.188.33 に設定せよということであり、設定された経路は RIP の着信等によっては影響されない。

```
net 192.52.188.0 gateway 192.52.188.33 metric 2 hardwired
```

図 3.3: gateways ファイルの例

ANNEX は RIP を着信する機能はあるが、SLIP で接続された相手方から送られた経路情報を他のゲートウェイに広告することはしない。従って、Internet 側へは別なホストが経路情報を代わりに広告するしなければならない。この別なホストは SLIP が動作しているかどうかを知る機構がインプリメントされていないので、この広告する経路情報は静的にアナウンスされることになる。そのため、例えば `gated.conf` (1.9.1.7) に次のような行が必要になる：

```
net      192.52.188.0    gateway 133.4.3.5 metric 1 rip
```

この場合、133.4.3.5 が ANNEX の IP アドレスであり、192.50.188.0 が ANNEX が収容している SLIP の向こう側のネットワークである。

3.3.7 gated 2.0.1.10

gated の最新バージョンは、1991 年 4 月時点で 2.0.1.10 である。gated はバージョン２になってから大幅に変更された。コンフィギュレーションファイルの文法や記述方法も変更されているので、バージョン１での記述では動作しない。

コンフィギュレーションファイルの記述の主な変更点は次の通りである：

- 各記述はセミコロンで終了する。
- Definition — Protocol — Route — Control の順番に記述しなければならなくなった。
- インタフェースの指定は `ie0` などのインタフェース名でも良くなった。
- 経路情報プロトコルの好み（preference）を指定することができる。
- ICMP redirect の制御が書けるようになった。
- 「何もしない」が default になった。`accept` せよと指定しないと受理しないし、`announce` せよと書かないとアナウンスしない。

また、gated のプログラムとしては、`/etc/gated.conf` 以外のファイル名のコンフィギュレーションファイルでも良くなったし、コンフィギュレーショファイルの文法をチェックすることができるようになった。また、`SIGHUP` によってコンフィギュレーションファイルの再読み込みができるようになったなどの拡張がなされている。

第 4 章

sendmail

広域 IP ネットワークに接続されたホスト間において電子メールの送受信を支援するプロトコルとして、DARPA インターネットサービスの一つである SMTP [124] が用いられている。ここでは 4.3BSD UNIX における SMTP の実装である sendmail についてその設定などについて記述する。

4.1 さまざまな sendmail

現在各ベンダにより各マシンに実装されている sendmail にはさまざまなバージョンが存在することが確認されている。それらを大別すると、(1) Berkeley 系、(2) NFS 系の 2 つに大別することができる。また WIDE プロジェクトで開発された (3) WIDE 版 sendmail もかなり広く用いられているようになってきた。以下それぞれのバージョンの相違による機能および動作の違いについて説明する。

なお、sendmail のバージョンは通常の sendmail.cf を用いている限り、各メールヘッダの Received 行に、

```
Received: from ucbvax.Berkeley.EDU by jp-gate.wide.ad.jp (5.64/2.7W)
        id AA04871; Mon, 18 Mar 91 18:25:11 0900+
```

のように、(sendmail のバージョン/sendmail.cf のバージョン) という形式で表示されるので、それによって判断することもできる。ローカルな sendmail のバージョンは、

```
/usr/lib/sendmail -bt -d21.8 < /dev/null
```

によってもわかる。

4.1.1 Berkeley 系 sendmail

Berkeley 系 sendmail とは、4BSD UNIX のディストリビューションに含まれている sendmail を示す。1991 年 3 月時点でのこの版の sendmail の最新版は 5.65 である。またこの版の sendmail は、U.C.Berkeley が AT&T フリーであると宣言したネットワークコードの中に含まれており、無償で入手することが可能である。以下、現在出回っている主なマシンに実装されている Berkeley 系 sendmail のバージョン、およびその機能の相違について説明する。

4.12 4.2BSD のディストリビューションに付属の sendmail のバージョンである。NEWS-OS 2.x 等、4.2BSD 相当のネットワークコードを持つ UNIX に実装されている。また一部のΣ規格のマシンには、現在もこのバージョンの sendmail が実装されていることが確認されている。この sendmail はかなり古く、さまざまな不都合や障害があることが判明しているため、手元のマシンの sendmail がこのバージョンであった場合、至急新しいバージョンのバイナリをベンダより入手するか、または新しい sendmail のソースを入手、インストールする必要がある。

5.51 4.3BSD のディストリビューションに付属の sendmail のバージョンである。OMRON UniOS-B 等、初期の 4.3BSD 相当のネットワークコードを持つ主な UNIX ワークステーションに実装されている。この sendmail は、以前のバージョンである 4.12 と比較してかなり品質も向上しており、比較的安定に動作するが、debug コマンドというネットワーク上非常に大きなセキュリティホールが存在しているため、新しい sendmail を入手し、入れ換えを行なう必要がある。

5.61 U.C.Berkeley から、4.3BSD Tahoe Release の network free software として 1988 年に公開された sendmail のバージョンがこれである。このバージョンは 5.51 に存在した様々な障害の FIX が行なわれており、またライセンスフリーであるため、近年まで Internet でかなり広く用いられていた。またこのバージョンは Internet 環境でメールを配送する際に非常に重要な要素となる MX レコードをサポートしており、より効率的なメール配送を行なうことが可能である。しかしながら MX lookup を有効にするには、コンパイル時にあらかじめ conf.h で NAMED_BIND フラグを定義しておく必要がある（当然 MX レコードはそれ以降のバージョンにおいてもサポートされている）。

5.64 このバージョンはメールをキューイングする際に、メールファイルの保護のために flock() を用いていることが特徴である。そのため NFS が実装された UNIX でこの版の sendmail を用いる場合、sendmail の起動を inetd および rpc.lockd の起動よりも後で行なうように変更しなければならない。このため例えば SunOS 等では rc.local の修正が必要になる。

5.65 5.65 は 5.64 に対する軽微なバグを修正したものであり、本書記述時点における sendmail の最新バージョンである。

4.1.2 NFSSRC に付属の sendmail

NFSSRC には Berkeley 系 sendmail を元に Sun Microsystems がいくつかの改良および機能の追加を行なった sendmail が含まれている。これは NFS 系 sendmail と呼ばれ、いくつかのバージョンが存在している。またバージョン番号の管理は Berkeley 系とは別個に行なわれており、対応する NFSSRC のバージョンに合わせている。この sendmail

のソースを利用するには、対応する NFSSRC のライセンスが必要になる。以下、主なバージョンの相違について説明する。

3.2 NFSSRC 3.2 に付属の NFS 系 `sendmail` である。SunOS 3.x に標準実装されている。Berkeley 4.12 $+\alpha$ 程度のバージョンを元にしているため、4.0 以降のものにリプレースした方が良い。

4.0 NFSSRC 4.0 に付属の NFS 系 `sendmail` である。SunOS 4.0.x、NEWS-OS 3.x/4.0 等に標準実装されている。このバージョンは当初Berkeley 5.51 $+\alpha$ のバージョンを元にしていたが、上記のセキュリティホールが明らかになった時点で Sun から Berkeley 5.59 に準拠したオフィシャルパッチがリリースされた。これは Sun およびその各ベンダより sun-fixes として入手可能である。またこの 4.0 版から、`sendmail.mx` が付属しているが、オリジナルの SunOS 4.0.x に付属のものにはバグがあり、`sendmail.mx` であるにもかかわらず MX レコードを参照しないという不具合が知られている。これを修正したパッチも前述の sun-fixes に含まれている。

4.1 NFSSRC 4.1 に付属の NFS 系 `sendmail` である。当初の SunOS 4.1 に含まれているものには、ある種の security hole が存在することが知られており、Sun から sun-fixes として Berkeley 5.64 に準拠したオフィシャルパッチがリリースされている。また SunOS 4.1 に付属の `sendmail.mx` には、MX lookup の際の `_res` 構造体のフラグの設定が間違っているため、MX lookup が正常に動作しないというバグがあった。これを修正したパッチも前述の sun-fixes に含まれている。

NFS 系 `sendmail` 4.0 以降のバージョン（NFSSRC 4.0 以降のもの）には、Sun Microsystems により付加されたいくつかの拡張機能が含まれている。以下それについて述べる。

1. `sendmail.cf` 中で定義されるクラスにドメイン名を用いることが可能である。Berkeley 系 `sendmail` では、アドレス解釈の際に “.” がデリミタとして解釈されてしまうため、クラスにドメイン名を用いることが出来ない。

2. `$m` マクロにそのホストが属する NIS ドメイン名が代入される。このドメイン名は Name Server のドメイン名ではないことに注意されたい。また、NFS 系の `sendmail` である NewsOS 4.0 の `sendmail.mx`、および後述の WIDE 版 `sendmail` では、Name Server のドメイン名が代入される。

3. いくつかの NIS 関連の拡張機能をサポートしている。これについては SunOS 等に付属のマニュアルを参照していただきたい。

4. 宛先アドレス別の queue の処理、Message-ID を指定した queue の処理をサポートしている。 Berkeley 系では linear なキューの処理以外はサポートされていない。

5. postmaster へのフェイルメールに本文が入らず、ヘッダだけが送られる。これによってある程度のプライバシーを保護することができる

6. -bt オプションを用いてデバッグをする際に、“^V”, “^W” といった出力の代わりに “$#”, “$@” のように、出力が読みやすくなっている。

4.1.3 WIDE 版 sendmail

国内 の Internet では、WIDE Project で Berkeley 系 sendmail へパッチを加えることにより開発された、“WIDE 版 sendmail” も一部で使用されている。これは前述の NFS 系 sendmail に存在する付加機能を、NIS 関連以外は全てサポートしており、またその他 mail queue の処理、より細かな SMTP のロギング機能等、独自のさまざまな拡張機能を含んでいる。また WIDE 版 sendmail は Berkeley 版 sendmail と完全に上位互換を保つように設計されており、従来の Berkeley 系と同一の sendmail.cf で動作させることが可能である。この WIDE 版 sendmail に新たに付加された機能のコードは NFSSRC とは無関係であるため、このソースコードは NFSSRC 等のライセンスによる制限を受けない。

現在の WIDE 版 sendmail は Berkeley 版 sendmail 5.65 に対してアプライするパッチパッケージとして提供されており、現時点における最新バージョンは 1.5W である。この sendmail のバージョン番号は、「元になった Berkeley 版のバージョン番号 + WIDE 版コードのバージョン」という形式のバージョン番号を出力するように設定されている。また WIDE 版 sendmail は現在も機能拡張作業中であり、将来は MB/MG/MR 等の MX 以外のリソースへの対応等の新機能を付加する予定である。

4.2 sendmail を動作させる際のポイント

ここでは、Internet 環境で Berkeley 系 sendmail を動作させる際、注意すべきいくつかの点について説明する。

4.2.1 コンパイル時の注意

sendmail をコンパイルする際に、conf.h というヘッダファイルでの設定がその sendmail の動きを決定する。以下、コンパイル時に重要な意味を持つ conf.h 中の 2 つのフラグについて説明する。

NAMED_BIND

SMTP によるメール配送の際に Name Server の MX をサポートするか否かを決定するフラグである。これを定義しておくと SMTP によるメール配送の際に MX レコードが用いられる。

NO_WILDCARD_MX

sendmail.cf 中でホスト名をサーチしにいく際に、デフォルトのドメイン名を付加するか否かのフラグと考えてよい。日本の Internet 環境で用いる場合には、このフラグは off (undef) にしなければならない。

conf.h 中では各フラグは

```
#define NAMED_BIND 1
```

のように 1 と定義している。ところが、sendmail のソース内では全てのフラグは#ifdef で参照されているため、各フラグの設定を disable にする場合、0 にするのではなく、該当する行をコメントアウトしなければならない点に注意すべきである。

MX をサポートした sendmail を作成する場合には、<resolv.h> や <arpa/nameser.h>、<netdb.h> 等も忘れずにアップデートしておく必要がある。また勿論 resolver（libresolv.a ）も最新のものに入れ換えておくのが適当である。

