第 18 部

WIDE インターネットの現状と運用技術

第 1 章

TWO

1.1 現在の WIDE バックボーン

WIDE プロジェクトは、1997 年 3 月 31 日現在、国内に 16ヶ所、海外に 1 ヶ所のネットワークオペレーションセンタ (以下、NOC) を維持している。各 NOC は、専用線で相互接続され運用されている。図 1.1に、各 NOC の相互接続図、及び使用している専用線の回線帯域を示す。

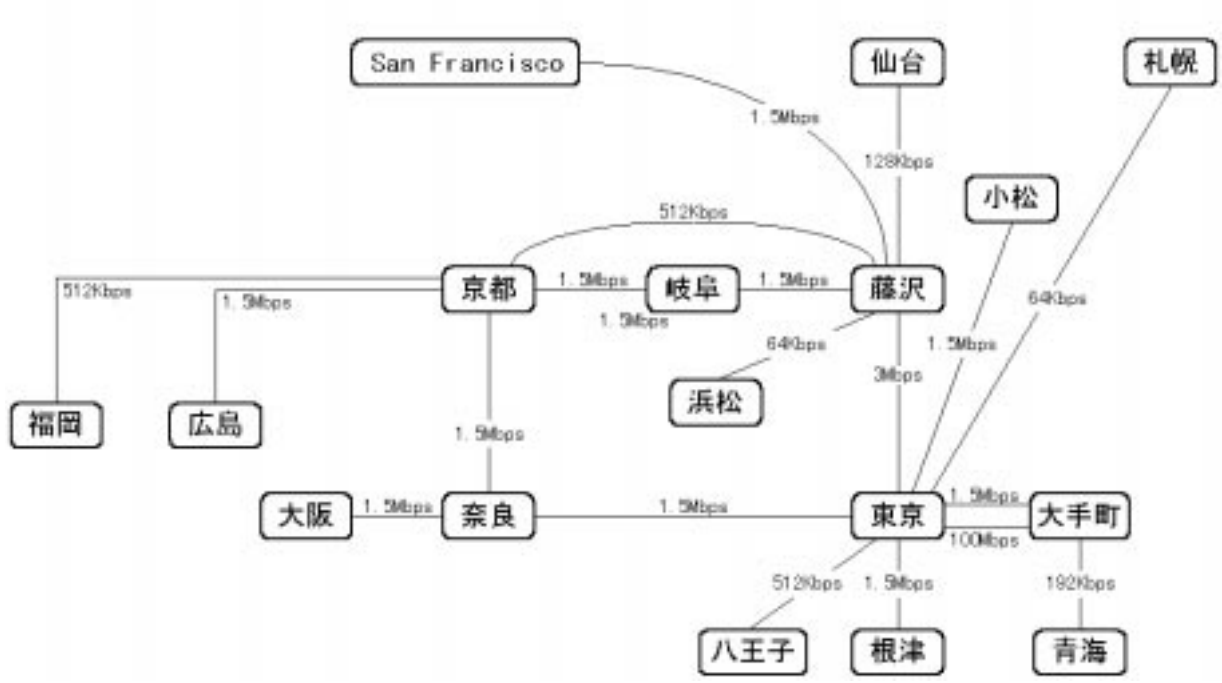

図 1.1: WIDE インターネットバックボーン

これらの NOC 及びそれらを結ぶネットワークを WIDE インターネットバックボーンと定義する。

WIDE インターネットバックボーンは、東京、藤沢、奈良、京都をバックボーンの中心とし、他の NOC はそれらに接続する形で構成されている。

図 1.1から分かるように、東京阪神間を結ぶ経路として以下の 3 経路が存在する。

- 東京 → 奈良 → 京都
- 東京 → 藤沢 → 京都
- 東京 → 藤沢 → 岐阜 → 京都

これら複数の経路は OSPF で管理され使い分けられている。回線障害などのトラブル発生時には、動的に経路が変化することで障害に強いネットワークを実現している。

また、次節で示すとおり、東京 NOC では Ethernet を用いて NSPIXP-1 と相互接続し、大手町 NOC では FDDI を利用し NSPIXP-2 と相互接続している。その他、BGP を利用して地域系のプロバイダや、各種 ISP と相互接続を行い経路の交換を行っている。