バージョン 5.64 以降の sendmail には、4.3BSD Reno の pmake 用の Makefile しか付属していないため、これをコンパイルするためにはバージョン 5.61 の Makefile を修正して使用するのがよい。

ある種の OS では、未解決な名前のためリンクが正常に行なわれない場合がある。その場合は配布キット中の ./support に含まれているライブラリをリンクしてみるとうまくいく場合が多い。この ./support には、4.3BSD Tahoe の libc ライブラリ、および関連するヘッダファイル等が含まれている。これらはいずれもライセンスフリーである。

sendmail がロードアベレージを取得する際に用いている getloadavg() がサポートされていない OS の場合は、X Window の配布キットに含まれている getloadavg.c を用いると良い。このソースはほとんどの UNIX ワークステーションに対応している。またこれを用いる場合、NewsOS のように OS のバージョン、および RISC/CISC アーキテクチャによってカーネル内のロードアベレージの持ち方が異なっている場合があるので設定には十分注意する必要がある。

4.3　sendmail.cf ファイル

ここでは、sendmail の挙動を決定する sendmail.cf ファイルの設定について記述する。

4.3.1　Internet 環境におけるメールの配送

Internet 環境においては、外部組織へ発信されるメールに関してはよほどの特殊な事情がない限り MX レコードを用いた配送が行なわれるべきである。これによって、従来の static な経路設定に比較して経路設定/経路変更が容易になる上、不慮のリンクダウン等に対しての対応が容易になる。また、sendmail.cf ファイルを小型化できるため、sendmail が消費するメモリを節約することもできる。sendmail は頻繁に fork するた

め、案外パフォーマンスに関して有効である。MX に関する詳細は、Name Server の章を参照されたい。

4.3.2 sendmail.fc ファイルについて

フリーズファイル（sendmail.fc）は sendmail が何度も exec （fork ではない）される環境（例:実ユーザの多いホスト、UUCP のゲートウェイホスト等）でない場合、作成しない方が安全である。特に、

```
# sendmail -bt -C./test.cf
```

のように、標準ではない構成ファイルのテストを行なう際に sendmail.fc が存在すると、一部のデータをオリジナルの sendmail.fc から取り込んでしまうという障害があることが報告されている。

4.3.3 MX 対応の sendmail.cf

現在 WIDE をはじめとする国内インターネットで主に用いられている MX lookup に対応した sendmail.cf には大きく分けて次の２つがある。

WIDE 系
: WIDE Project で開発/使用されているもので、もともとは Berkeley の sendmail のディストリビューションに含まれている ucbarpa の sendmail.cf を基に、.JUNET/.BITNET 等のアドレスの処理、およびさまざま改良を重ねて現在に至っているバージョンである。現在広く利用されている WIDE 系 sendmail.cf のバージョンは 2.7W というバージョンである。これは一般には「WIDE 版」と呼ばれている。また現在、WIDE 版および NFS 版 sendmail の拡張機能がサポートされていることを前提にシェイプアップされた新バージョンの sendmail.cf （2.8W-beta）が数箇所で試験的に運用されており、現在の所ほぼ問題なく動作しているようである。なお、NEWS-OS 4.0 では /usr/lib/sendmail.mx.cf にこの 2.8W-beta 版相当の sendmail.cf （SONY-4.0MX）が提供されている。

mailconf 系
: 従来の JUNET 用 sendmail.cf 生成ツールである mailconf に簡単な改造を加え、MX lookup サポートを追加したものである。設定が比較的容易であるのと、従来の環境の延長で使用できるためにこれもかなり広範囲に用いられているようである。この mailconf はこのパッチの作者である九州大学の平原正樹氏の名前から、「平原パッチ」とも呼ばれている。

以下では、WIDE Project で最も一般的に用いられている 2.7W を例に実際の sendmail.cf の設定について説明する。mailconf 版については、mailconf および平原正樹氏のドキュメントを参照されたい。

4.3.4　2.7W の設定

2.7W においては mailconf と異なり、サンプルの sendmail.cf のみが提供されている。それをファイルの説明にあるコメントを頼りに自分のドメインに合わせてカスタマイズしていくことになる。ここでは 2.7W をカスタマイズする際にポイントとなるいくつかの点について以降に説明する。またいずれにせよ 2.7W を用いる場合、sendmail.cf に関するある程度の知識が必要であることは言うまでもない。

1. $w マクロには自分の名前がドメイン形式で代入される 2.7W に限らず、MX 対応の sendmail.cf では本当のホスト名[1] がドメイン形式かそうでないかに関わらず長い名前が代入される。ここでいう長い名前とは、 例えば *jp-gate.wide.ad.jp* のような名前である。このため、例えば *jp-gate!user*（あるいは *user@jp-gate.UUCP*）のような形式のアドレスを扱うためには、あらかじめ予約クラス w に対して、

   ```
   Cwjp-gate
   ```

   という記述を sendmail.cf の最初で追加してやる必要がある。なおこれは、現在の WIDE 版 sendmail では自動的に行なわれているため、明示的にこのような記述を行なう必要はなくなっている。

2. 2.7W では特殊な記述を行なわない場合、“mydomain.JUNET” というアドレスの形式は自らのドメインとして認識されないようになっている。これを認識させるためには、ルールセット 0 の後半部にある、

   ```
   #R$*<@$*ALIAS_NAME>$*        $:$1<@$2REAL_NAME>$3
   ```

   という部分に、例えば

   ```
   R$*<@$*u-tokyo.JUNET>$*         $@$>0$1<@$2u-tokyo.ac.jp>$3
   ```

   というような記述を追加してやる必要がある。

3. “*user@mydomain.jp*” という形式をローカルに処理させたい場合（ドメインマスタにおける設定）、ルールセット 0 にある、

   ```
   # domain that I take it as local         XXXXXXXX
   #R<@$D>:$*       $@$>0$1                 @mydomain:... -> ...
   #R$*<@$D>        $@$>0$1                  ...@mydomain -> ...
   ```

   という行のコメントを外す必要がある。これによりそのホストはそのドメインのドメインマスタとして振舞う。

4. あまり知られていないが、2.7W はローカルな UUCP によるメールの配送も処理することが出来る。そのためには、ruleset 0 にある

   ```
   # domain that connects this host by uucp to gateway mapping XXXXXXXX
   #R$*<@$*DOMAIN_NAME>$* $:<@GATEWAY.UUCP>:$1<@$2DOMAIN_NAME>$3
   ```

   というエントリの後に、

[1] hostname(1) で出力されるもの

```
R$*<@$*dom1.ac.jp>$*    $:<@dom1gw.UUCP>:$1<@$2dom1.ac.jp>$3
R$*<@$*dom1.JUNET>$*    $:<@dom1gw.UUCP>:$1<@$2dom1.JUNET>$3
```

のようにエントリを追加してやればよい。この際上記のように.JUNET に関するエントリも記述しておくのが現在の所望ましいと言えよう。

4.3.5 海外に関する設定

海外に unreachable な IP 接続ホストからの海外向けメールの送信に関しては、適切な該当ホストに static にリレーする形式を取るが、この場合の設定に関しては必ずそのリレーとして指定するホストの管理者、およびそのホストが所属するネットワークコミュニティの承認を得た上で行なわれるべきである。その場合、ruleset 0 の後半部でメイラへ渡す部分を改造することにより行なわれることになる。

4.3.6 _ipforwarding を off にした場合の sendmail.cf の設定

一般的な MX によるメールの配送は全てのホストに IP パケットが届くことを想定している。そのため_ipforwarding を off にした場合、さまざまな細工を sendmail.cf 等の設定ファイルで行なう必要がある。

以下にいくつかの例を示す。

1. 自ドメイン内へのメールの配送には MX を用いない方法。
   static に自ドメイン内の特定ホストへメールを配送するように ruleset 0 を書き換えることで実現することができる。この際注意すべきことは、そのホストにホスト名を記述してしまうと、今度はそのホスト名に対する MX の参照が起こってしまうことである。これは 2.7W では ruleset 0 を書き換える（IP アドレスを直接 sendmail.cf に記述する）ことによって回避できる。また 6.4J では static に設定する（直接 IP アドレスを host.dat に書く）ことで実現することができる。

2. MX を書く際に unreachable な内部のホストを最初に記述してしまう方法。
   最初にトライするホストに、外から届かないホストを書いてしまうという方法が用いられることがある。この場合外部からの MX レコードの参照の際に、最初の送達は必ず失敗することになり、スマートな方法ではない。しかし、そのホストへのアクセスに対して、gateway で ICMP Host Unreach を返すよう設定しておくことによって、ロスタイムが実用上問題にならない程度に抑えることができる。

3. “中向け” の MX を別系統とする方法。
   中向けの送達に際しては、自らのホストで動作している named を参照せず、resolv.conf で指定したある内部の別ホスト上の named を参照する、という方法が用いられることがある。この場合、内部用と外部用のネームサーバを 2 系統管理しなければならないというオーバヘッドがあるが、内部、外部を問わずにすべてのメールの配送を MX によることができる利点がある。

4. SMTP を fake する方法。
   SMTP を内部の特定のホストの sendmail にリレーするデーモンがゲートウェイホストの SMTP のポートをオープンし、SMTP のメッセージをすべて内部のホストに転送し、そこで処理する、という方式である。この方式は NNTP 等にも応用が可能である。

現在の所、「これが一番良い」という方法は今だ確立されておらず、各参加組織がそれぞれの都合に一番合致したさまざまな方法を用いているのが現状である。米国の Internet でもその状況は同様のようである。

これらの非標準の設定を行なう際には、いずれも設定を誤った場合、sendmail が自分自身に SMTP してしまう "Local Configuration Error" による mail fail が起こりやすいため、いずれの場合も設定には細心の注意が必要である。

4.3.7　その他のメモ

- I オプション
  5.61 以降の MX 対応の sendmail に対して有効であり、Named と通信できなかった（ECONNREFUSED）際に、そのメールを temporary fail にするためのフラグである。WIDE 版 sendmail では指定する必要はないが、その他のバージョンの sendmail を使用している場合、sendmail.cf で OI を定義しておいた方がよい。

第 5 章

Name Server

Domain [18] [19] は階層的な名前空間を提供し、それに対する分散的な名前サーバを提供しているものである。Internet においては、ホスト名や IP アドレス、メールサーバなどの情報が Name Server によって管理・運用されている。Domain の詳細やその運用について [125] [126] は参考文献を参照して頂きたいが、ここでは 4.3BSD における Name Server の実装である BIND についてその設定などについて補足する。

なお、現在の BIND のバージョンは 4.8.3 である。このバージョンは Zone データの転送などで 4.8 以前のものに比べて大きく改善されているので、手元の BIND が古い場合、最新版を入手しそれをインストールすることが望ましい。SunOS 4.1 の `named` は 4.8.2 相当とのことであるが、SunOS 4.1.1 では 4.8.3 相当のものになっている。NewsOS 4.0 の `named` は 4.8.3 相当である。

5.1 Zone データ作成上のヒント

5.1.1 ファイルの位置

Named の各ファイルは全部を一つの directory にまとめておくと管理が楽である。例えば `/etc/ns` というディレクトリに集めておくことにする場合、まず `named` が立ち上がるときに読み込むファイル `/etc/named.boot` をこのディレクトリに移す。そして、`/etc/named.boot` という名前でそのファイルがアクセスできるように Symbolic Link を張っておく。

次に、`named.boot` の先頭で、それ以降のファイルアクセスのディレクトリを次のように指定しておく：

```
directory   /etc/ns
```

Primary な Zone データは通常それほど大きくならないので、root ファイルシステムでも差し支えないが、きちんとバックアップが取られるファイルシステムに保存するべきである。一方 secondary な Zone データは場合によってはかなり（2MB 程度）大きくなるので、それを保存するディレクトリを `bak` とした場合、これは `/var` のようなファイルシステムにおいた方がよい。Secondary として Zone データのバックアップは `named.boot` の `secondary` の行の最後のフィールドに指定されるが、そこに

bak/jp.zone

などとすればよい。`directory` として */etc/ns* が指定されている場合、

/etc/ns/bak/jp.zone

というファイルが使用される。

5.1.2 `named.boot` ファイル

BIND Name Server の挙動を決定するのが `named.boot` である。このファイルには、前述のように関連ファイルのディレクトリを記す `directory` 行を初めとしていくつかの制御情報を書くことができ、それによって Name Server の運用が決定される。

`forwarders` 行
: あるサーバが問い合わせを処理する場合、サーバで保持しているデータで解決できなかった場合、Root Name Server に対して問い合わせを行うか、Root Name Server に対して問い合わせを行うような指示を返答する。この場合、Root Name Server ではないが、ある程度の情報を収集している Name Server がある場合、それに対して次の問い合わせを行う方が Root Name Server の負荷やネットワークトラフィックなどの点で望ましい。

  このような次善のサーバを指定するのが `forwarders` 行である。Name Server で保持している cache データのみでは質問に対応することができないとき、通常は `root.cache` に指定されている Root Name Server に問い合わせを行うが、`forwarders` が指定されている場合、まずここで指定されているサーバに対して recursive に問い合わせを行う。

  `forwarders` には複数のサーバを指定することができるので、最寄りのサーバからアクセスが順に行われるように順番を考慮すべきである。そして `forwarders` に指定されたサーバすべてから回答が得られなかった場合には `root.cache` のサーバに対して問い合わせを行う。

`slave` 行
: `slave` を指定したサーバは、問い合わせは `forwarders` に指定したマシンに対してしか行わない。そのため、直接外部のサーバに問い合わせを行うことがない。

5.1.3 Zone データの Serial No.

Name Server 間では、各々の Zone データがすでに更新されているかどうかなどの判断を、その SOA レコード中に記された Serial No. を比較することによって行う。この Serial は “*234*” のような数字または “*3.26*” のように小数点を一つ伴う数字のいずれかである必要がある。

WIDE の各参加組織で多く用いられている方法は、RCS と併用し、RCS の Revision 番号と Serial No. を一緒にするという方法である。check-in する度に Serial No. を書き換えることが必要になるが、Serial No. が変わっておらず更新されたデータが伝搬しないというトラブルを防ぐことができる。

5.1.4 絶対名と相対名

BIND で設定が必要なファイルで、RR (Resource Record) を書くものに関しては、名前の末尾の “.” が重要な役割を担っている。例えば、

```
wide.ad.jp
```

と

```
wide.ad.jp.
```

は異なった意味を持つ。従ってゾーンデータ作成の際には十分に名前の末尾の “.” の有無に注意する必要がある。

後者のように “.” がある場合はその字面通りの解釈がなされるが、前者のように名前の末尾に “.” がない場合、これは相対的な名前として解釈され、そのデータの origin が “*keio.ac.jp*” である場合、実際には “*wide.ad.jp.keio.ac.jp*” という名前であるものと解釈される。つまり current origin が省略されたものとして取り扱われる。

Current origin は $ORIGIN によって変更することができるが、一般には `/etc/named.boot` でそのゾーンファイルを引用するときのドメイン名

`primary` *domainname* *filename*

の第２フィールドで決定される。

末尾の “.” を忘れやすい場所としては、SOA レコードの管理者の Email アドレスの末尾などがある。`in-addr.arpa` ドメインの PTR レコードの RR データ部には通常 “.” が必要である。

5.1.5 localhost definition

ある OS のあるソフトウェアは、“localhost” に対する IP アドレスを求める際、RES_DEFNAMES を立てているらしい。そのために resolver で指定された[1]に指定された default ドメインを付加し、

`localhost.`*default-domain*.

に対する lookup を行なってしまう。このような場合にも対応するため、全てのドメインについて

```
localhost  IN  CNAME  localhost.
```

というエントリを入れておくと良い。

[1] `/etc/resolv.conf` の `domain` 行

5.2 Server のクラス

5.2.1 Primary Server

Primary Server はある Zone データのマスタであり、更新に関する責任を追っている。named がある Zone "*wide.ad.jp*" の Primary Server であり、その Zone 情報が "*zone/wide.ad.jp*" というファイルに格納されているとしよう。このとき、named.boot ファイルで次のように指定する：

```
primary   wide.ad.jp   zone/wide.ad.jp
```

5.2.2 Secondary Server

Secondary Server は Primary Server で設定された Zone データを cache し、その Zone に関しては Primary Server と同等の能力を持つものである。Secondary Server になるためには、named.boot ファイルで次のように指定しなければならない。

```
secondary   wide.ad.jp   133.4.1.1   bak/wide.ad.jp
```

なお、３番目のフィールドはその Zone データをどのホストから転送するかということを示しており、通常はその Zone データの Primary Server を指定する。

ところで、Secondary Server には、ある Zone データの内容を（勝手に）引用している Unauthlized Secondary Server と、きちんと引用している Autholized Secondary Server との２種類を考えることができる。勝手に引用した場合、たまたま問い合わせがその Unautholized Secondary Server に到達した場合、cache したデータが使われるのみである。ローカルな問い合わせなどが多い場合に使われる。その Zone の Authority はどのホストで Zone データの複製が保持されているかを管理していない点が特徴である。

一方 Autholized Secondary Server は、その Zone データ中にそのサーバが NS レコードによって指定されているものである。従って、Primary Server がダウンした場合にはこのサーバが Primary Server の代わりをすることになる。Autholized Secondary Server はその Zone の管理者が戦略的に配置するものであって、root から NS レコードを辿ることによってその存在を知ることができる。

5.2.3 Caching-only Server

各組織の primary な、あるいは secondary なサーバ以外のサーバに関しては、caching only server として設定するのが良い。そのためには、forwarders としてその組織の primary / secondary なサーバを指定する。他の組織のサーバを forwarders として指定するのはあまり勧められない。やむを得ない場合でも、 forwarders の第３位以降に設定するべきであろう。

Caching-only Server には必要に応じて local に問い合わせを解決できるように、ドメイン内部の Zone データを保持するように設定しておくことが望ましい。

5.3 日本の Name Server の系列

Name Server は US の Internet を含めて世界的に一つの体系で運用されている。このための条件は

IP パケットが相互に届くこと

であり、海外の Internet に IP reachable でない場合にはこれとは異なった設定が必要になる。

日本の Internet では、Name Server を次のように３つのクラスに分類することによって、この状況下で必要な範囲の情報を検索できるように対応している。

系列 A 国内の情報で、国外から reachable な部分のみを保持する。

系列 B 国内で reachable な情報を保持する。

系列 C 国内部分は系列 B の情報を用い、海外部分は適切なサーバへのアクセスを行なう。

系列 A の Name Server は海外からアクセスできる情報のみを取扱い、海外から日本への問い合わせに答える役割を担っている。このサーバは Root Name Server に `jp.` のサーバとして登録されており、現在このサーバが走っているホストは

```
jp-gate.wide.ad.jp   133.4.1.1
ns.tisn.ad.jp        133.11.11.12
```

がある。この系列のサーバは、通常海外のホストからの問合せのみを処理し、国内一般のサーバからアクセスされることはない。

系列 B の Name Server は国内のデータのみを管理しており、海外のデータは全く情報を保持していない。現在、系列 B の Name Server のマスタは

```
ccut.cc.u-tokyo.ac.jp   192.41.197.4
```

が担当している。

系列 C は国内の海外にアクセスできるところで運用される。この系列のサーバは、海外の部分は海外のサーバから情報を得、国内部分は系列 B のサーバから得ることによって、IP で接続されるすべてのホスト等に関する情報を lookup することができる。

このような複雑な状況がでてくるのは、国内のすべての組織から海外にアクセスすることができないということにある。そのため、Name Server を立ち上げる場合には、IP reachability によってどのクラスのサーバを立ち上げるべきであるか、あるいはどのクラスの Primary Server にそのドメインの情報を登録してもらうかということが異なってくることに注意しなければならない。

各組織の primary・secondary 以外のサーバは caching-only server として設定するのが妥当である。

5.3.1　海外から reachable な場合

root.cache　海外に対して IP が届く場合の設定は比較的簡単である。つまり BIND の配布キットに付属している root.cache をそのままインストールすればよい。ただし 1990 年 4 月に Root Domain Server のホストが変更されているので、古い配布キットに含まれている root.cache を使う場合には注意されたい。現在の root.cache を表 5.1 に示す。

表 5.1: 海外到達可能なホスト用の root.cache

```
;
; Initial cache data for root domain server.
;
.                       99999999  IN  NS  ns.nasa.gov.
                        99999999  IN  NS  ns.nic.ddn.mil.
                        99999999  IN  NS  aos.brl.mil.
                        99999999  IN  NS  a.isi.edu.
                        99999999  IN  NS  gunter-adam.af.mil.
                        99999999  IN  NS  c.nyser.net.
                        99999999  IN  NS  terp.umd.edu.
;
; order does not matter.
;
ns.nic.ddn.mil.         99999999  IN  A   192.67.67.53
aos.brl.mil.            99999999  IN  A   128.20.1.2
                        99999999  IN  A   192.5.25.82
a.isi.edu.              99999999  IN  A   26.3.0.103
                        99999999  IN  A   128.9.0.107
gunter-adm.af.mil.      99999999  IN  A   26.1.0.13
c.nyser.net.            99999999  IN  A   192.33.4.12
terp.umd.edu.           99999999  IN  A   128.8.10.90
                        99999999  IN  A   192.52.195.10
```

jp ドメイン　各組織では、少なくとも 1 つ jp. とその直下の subdomain である ac.jp., ad.jp., co.jp., go.jp., or.jp. の (unautholized) secondary server を持つべきである。また、in-addr.arpa. ドメインに関しても jp. と同様である。この場合、jp. の各 Zone データは最寄りの autholized secondary server から取り寄せるように設定する。このようにすることによって、組織を跨る問い合わせを減らすことができる。1991 年 3 月時点での jp. の各 Zone データの autholized secondary server を表 5.2 に示す。

Named を走らせる程十分な仮想記憶空間が確保できないなどの理由で、上記のような設定ができない場合には、slave にして、forwarders として表 5.2 を指定すべきである。そうしない場合、root.cache によって Root Name Server に問い合わせを行ってし

まう。jp. の検索を行っていた場合、結局系列 A の海外向けサーバに問い合わせを行うことになり、国内向けの情報を得られないことがある。

表 5.2: 国内の jp. の各ドメインのサーバ

地区	ホスト名	IP アドレス
マスタ	ccut.cc.u-tokyo.ac.jp	192.41.197.4
東京	relay.cc.u-tokyo.ac.jp	192.41.197.3
TISN	utsun.s.u-tokyo.ac.jp	133.11.11.11
藤沢	endo.wide.sfc.keio.ac.jp	133.4.11.2
		133.27.48.2
大阪	vanilla-ice.gw.osaka-u.ac.jp	133.1.192.4
福岡	nakasu.csce.kyushu-u.ac.jp	133.5.19.3

5.3.2 海外から reachable でない場合

海外にアクセスできない組織の場合、root.cache 等に海外のサーバを指定することはできないので、それに代わる設定をする必要がある。1991 年 3 月時点での国内向けのサーバのマスタは表 5.3 に示すとおりであるから、BIND の配布キットに含まれている root.cache を変更しておく。

また、トラフィック等を軽減するため、組織で 1 つは jp. 等の Zone データを cache しているサーバを設定しておくべきである。このサーバは表 5.2 に示すサーバの中で適当なものから Zone データを得るように設定する。

ただし in-addr.arpa. Zone に関しては、表 5.2 のサーバは海外の情報も含んでおり、その Zone データには沢山の海外を指している NS レコードがあるので、国内向けサーバとしては適当でない。従って in-addr.arpa. Zone に限り ccut.cc.u-tokyo.ac.jp からデータを得るようにしなければならない。