1.2 ネットワークオペレーションセンター

本節では、各 NOC のトポロジを示す。WIDE インターネットに接続している組織は省略し、各 NOC との相互接続、及び他 AS との相互接続を表すルータのみ記述する。また、東京、藤沢、奈良、京都の四ヶ所はコア NOC とし、それ以外の NOC はリーフ NOC と表現する。

トポロジを示す NOC は、コア NOC である東京、藤沢、奈良、京都の四ヶ所と、海外線の終端であるサンフランシスコ NOC、NSPIXP2 が設置されている大手町 NOC、リーフ NOC のトポロジの例として青海 NOC とする。

1.2.1 東京ネットワークオペレーションセンター

東京 NOC は、WIDE プロジェクトに参加する多くの組織の相互接続点である。また、NSPIXP-1 も東京 NOC 内に存在している。NSPIXP-1 はイーサネットスイッチで構成され、cisco9.tokyo 及び cisco12.tokyo の二つのルータが NSPIXP-1 に接続されている。その他、バックボーン内に RIP のパケットが流れないようにするために、cisco8.tokyo を介して RIP 用のセグメントを構築している。

藤沢 NOC と東京 NOC は、1.5Mbps の専用線、及び 1.5Mbps の Flame Relay を用いて接続されている。その運用は、OSPF の Equal Cost Multi Path 機能を用いて行われ、トラフィックの分散を行っている。

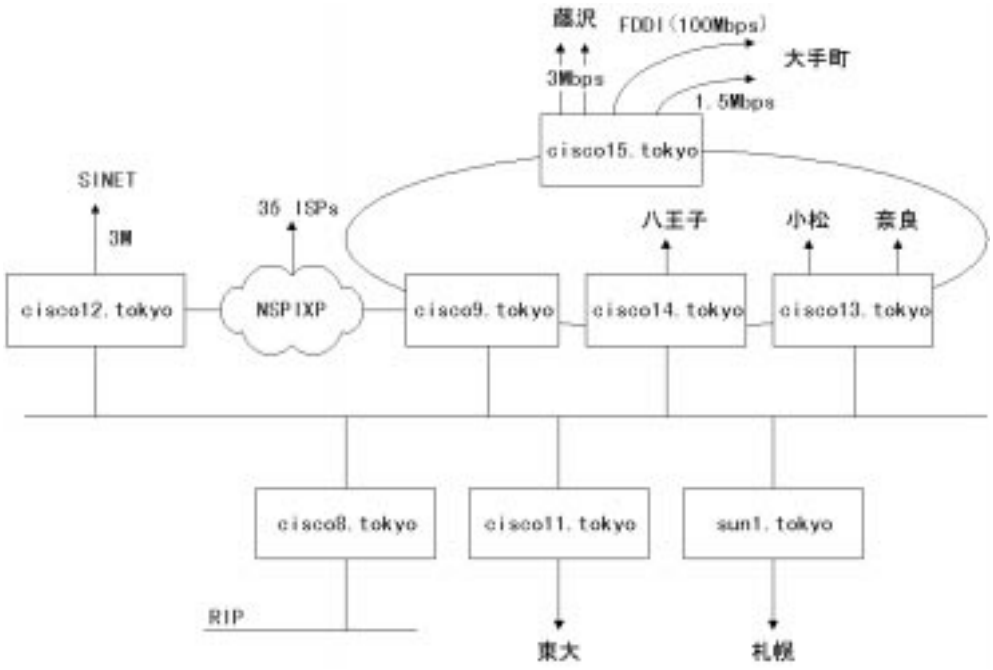

図 1.2: 東京 NOC トポロジー図

1.2.2　大手町ネットワークオペレーションセンター

大手町 NOC は、一台のルータのみで構成されている。そのルータから NSPIXP-2 に接続されている。その他、各種 ISP や他の AS とも相互接続している。