またこのサーバでは、海外に対する情報を国内向けのものとして正しく取り扱う必要があるので、表 5.2 のサーバを forwarders として指定してはならないことに注意する。

また、系列 B のサーバは海外のドメインに関する情報は全く含まれていないので、.edu などへメールを送る際には Name Server の MX を使うことはできない。sendmail.cf などで直接海外向けゲートウェイを指定する必要がある。

表 5.3: 海外到達不能なホスト用の root.cache

```
.                       99999999  IN  NS  ccut.cc.u-tokyo.ac.jp.
ccut.cc.u-tokyo.ac.jp. 99999999  IN  A   192.41.197.4
```

5.4　MX レコード

MX は電子メールの交換を行なう場合に、どのマシンに送達すればよいかという示唆を与えるために運用されているリソースである。

some.domain.　　IN　MX　*20*　*ms.some.domain.*
ms.some.domain.　IN　A　　*ppp.qqq.rrr.sss*

という記述があった場合、

user@*some.domain*

宛にメールを出す場合、“*some.domain.*” という名前を持つリソースで type が MX であるものを検索する。そして、そのデータ部に指定されている *ms.some.domain.* というマシンがそのアドレスのメールを処理できることが分かるので、そのマシンに対して SMTP [17] でメールを送ることができる。

この時、複数の MX が指定されていた場合、最も優先順位の高い（優先順位の数字の小さい）メールサーバから順にメールの送達が試行されるので、複数のメールサーバを指定しておくことが好ましい。

5.4.1　MX ホスト

MX で指定されているホストは、外部に対して IP reachable でなければならない。少なくとも国内での IP reachability は確保されていなければならないし、系列 A のサーバに登録されているドメインとそのサブドメインに関する MX は海外も含めて IP reachable でなければならない。従って、そのホストは経路情報交換に参加するか（着信のみでもよい）、default を静的に設定しておく必要がある。

組織として一部のホストしか IP reachability が確保されていない場合、それらのホストにも MX を定義しておくことが大切である。この場合、組織内部向けと外部向けと異なった系列のサーバを用意することは可能であるが、煩雑になる問題点がある。組織内部のメールの送達に MX を使わない場合は単に IP reachable なマシンのみに MX を設定しておけば済むが、組織内部でも MX を使う場合には工夫が必要になる。

このような場合、最終的な MX ホストに対する優先順位を上げて（数字を小さくして）おき、IP reachable なホストの MX の優先順位を下げる（数字を大きくして）おくという方法がある。外部の計算機はメールを送る際、数字のもっとも小さな MX ホストに送達を試みるが、そのホストは外部から IP reachable でないため、SMTP のコネクションを張ることができず、timeout した後、IP reachable な次善の MX にメールを届けることになる。

この方法は外部からメールを送る場合、必ず一度 timeout する点を除いては正常に稼働する。Timeout を防ぐには、IP reachable でない MX ホストに SMTP コネクションを張る段階で、適当なゲートウェイが例えば ICMP unreach host を返答することにより、

すぐに IP reachable な MX ホストへメールを送り届けることができる。メールの送達に必要なパケット数は若干増加するが、timeout を起こさないので気分がいい。

5.4.2 Wildcard

メールを送るとき、メールアドレスのドメイン部と MX レコードの名前が正確に一致した場合のみ MX として採用される。例えば、

user@*host.domain*

にメールを出す場合、

domain IN MX *20* *ms.domain*

という MX レコードは使用されない。*host.domain* と *domain* が一致していないからである。従って、全てのホストに対して MX を宣言しておく必要があるが、これは結構大変である。

このような場合、MX の wildcard（“*”）を使用して MX を宣言すれば良い。つまり

*.*domain* IN MX *25* *ms.domain*

のように宣言しておけばよい。

また、Zone が定義するドメイン自身に対する MX も忘れないようにする。一般的には、次のような記述（あるいは同等なもの）が Zone データに含めるのを忘れないように注意されたい：

```
@  IN  MX  30  ms.domain
*  IN  MX  30  ms.domain
```

ここで *ms.domain* はそのドメインのメール配送を行う、JUNET でいうところのドメインマスタである。

5.4.3 A レコードと MX

ところで、あるホストのアドレスが A レコードで宣言されていた場合、陽に MX が宣言されていない限り `sendmail` はそのホストに直接メールを送ろうとする。そのホストが小さなワークステーションで `sendmail` を走らせていない場合や、X 端末などのようにメールを受信する機能のないホストの場合、そのメールは送信元で timeout してしまうことになる。この場合、例え * で MX が定義されていてもそれは参照されない。

従ってメールを送ってほしくないホストに関しては、適当な MX をいちいち指定することが要求される。

5.4.4　安定に IP reachable でない場合

64Kbps で接続されている場合にはあまり問題ではないが、一部の 9.6Kbps や X.25 網など IP 接続の安定性が悪い場合がある。このような場合、その組織の近隣で安定して IP reachable なホストにもより低い（数字が大きい）優先順位で MX を宣言しておいた方が望ましい。この場合、必ずその MX ホストの管理者に承認を得ておくこと。

5.5　Secondary Server setup

ドメイン名からアドレスへの変換や電子メールの送達に Name Server は重要な役割を果たしている。従って Zone データにアクセスできなくなるような事態は極力避けなければならない。

例えばリンクやゲートウェイがダウンしても、Zone データがに外部からアクセスできる場合、次善の MX ホストにメールを配送することができる。場合によってはその MX ホストから UUCP でメールを転送することができるし、その MX ホストを調べれば何通のめメールが pending になっているかを知ることもできる。

Zone データを常時アクセスできるようにしておく方法は、Autholized Secondary Server を組織外部に依頼することである。それもなるべくネットワークの単一故障に対して影響を受けないようにある程度離れた組織に依頼することが望ましい。

WIDE Internet では、WNOC Fujisawa に接続されているドメインに関しては東京方面に、WNOC Tokyo に接続されているドメインに関しては藤沢や京阪神方面に Autholized Secondary Server を設定するようにしている。

5.6　Name Server の運用

Name Server が安定に動作していることは安定なネットワークサービスを提供する上では重要である。適切に autholized secondary server を設定することは、ホスト等の障害を考える上で重要であるが、Name Server 単体としてきちんと動作させておくことも重要である。

Name Server で、特に海外部分を含めた `in-addr.arpa.` の Secondary をインストールした場合、`named` プロセスは 3MB 近くの仮想空間を消費することになることにも注意が必要である。

また定期的に（例えば 10 分間に 1 回）`cron` で `named` が実際に動作しているか（何らかの原因で落ちていないか）ということを調べ、必要なら再起動するようにすることが望ましい。

5.7　`named` の起動

`named` は経路情報が確立した後に起動するべきである。しかしながら、ホスト名から

アドレスへの変換を伴うプロセスよりは前に起動しておくべきである。特に named を頻繁に利用する sendmail が named より先に立ち上がらないように注意しなければならない。

/etc/gated.conf などの経路情報に関するプログラムの設定ファイルはホスト名からアドレスへの変換を伴わないように、シンボリックな形式ではなく IP アドレスを直接指定するのが無難である。やむを得ない場合には、gated などより先に named を起動する必要があるが、その場合には必要な経路情報を静的に設定しておき、named が正しく情報の授受ができるようにしておくことが大切である。

5.8 Resolver

5.8.1 Resolver の設定

Name Server に対して問い合わせを行なうのが resolver の役割であり、通常 libresolv.a というライブラリとして提供され、プログラムのリンク時に同時にリンクされる。SunOS 4.0/4.1 では dynamic linking がサポートされているので、その機能を活用することもできる。

Resolver の挙動は /etc/resolv.conf というファイルによって決定される。このファイルには、Resolver が問い合わせる Name Server や default ドメイン名が指定されている。このファイルの内容は例えば次のようになっている：

```
domain      wide.ad.jp
nameserver  133.27.48.2
nameserver  131.113.1.1
```

domain 行ではこのホストが発生する Name Server の問い合わせが relative であった場合、この行で指定された default ドメイン名を付加して問合せを行なうことを意味する。nameserver 行は resolver が問合せを行なう Name Server が存在するホストの IP アドレスを指定する。この nameserver 行は最大 MAXNS 個（現在は３）指定することができ、指定された順番に問合せを試みる。最初のサーバからの応答がない場合には次に指定されたサーバ、さらにその次というように問合せを行なう。従って、Name Server が動いているホストのうち、問合せを行なうホストに近い順番に指定するのがよい。

5.8.2 NIS との共存

Sun Microsystems の NIS を運用している場合、ホスト名と IP アドレスの変換は NIS の hosts.byname、hosts.byaddr の各マップによって行なわれる。従って、全てのホストのアドレス情報をこのマップに含めなければならないが、Name Server へ問合せを行なうことによってこのことを実現する方法がある。

条件として、NIS の全てのサーバで Name Server への問合せをサポートしていなければならない。現在知られている Name Server をサポートしている ypserv を含む OS としては、SunOS 4.0.3/4.1[2], NewsOS 4.0 などがある。

ことも必要である。

Name Server への問合せを行なうようにするためには、まずすべての ypserv が動作しているマシンで正しく /etc/resolv.conf が設定されており、組織外部に IP reachable であることを確認する。ついで NIS のマスタサーバマシンの NIS の Makefile を次のように修正する：

> hosts.time を make するところの $(MAKEDBM) のオプションに “-b” を追加する。hosts.byname に 1 箇所、hosts.byaddr に 1 箇所修正箇所がある。

SunOS の場合は NIS マップを検索し、発見できなかった場合には Name Server にアクセスするが、NewsOS の場合はこの順序を変更することができる。詳しくはマニュアルを参照されたい。ただし、この順序関係は ypserv で決定されるので、両者を混合することは避けた方が良い。

5.9 Name Server の問題点

5.9.1 PTR に関するセキュリティホール

Name Server に関するセキュリティホールとして、in-addr.arpa. の PTR レコードの問題が知られている。IP アドレスからホスト名を求める際、Name Server では PTR レコードの検索を行なうが、この検索結果のドメイン名をそのまま信用することは危険である。例を示すと次のようになる。

いま仮に *133.4.1.1* の正しいドメイン名は *jp-gate.wide.ad.jp* であるとしよう。これに対応したレコードは

> *1.1.4.133*.in-addr.arpa.　IN　PTR　*jp-gate.wide.ad.jp*

であるが、これを

> *1.1.4.133*.in-addr.arpa.　IN　PTR　*fake.fooled.edu*

と書き換えた場合、Name Server を経由したアドレス−ドメイン変換の結果は、*fake.fooled.edu* に（*fooled.edu* の許可なく）なってしまう。~/.rhosts にこれが含まれている場合など、第 3 者に login されてしまうことになる。

これを防ぐため、Name Server 経由で得られたドメイン名を用いて、再度そのアドレスを検索し、最初のアドレスが含まれていた場合に限りそのドメイン名を信用するという方法が提案されている。

---

[2]SunOS 3.5 に添付されている ypserv はそのままでは動作しない。また SunOS 4.0 で 4.0.3 以前のものは 4.0.3 に更新が必要である。

これは現在個々の Name Server を用いるアプリケーション毎に対応しなければならない。また NIS を経由して lookup を行なっている場合、ypserv がこの機構をサポートしていることが要求されるが、対応されていない様である。従って個々のアプリケーション毎に、きちんと対応を取った resolver をリンクし直す必要がある。

5.9.2 シリアル番号の問題

Named の各 Zone では、そのシリアル番号を SOA レコードに記述し、Zone データが更新されたことを secondary server などが知ることができる。管理者は Zone データを更新した場合、シリアル番号の更新を忘れないようにしなければならない。

ところで、このシリアル番号には小数点を一つ含めることができるが、その比較は次のようなアルゴリズムで行なわれる：

シリアル番号の一番左の桁から順に、

- 一桁数字を読むたびに値を 10 倍したものにその数字を加える。
- 小数点が現れたら整数部を 1000 倍する。

従って、1 は 1 であるが、1.0 は 10000 になる。また 1.9 は 10009 であるが 1.10 は 100010 になる。

このことは、通常はあまり問題ではないが、小数点ありのシリアル番号の体系を使っている場合、整数部（メジャー番号として用いられることが多い）を変えた場合に問題になる。例えば 1.23 というシリアル番号の Zone データを更新し、2.0 というシリアル番号にしたとしよう。この場合の Named 内部での比較は、前者が 100023 であるのに対して後者は 20002 となってしまい、$100023 > 20002$ であるから Zone データが更新されたことを伝搬することができない。この場合、例えば 2.00 などのように小数部分の桁数を減らすことがないようにすればこのようなことを防ぐことができる。

また、整数部を更新した日付にすることもよく行なわれるが、この場合小数部分があるとその値が Named の内部で unsigned int の範囲で表現できない。従って、この場合には小数部分を使わないようにしなければならない。

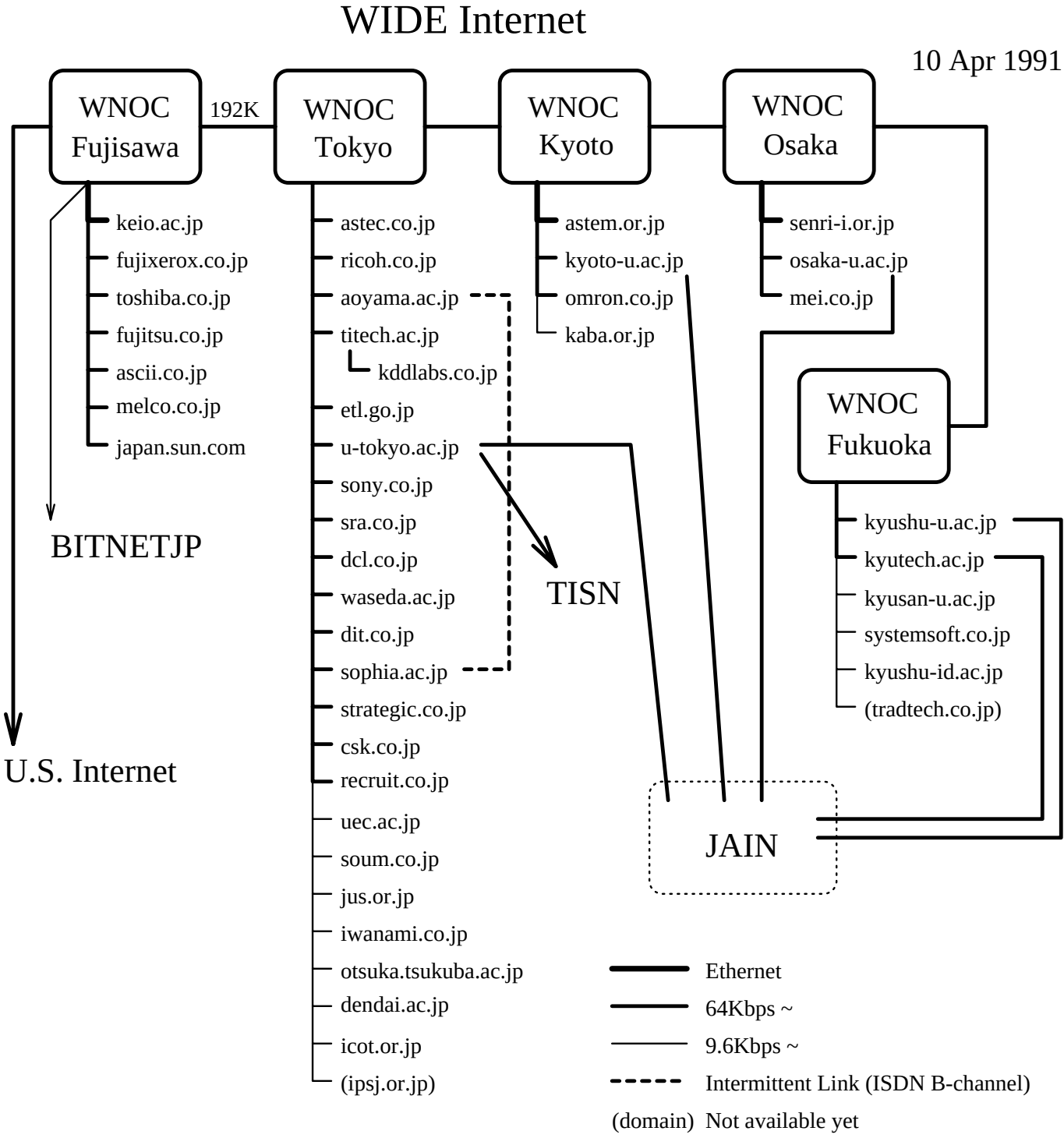
結局次のような方法がお勧めである。

- 小数部分を初めから（例えば）3 桁取って、1.034、2.001 のようにして用いる。
- 一日に十回もデータを更新することはないと仮定して、9104011 のように日付に、その日でのシリアル番号を一桁付け加える。一日に十回以上データ更新の可能性がある場合には、91040101 のように二桁付け加えればよい。