1.2.3　青海ネットワークオペレーションセンター

青海 NOC は典型的なリーフ NOC の構成をしている。一本のイーサネットセグメントから構成され、一台のルータがコア NOC 接続用として設置されている。また、イーサネットには NOC に接続される各種組織のルータが設置される。青海 NOC の場合、APNIC などが接続されている。

1.2.4　藤沢ネットワークオペレーションセンター

現在、藤沢 NOC はリナンバリングの最中であり、新旧のバックボーンが混在している状況である。古いバックボーンはイーサネットを利用したネットワークであり、新しいバックボーンは FDDI 及び FDDI イーサスイッチを利用して構成される。今後、FDDI インターフェースを利用しているルータは FDDI のリングに接続され、イーサネットのインターフェースを利用しているルータは、FDDI イーサスイッチに接続される予定である。

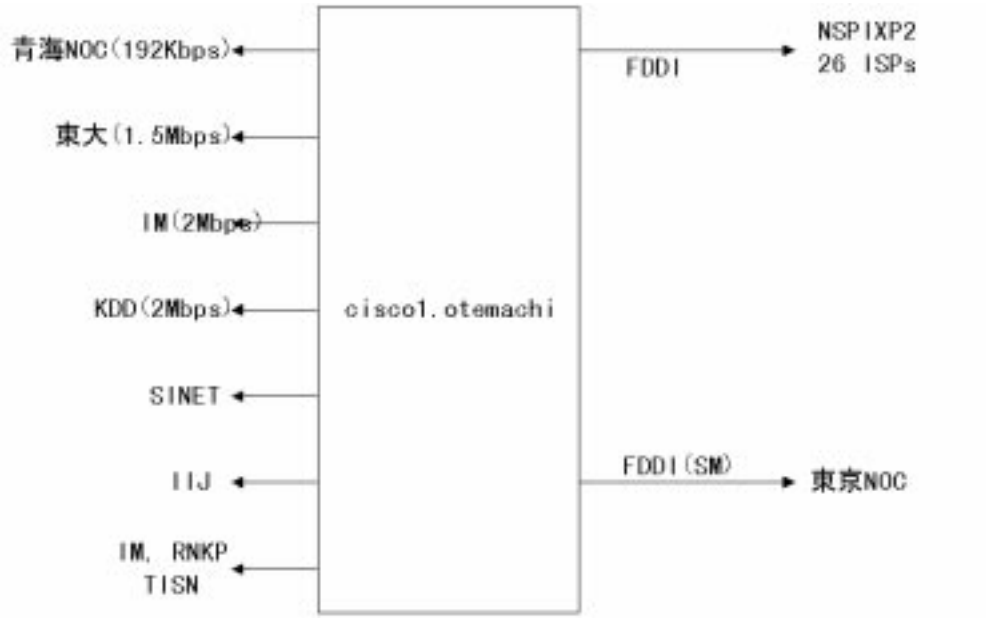

図 1.3: 大手町 NOC トポロジー図

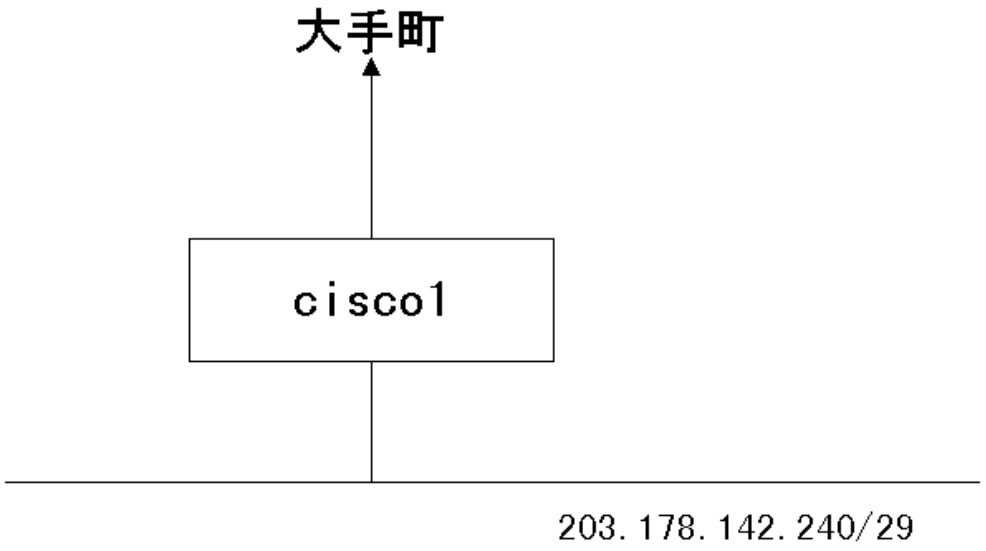

図 1.4: 青海 NOC トポロジー図

また、東京 NOC と同様に、バックボーン内に RIP のパケットが流れないように、RIP しか利用することのできないルータを別セグメントに接続している。

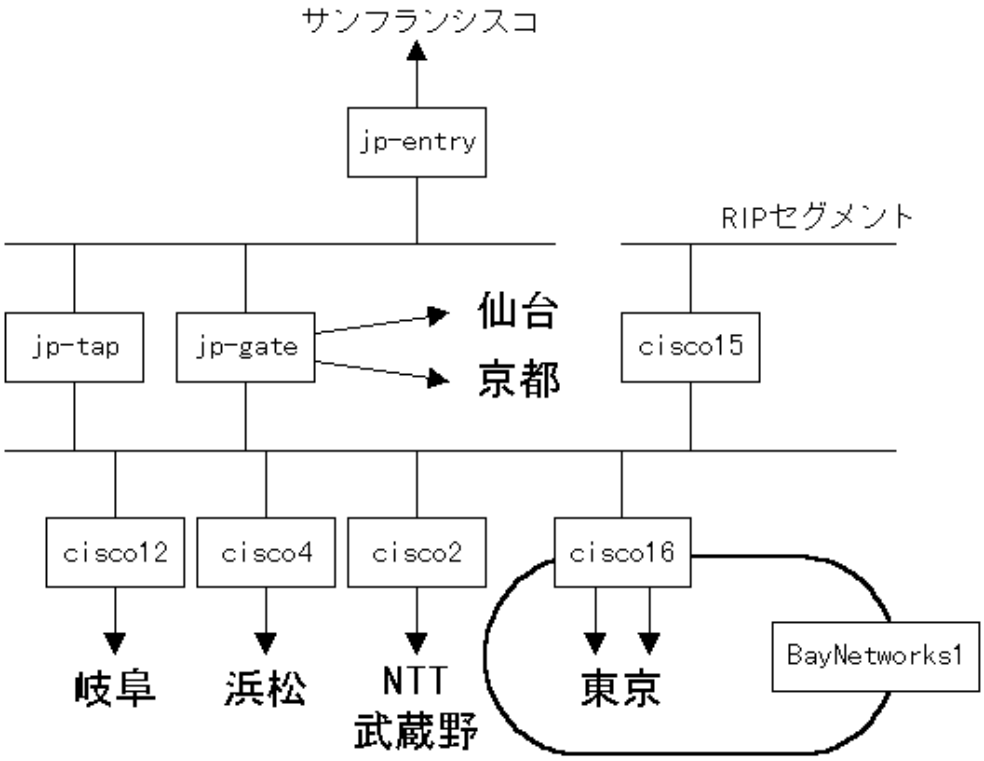

図 1.5: 藤沢 NOC トポロジー図

1.2.5　京都ネットワークオペレーションセンター

京都 NOC は関西地区より西に位置する NOC の HUB になっている。トポロジとしては、一つのイーサネットセグメントに複数のルータが接続している形で構成されている。