但し、中途で下の方法から上の方法に変更することは避けるべきである。

付録 A

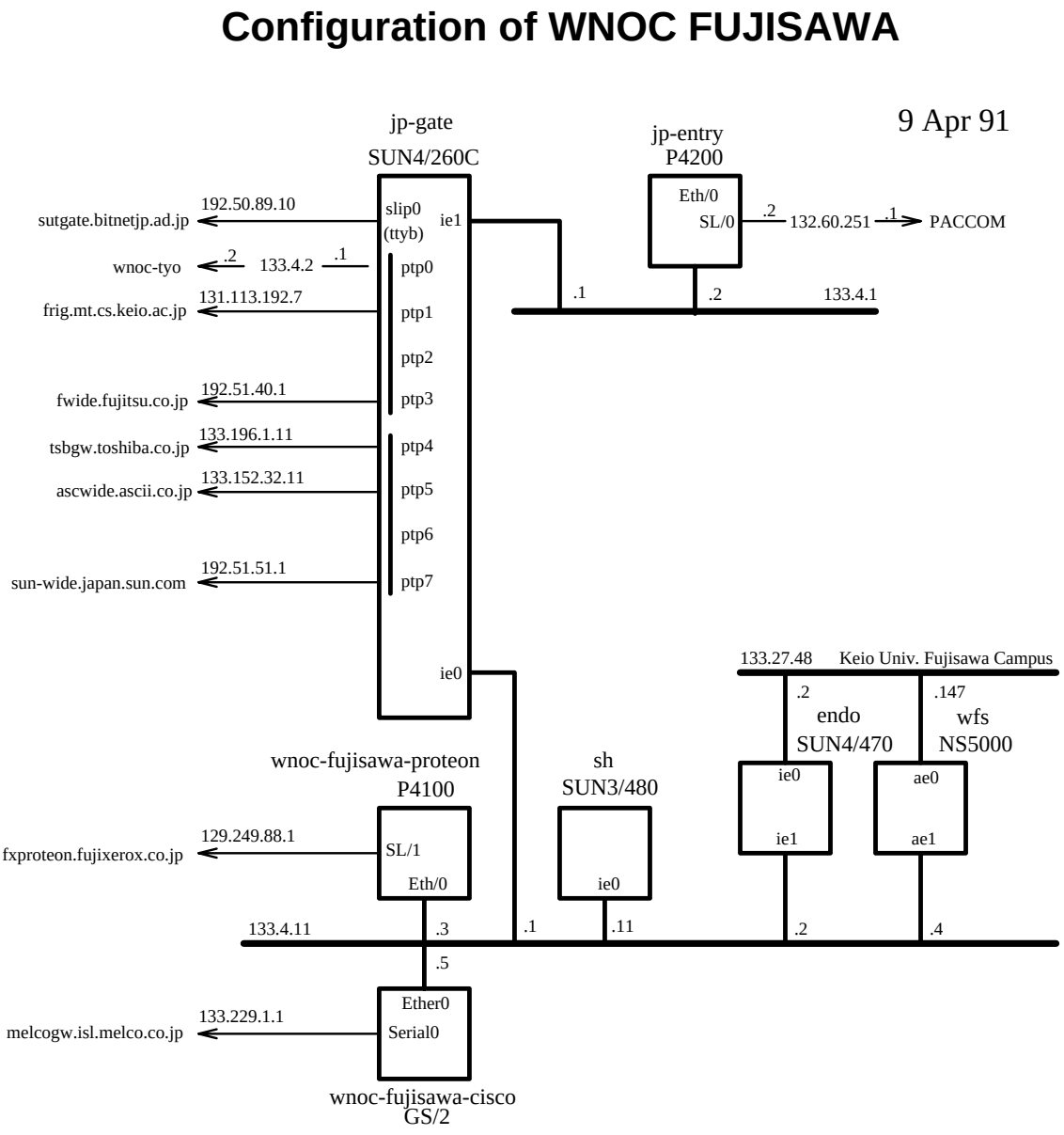
WIDE Internet の構成



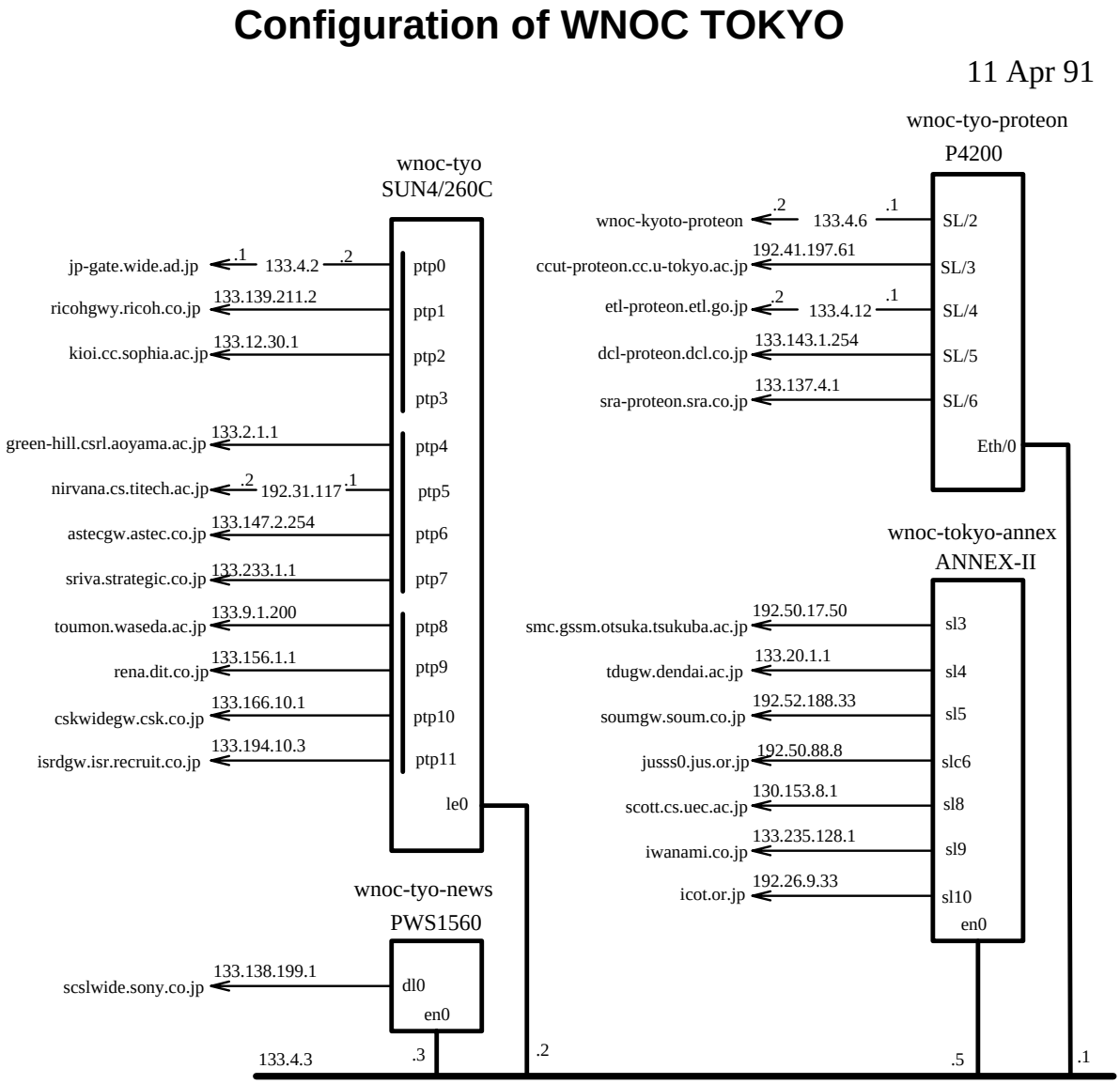
付録 B

WNOC の状況

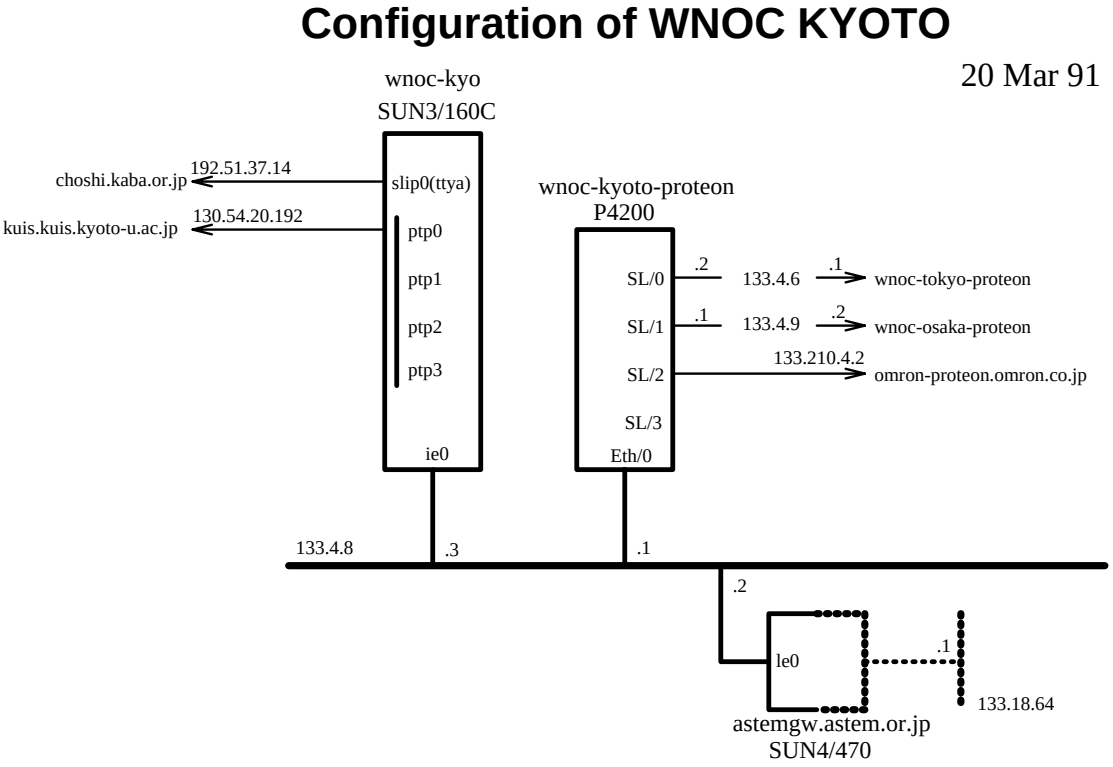
B.1 藤沢



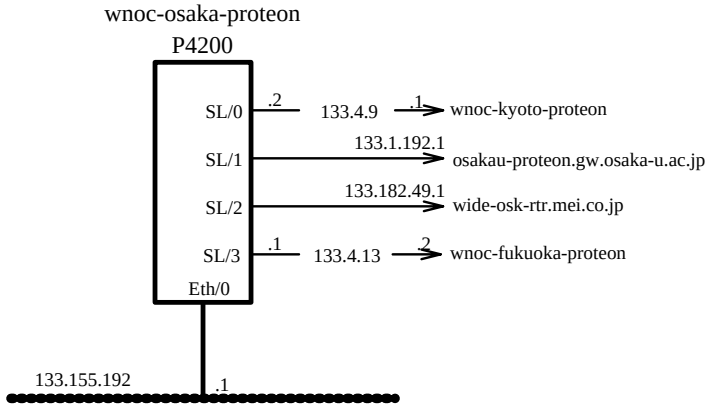
B.2 東京


Configuration of WNOC TOKYO
11 Apr 91
wnoc-tyo-proteon
P4200
wnoc-tyo
SUN4/260C
wnoc-kyoto-proteon
.2
133.4.6
.1
SL/2
jp-gate.wide.ad.jp
.1
133.4.2
.2
ptp0
192.41.197.61
ccut-proteon.cc.u-tokyo.ac.jp
SL/3
ricohgwy.ricoh.co.jp
133.139.211.2
ptp1
etl-proteon.etl.go.jp
.2
133.4.12
.1
SL/4
kioi.cc.sophia.ac.jp
133.12.30.1
ptp2
dcl-proteon.dcl.co.jp
133.143.1.254
SL/5
ptp3
sra-proteon.sra.co.jp
133.137.4.1
SL/6
green-hill.csrl.aoyama.ac.jp
133.2.1.1
ptp4
Eth/0
nirvana.cs.titech.ac.jp
.2
192.31.117
.1
ptp5
wnoc-tokyo-annex
ANNEX-II
astecgw.astec.co.jp
133.147.2.254
ptp6
sriva.strategic.co.jp
133.233.1.1
ptp7
smc.gssm.otsuka.tsukuba.ac.jp
192.50.17.50
sl3
toumon.waseda.ac.jp
133.9.1.200
ptp8
tdugw.dendai.ac.jp
133.20.1.1
sl4
rena.dit.co.jp
133.156.1.1
ptp9
soumgw.soum.co.jp
192.52.188.33
sl5
cskwidegw.csk.co.jp
133.166.10.1
ptp10
jusss0.jus.or.jp
192.50.88.8
slc6
isrdgw.isr.recruit.co.jp
133.194.10.3
ptp11
scott.cs.uec.ac.jp
130.153.8.1
sl8
le0
iwanami.co.jp
133.235.128.1
sl9
icot.or.jp
192.26.9.33
sl10
en0
wnoc-tyo-news
PWS1560
scslwide.sony.co.jp
133.138.199.1
dl0
en0
133.4.3
.3
.2
.5
.1

B.3　京都


Configuration of WNOC KYOTO
20 Mar 91
wnoc-kyo
SUN3/160C
choshi.kaba.or.jp
192.51.37.14
slip0(ttya)
kuis.kuis.kyoto-u.ac.jp
130.54.20.192
ptp0
ptp1
ptp2
ptp3
ie0
wnoc-kyoto-proteon
P4200
SL/0
.2
133.4.6
.1
wnoc-tokyo-proteon
SL/1
.1
133.4.9
.2
wnoc-osaka-proteon
SL/2
133.210.4.2
omron-proteon.omron.co.jp
SL/3
Eth/0
133.4.8
.3
.1
.2
le0
.1
133.18.64
astemgw.astem.or.jp
SUN4/470

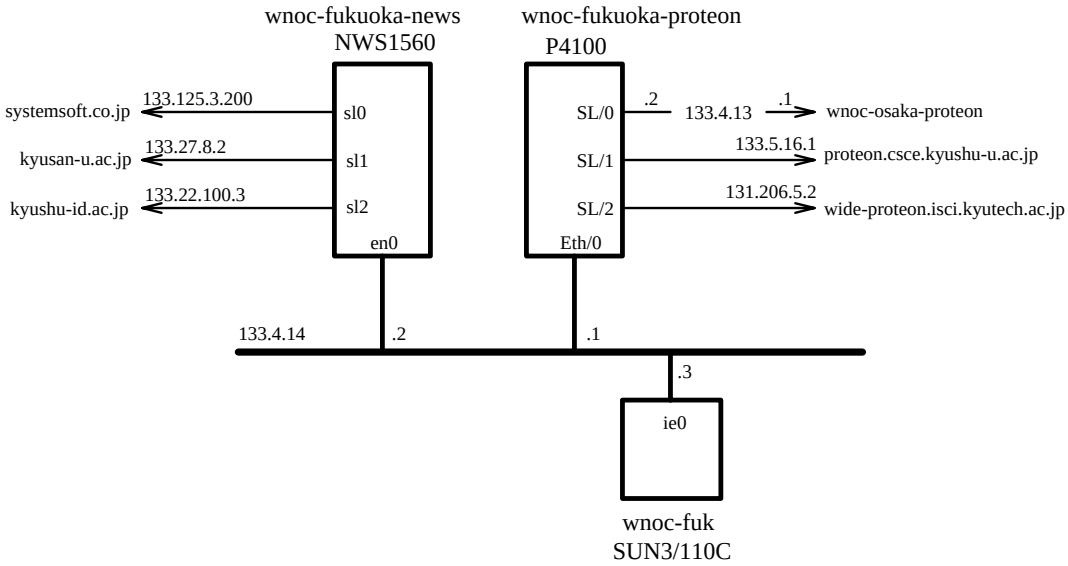
B.4　大阪


Configuration of WNOC OSAKA
20 Mar 91
wnoc-osaka-proteon
P4200
SL/0
.2
133.4.9
.1
wnoc-kyoto-proteon
SL/1
133.1.192.1
osakau-proteon.gw.osaka-u.ac.jp
SL/2
133.182.49.1
wide-osk-rtr.mei.co.jp
SL/3
.1
133.4.13
.2
wnoc-fukuoka-proteon
Eth/0
133.155.192
.1

B.5 福岡

Configuration of WNOC FUKUOKA

20 Mar 91



付録 C

高田メモ

このメモは東京大学理学部情報科学科の高田広章氏によるものである。現在の設定にマッチしていない部分もあるが、参考として氏の許可を得て引用する。

ネームサーバとその設定について（第 1.5 版）

`８９年１０月３日
（最終更新：`９０年５月２４日）

東京大学 理学部 情報科学科
高田広章
hiro@is.s.u-tokyo.ac.jp

このドキュメントは，ネームサーバの設定を行なう際に必要となるテクニカルな面を中心に解説したものである．ＩＰアドレスの取得方法，ドメイン名の取得方法，国内のＩＰネットワークへの接続許可，インターネットへの接続許可については，別のドキュメント（まだできていない？）を参照されたい．

1. ネームサーバとは

ネームサーバとは，一般にホスト名からそのＩＰアドレスを検索するためのサーバのことを言う．ネームサーバの規格には何種類かあるが，以下ではインターネットで使われている BIND（Berkeley Internet Name Domain Sever）について述べる．なお，BIND の詳細については参考文献を参照されたい．

BIND の特徴は，階層構造を持つドメイン名をサポートする分散データベース構造をとっていることで，それぞれのドメイン内のアドレスの管理は，それぞれのドメインの管理者が行なうことになる．

またネームサーバは，電子メイルの転送の際にも重要な役割を果たす．現在 JUNET で使われているメイルの配送ルールは，ドメインごとに転送先を静的に記述するものであるが，このような方法は各ドメインのドメインマスターマシンに大きな負担を強いることになる．ネームサーバを使ったメイルの転送では，ネームサーバに問い合わせることにより目的マシンのアドレスを取りだし，そのマシンにダイレクトにメイルを転送するため，ドメインマスターの負担を軽減し，保守作業を楽にすることができる．また，ＩＰで接続されていないドメインやマシンに対しても，メイルの中継マシンを指定する機能がある（MX レコード）．

インターネットにおけるメイルの転送は，すでにネームサーバを使ったものに移行しており，インターネットからダイレクトにメイルを受け取りたい場合には，ネームサーバの正しい設定が不可欠になる．

2. サーバと resolver

一言にネームサーバの設定と言った場合に，その内容は，サーバ側の設定と resolver 側（ないしはクライアント側）の設定に分かれる．

サーバ側の設定とは，自分のドメインのアドレスを内外に知らせるために，ネームサーバ（unix では `named` または `in.named`）のための設定ファイルを書き，ネームサーバを立ち上げ，そのネームサーバの存在を必要な範囲に知らせることである．このようなネームサーバをそのドメインの primary server という．設定すべきファイルは，`/etc/named.boot` およびその中に記述されるファイルである．また，primary server がダウンしている時に来る問い合わせに対応するため,（authorized）secondary server を立ち上げることも重要である．

resolver とは，ネームサーバを呼び出してホスト名をＩＰアドレスに変換するためのライブラリ群のことで，resolver 側の設定とは，自分のマシンからネームサーバを使うための設定の事を言う．設定すべきファイルは，`/etc/resolv.conf` 等である．ホスト名からＩＰアドレスを得る必要がある各アプリケーション（telnet，ftp，rlogin などなど）は，`/etc/hosts` を検索する代わりに resolver を呼び出してＩＰアドレスを獲得することになるが，このあたりのメカニズムはマシンによって大きく異なるため，自分のマシンがどのようにネームサーバに対応しているか知る必要がある．

また，他のネームサーバに負担をかけないために，他で設定されたデータをキャッシュするためのネームサーバ（プログラムとしては primary server と同じ）を立ち上げる場合が多い．このようなサーバを，secondary server ないしは caching server（この２つは動作が違う）というが，この立ち上げ作業も，resolver 側の設定と考えた方が考え易い．１つのネームサーバが，あるドメインに対しては primary server として働き，他のドメインに対しては secondary server となることも可能である．

この２つの設定は独立のもので，片方だけ行なってもう片方は行なわない事もありうる．

3. BIND の分散サーバの仕組み

BIND の分散ネームサーバの仕組みについて説明する．階層的なドメイン名の意図は，各ドメイン内の設定は，各ドメインの管理者が行なうと言う事である．すなわち，まずインターネット全体に１つ，ルートの primary server が決められている．ルートの primary server は，各トップドメインのネームサーバがどのマシンかを知っている．例えば，インターネットのルートの primary server は，jp のネームサーバがどのマシンかを知っている．同様に jp のサーバは *.jp（ac.jp，go.jp などなど）のサーバを知っており，ac.jp のサーバは，*.ac.jp（u-tokyo.ac.jp，keio.ac.jp などなど）を知っている．それを検索する側のネームサーバは，最低限ルートのネームサーバのみ知っていれば，そこから順にたどって任意のドメインに属するホストのアドレスを取り出せることになる．

そのため，あるドメインの primary server の設定が終ったら，上位ドメインのネームサーバの管理者に連絡して，上位ドメインの設定内に，下位ドメインのネームサーバに関する記述を追加してもらう必要がある．

4. 日本国内の事情

最近，日本国内でもＩＰリンクのある範囲が急速に拡大しており，またインターネットとのＩＰリンクができたこともありネームサーバの設定の必要性が高まってきた．

ところが，国内のネットワークでネームサーバを設定する際に重大な問題があることが指摘された．それは，以下のような問題が起こる事である．インターネットから見えるネームサーバに，インターネットから接続できないマシンがはいっていた場合に，インターネットからのメイルはそのマシンに直接送られようとするが，接続できないためメイルの送信ができなくなってしまう．かと言って，インターネットからは接続できないが，国内では接続されているマシンがネームサーバを使って検索できないのは不便である．

そこで，日本国内のＩＰネットワークを運営するにあたって，次のような方法を用いてこの問題を解決することにした．すなわち，サーバ側の設定を行なう際に，インターネットから見えるネームサーバと，国内から見えるネームサーバとを別々に立ち上げることとし，前者のサーバには，国内的には接続されているが，インターネットからは接続できないマシンを登録してはならないものとする（誤って登録した場合，メイルが届かなくなることがある）．もちろん，この２つのサーバに登録されている内容が全く同じ場合（つまり，すべてのマシンにインターネットから接続できる時）には，１つのサーバで兼用できる．また，インターネットから接続できるマシンを持たないドメインに対しては，国内向けのサーバのみでよいことは言うまでもない．

つまり，インターネットから接続できるマシンのみを登録したネームサーバ系列Ａと，国内で接続できるすべてのマシンを接続できるネームサーバ系列Ｂの２つの系列があることになる．ところが，前者のネームサーバからは国内のマシンの一部が検索できず，後者のネームサーバからは海外のマシンが検索できないため，これに加えて，国内のマシンも国外のマシンも検索できる第３のネームサーバ系列Ｃが必要になる．このネームサーバの系列は，他の２つのサーバ系列へ問い合わせに行くだけ（つまり，secondary server

または caching server）なのでサーバ側の設定は必要なく，ドメインの階層の意味で系列化されているわけではない．

5. ネームサーバ系列の組織

4. で述べた３つのネームサーバ系列を整備する必要があるわけだが，この際に重要となるのは，実際にどのマシンがどの系列のサーバとなるかをはっきりさせることである．それぞれの系列のマシンは次のように使われる事になる．

系列A 海外から見えるマシンを管理する．海外からのみアクセスされるため，海外とのリンクがある所に置いた方が信頼性が上がり，トラフィックも減らせる．なお，この系列に属するマシンは，自分の上で動いているネームサーバを使わず，`/etc/resolv.conf` の設定により他のマシンのネームサーバを呼ぶ事になる．

系列B 国内のＩＰネットワークに接続されているすべてのマシンを管理する．国内の海外に接続できないマシンからアクセスされる．インターネットとは無関係なので，国内からアクセスしやすい場所に置き，必要なら secondary server を立ち上げるのが良い．

系列C このサーバではマシンの管理は行なわない．国内の海外に接続できるマシンからアクセスされる．海外のサーバへもアクセスするため，海外リンクがあるところに置き，必要なら secondary server を立ち上げるのが良い．

これらの事情を考えて，現在次の様に組織している．(* は primary server．系列A，Bのその他は，authorized secondary server）

	系列B	系列C	系列A
root	ccut.cc.u-tokyo(*)	—	nic.ddn.mil. (*) などなど
jp	ccut.cc.u-tokyo(*)	kogwy.cc.keio utsun.s.u-tokyo	jp-gate.wide.ad.jp.(*) ns.tisn.ad.jp.
*.jp	ccut.cc.u-tokyo(*)	kogwy.cc.u-tokyo utsun.s.u-tokyo	jp-gate.wide.ad.jp.(*) ns.tisn.ad.jp.
u-tokyo	ccut.cc.u-tokyo(*)	relay.cc.u-tokyo utsun.s.u-tokyo	ns.tisn.ad.jp.(*) jp-gate.wide.ad.jp.

前にも述べたように，系列Cの系列化は，他の２つの系列と意味あいが異なる．各ドメインにネームサーバを立ち上げる際には，この表に，それぞれのドメインについての欄を追加していく事から始めることになる．

なお，ネームサーバの設定にあたっては，これらのマシンでの設定を参考にするとよい．

逆引き用のサーバ（in-addr.arpa）に対しても同様の方法をとる．jp-gate および ns.tisn は，海外に接続されたすべてネットワークの逆引き用の secondary server となり，インターネットに対しては jp-gate および ns.tisn をサーバとして登録する．ccut は，国内のネットワークのみを含んだ in-addr.arpa の primary server となる．系列Cのマシン（つまり，そのおおもとの kogwy および utsun）からは，インターネットの arpa ドメインの secondary server となり，かつ国内の in-addr.arpa ドメインの secondary server となることで，インターネットに接続されているネットワークと，国内からのみ見えるネットワークの両方を検索できる．これは，インターネットのネームサーバにおいて，逆引きのアドレスが arpa ドメイン内に直接記述されている事を利用している．

6. ネームサーバ設定の際の注意

ネームサーバ設定の際に，参考文献 [1][2] には書いてないが注意すべき点を挙げる．

6.1 ANY について

参考文献の [1] には，ＩＰに限らない情報欄には ANY と書くようにあるが，named は ANY に対応していない場合がある．すべて IN とするほうがよい．

6.2 MX レコードについて

MX レコードは，メイルの配送の際に重要となる設定である．ワイルドカードを用いた MX レコードの書き方は，便利であるが使い方に注意が必要である．それは，ワイルドカードで MX レコードが書いてあっても，そのもののアドレスが書いてあれば，MX レコードが不優先になることである．例えば，

```
$ORIGIN                 is.s.u-tokyo.ac.jp.
*       IN      MX      0 spica.is.s.u-tokyo.ac.jp.
spica   IN      A       133.11.14.1
acrux   IN      A       133.11.14.5
```

とあると，acrux.is.s.u-tokyo.ac.jp 宛のメイルは spica ではなく，acrux にダイレクトに送られる．これを変更したい場合には，

```
$ORIGIN                 is.s.u-tokyo.ac.jp.
*       IN      MX      0 spica.is.s.u-tokyo.ac.jp.
spica   IN      A       133.11.14.1
acrux   IN      A       133.11.14.5
        IN      MX      0 spica.is.s.u-tokyo.ac.jp.
```

というように，ワイルドカードを使わずに個別に MX レコードを指定する必要がある．そのため，多くのマシンについて，アドレスは登録したいがメイルはメイルサーバで受け取りたい場合や，ネームサーバには登録されているがＩＰ接続できない場合には，マシンごとに MX レコードを書く必要がある．

6.3 メイルを受け取れないマシンについて

ネットワーク上にＩＰアドレスを持つのは，メイルを受け取れるマシンばかりではない．例えば，端末サーバやＰＣは通常メイルを受け取る事はできない．このようなマシンに対して，アドレスのみを設定しておくと，万一そのマシンに対してメイルを送ろうとしたユーザがいた場合に，タイムアウトするまで（３日とか７日とか）リトライしつづけることになる．

これを回避する１つの方法は，6.2 に書いた方法でそのようなマシンに対する MX レコードを，他のメイルが受け取れる適当なマシンに対して設定し，そのマシンで適切に処置をすることである．適切な処置としては，そのマシンでユーザ名を解釈してしまうか，エラーとして返送するかのどちらかとなろう．前者のようにするには，メイルを受け取れないマシンの名前を，メイルを受け取るマシンの別名であるかのように `sendmail.cf` を設定すればよい．

6.4 ドメインに対する MX レコードについて

ユーザ名@ドメイン名でメイルを受け取りたい場合には，ドメインについての MX レコードを指定する必要がある．今までの JUNET のコンベンションにあわせて，これを指定する方がよい．ドメインに対する MX レコードを指定する時は，@ を用いると良い．例えば，

```
$ORIGIN                 is.s.u-tokyo.ac.jp.
@       IN      MX      0 spica.is.s.u-tokyo.ac.jp.
```

とすると，`user@is.s.u-tokyo.ac.jp` 宛のメイルは，`spica` に届く事になる．その他のホスト名でないアドレスに対してメイルを受け取りたい場合にも同様の方法が使える．ただし，`sendmail.cf` がそのアドレスを自分宛として解釈できる事が前提となる．

6.5 その他のレコードについて

HINFO レコードはコメントにすぎないので，設定するかどうかは各ドメインで決めてよい．WKS については，設定した方が安全である．その場合，今の段階ではとりあえずは smtp だけ書いておけばよい（メイルを受け取り損なう心配はなくなる）．MB，MR，MINFO の各レコードは，まだ実験中でほとんど使われていないようだ．

6.6 `forwarders` について

系列Cのネームサーバの内，jp ドメインの secondary server となっていないものは，forwarders の指定を自分より上位の系列Cのネームサーバに対して指定する必要がある．ここで上位というのは，系列Cのルートサーバ，すなわち kogwy.cc.keio.ac.jp または utsun.s.u-tokyo.ac.jp に近いサーバをいう．また，上位のサーバがダウンしていた場合のために，forwarders に複数のサーバを指定する事が望ましい．この場合，近いサーバの方を前に書くべきである．さらに安全にするためには，slave 指定をするか，jp ドメインの secondary server となればよい．

このようにしないと，一度でもルートサーバに検索に行った際に，jp の primary server として jp-gate.wide.ad.jp ないしは ns.tisn.ad.jp を教えられてしまい，正しく国内のアドレスを取り出す事ができなくなるからである．また，海外に対するネームサーバの検索を減らす効果もある．

6.7 authorized secondary server と unauthorized secondary server

ドキュメントには明確な記述がないが，secondary server には authorized と unauthorized の２種類がある．authorized secondary server とは，primary server の設定ファイル中および１つ上位のドメインの設定ファイル中にその server に対する NS レコードが書かれているもので，primary server の管理者の承認なしに立ち上げることはできない．authorized secondary server は，primary server のダウンに備えて，少なくとも１つ用意した方がよい．この場合に，authorized secondary server は，必ずしも自ドメイン内に置く必要はない．

unauthorized secondary server はその他の secondary server で，自由に立ち上げる事ができる．これはむしろ resolver 側の設定の一部と考えた方が良い．

この２種類の server の最大の違いは，ホスト名等をルートの側から探す場合に，もし primary server にアクセスできなければ authorized secondary server にアクセスするのに対して，unauthorized secondary server は/etc/named.boot ないしは /etc/resolv.conf の中で明示的にアドレスを指定されない限り，使われることはないことである．

6.8 その他の注意

ネームサーバの設定ファイルの SOA レコード中の Serial フィールドは，secondary server のデータの取り込みの際に重要な役割を果たしている．設定データを更新するたびに，必ず値を増やす必要がある．

また，ネームサーバの設定ファイルを更新した場合には，ネームサーバを起動しなおすか，HUP シグナルを送る必要がある．

ネームサーバの設定ファイル中のホスト名は，最後に “.” がない限り，そのファイルのオリジンからの相対名である．MX レコードや PTR レコード等で最後の “.” を忘れないように注意する必要がある．ファイルのオリジンは，$ORIGIN 指定で変更されない限り，/etc/named.boot から得られる．

NS または MX の記述は各設定ファイル中で完結していなければならないようだ．これは系列Aのネームサーバ（つまりインターネットのルートから検索できるドメインの

ネームサーバ）の場合は *推奨* だが，系列Bのネームサーバで系列Aのサーバを兼ねていないもの場合は *必須* となるようだ（確かではない）．特に逆引きファイル中では忘れがちなので，注意が必要である．

多くのドメインの secondary server になる場合，ネームサーバプロセスは思った以上にメモリを使うので注意が必要である．例えば，utsun のネームサーバは，約 2MB のメモリを使っている．

6.9 ヒント

ネームサーバを設定した後で，secondary server を立ち上げてそのキャッシュファイルをチェックしたり，named の状態のダンプをチェックする事は，設定の間違いの発見に役立つ場合が多い．

7. ネームサーバの嘘つき問題

rsh や rlogin を実行する際に，hosts.equiv ないしはホームディレクトリの .rhosts に呼びだし側のマシン名が書いてあるとパスワードのチェックが省略されてログインできるが，ネームサーバを使ってＩＰアドレスからホスト名への変換を行なっている場合，ネームサーバの管理者がその気になれば，自分の管理しているアドレスに対して嘘の名前を付けることができる．例えば，

```
v.u.y.x.in-addr.arpa.    PTR      utsun
```

という記述をいれることで，ＩＰアドレスが x.y.u.v のマシンから，.rhosts に utsun が含まれているアカウントに侵入できることになる．

この問題は分散ネームサーバに本質的な問題と考えられ，解決はなかなか難しいが，以下のような方法が考えられる．

1. 純粋な 4.3BSD の場合
   ucb-fixes の 85 番の修正を行なう（ucb-fixes は，utsun から anonymous ftp できる．unix/ucb-fixes）．この修正は、ネームサーバを使ってアドレスからホスト名を取り出したあとで，今度はホスト名からアドレスを取り出して，元のアドレスに一致するかどうかをチェックするものである．この方法はかなり有効であるが，自分の上位ドメインのネームサーバの管理者だけは信用する必要がある．

2. SunOS4.0 の場合
   あくまで暫定手段であるが，ypserv が両向けの検索を別々に設定できることを利用して，ＩＰアドレスからホスト名を引く際にネームサーバを参照しないように設定する（詳しくは９章を参照）．逆引きが必要になることは少ないので，通常の使用には影響が少ないはず．

3. その他の場合
   上のどちらの方法も適用できず，ソースリストもない場合には有効な手段はないと考えられる．とりあえずは，resolver を使わないように設定するしかない．

4. 根本的な解決
   そもそも hosts.equiv や .rhosts に書けるマシンは，管理者が認めた安全なマシンに限るべきものであると考えられる．このようなセキュリティにかかわるようなＩＰアドレス－ホスト名変換の場合には，ネームサーバを参照せずに，管理者が認めたマシンのみが記述されているローカルなデータベースを使うようにすべきであろう．

8. ネームサーバの配置の例

8.1 u-tokyo.ac.jp ドメインの場合

u-tokyo.ac.jp（東京大学）ドメイン内には，インターネットから接続できるマシンと接続できないマシンの両方がある．そのため，系列Ａと系列Ｂのネームサーバの両方を別々に立ち上げる必要がある．また，トラフィックを減らし，検索を効率良く行なうために，系列Ｃのネームサーバも立ち上げた方がよい．

実際には，5 章の表に示した通り，系列Ａの primary server として ns.tisn を，系列Ｂの primary server として ccut を用いる事とした．また，系列Ｃの name server として relay を設定する．6.6 に示したガイドラインにより，relay は forwarders 宣言を用いて kogwy と utsun を指定する．

東大内のクライアントマシンの内，インターネットと接続可能なものは，ネームサーバとして relay を用いる．ただし，１つの組織（学部，学科）内に多くのクライアントマシンがある場合には，relay の負担を軽減するために，ローカルにネームサーバを持つ事が望ましい．このローカルなネームサーバは，必要な（よくアクセスする）ドメインの unauthorized secondary server となり（全く secondary server にならなくても，ネームサーバはキャッシュを行なうので，立ち上げる効果はある），forwarders として relay を指定する．forwarders として kogwy または utsun を指定する事も可能であるが，トラフィックを減らすために relay を最初に指定するのが望ましい．

一方，インターネットと接続不可能なものは，ネームサーバとして ccut を用いる．この場合も，１つの組織内に多くのクライアントマシンがある場合には，ローカルにネームサーバを持つ事が望ましい．このローカルなネームサーバは，必要な（よくアクセスする）ドメインの unauthorized secondary server となり，forwarders として ccut を指定する．forwarders の指定はオプションである．

8.2 s.u-tokyo.ac.jp ドメインの場合

s.u-tokyo.ac.jp（東京大学理学部）ドメイン内のマシンは，すべてインターネットから接続できる．そのため，系列Ａと系列Ｂのネームサーバの両方を別々に立ち上げる必

要はない．トラフィックを減らし，検索を効率良く行なうために，系列Cのネームサーバは必要となる．

実際には，系列A，Bの primary server として ns.tisn を用いる事とした．また，系列Cの name server としては utsun が自ドメイン内にあるので，それを用いる．

理学部内のクライアントマシンはネームサーバとして utsun を用いる．ただし，１つの組織（学科）内に多くのクライアントマシンがある場合には，ローカルにネームサーバを持つ事が望ましい．このローカルなネームサーバは，必要な（よくアクセスする）ドメインの unautorized secondary server となり，forwarders として utsun を指定する．実際，いくつかの secondary server が立ち上げられており，utsun のネームサーバへ直接アクセスするマシンを減らすようにしている．

9. 各マシン／ＯＳにおける resolver 対応

以下で named が新しいかどうかは，BIND4.8 以降でサポートされている directory，forwarders，slave と言った機能があるかどうかを示す．なお，BIND の最新版は BIND4.8.2 であり，BIND4.8 よりかなり改良されているので，こちらを使うのが好ましい．

4.3BSD

1. named は新しいバージョンがリリースされている．
2. クライアントは，すべて BIND に対応している．ただし，ライブラリを作る際にそのように設定する必要がある．

Sun OS 3.5

1. named は古いバージョンが標準．BIND4.8.2 を若干の修正でコンパイル可能．
2. クライアントは BIND に対応していない．

サンマイクロから，BIND 対応のパッチテープを入手する事ができる．このパッチテープを入手すると以下のようになる．

1. named は新しいバージョンがはいっている．
2. ypserv が BIND に対応する．ただし，SunOS 4.0 以降とは対応が方法が異なる．

Sun OS 4.0

1. named は新しいバージョンが標準．BIND4.8.2 を若干の修正でコンパイル可能．
2. ypserv が BIND に対応する．ただし，SunOS 4.0.1 の ypserv にはバグがある．上で書いたパッチテープに修正版がはいっている．4.0.3 では修正済み．また，共有ライブラリ中の関係するライブラリを交換することによって対応させることも可能である．

SunOS4.0 で ypserv がネームサーバを見るように設定する方法はマニュアルが不親切でわかりにくい．SunOS4.0 では，YP 用の関係する DBM ファイルにネームサーバを参照するようにマークをつけることになる．具体的には，/var/yp/Makefile の

```
| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/hosts.byname; \
```

という行を

```
| $(MAKEDBM) -b - $(YPDBDIR)/$(DOM)/hosts.byname; \
```

に修正し，touch /etc/hosts; make することで，ホスト名前からアドレスを取り出す際にネームサーバを参照するようになる．さらに，

```
| $(MAKEDBM) - $(YPDBDIR)/$(DOM)/hosts.byaddr; \
```

という行にも -b をつけることで，逆引きの際にもネームサーバを見にいくことになるが，上述のネームサーバの嘘つき問題があるためにこの設定はお勧めできない．

VAX (Ultrix 3.0)

1. named は新しいバージョンが標準．
2. クライアントは，BIND に対応している．/etc/svcorder により，yp，bind，local をどういう順番で見にいくかが設定可能．

LUNA (UNIOS-B 1.20Beta)

1. named は古いバージョンが標準．
2. クライアントはすべて BIND に対応している．まず YP を見に行って，見つからなかった場合のみ named を見に行く．

Apollo SR10

1. named は古いバージョンが標準．BIND4.8.2 をいくつかの修正でコンパイル可能．
2. クライアントはすべて BIND に対応しているようだ．（resolv.conf があるかどうかではなく，ローカルに named がいるかどうかで named を呼ぶか決めているらしいが，詳細は不明．）

他のマシン／ＯＳについては未調査．

10. 配布

11. 謝辞

このドキュメントの内容に有益なコメントを下さった村井純氏，尾上淳氏，その他の皆様に感謝します．

参考文献

[1] Dunlap, K.J. and Karels, M. J., Name Server Operation Guide for BIND (Release 4.8).

[2] manual page for NAMED(8), RESOLVER(3)(5), GETHOSTBYNAME(3).

[3] RFC974, MAIL ROUTING AND THE DOMAIN SYSTEM.

[4] RFC1032, DOMAIN ADMINISTRATORS GUIDE.

[5] RFC1033, DOMAIN ADMINISTRATORS OPERATIONS GUIDE.

[6] RFC1034, DOMAIN NAMES - CONCEPTS AND FACILITIES.

[7] RFC1035, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION.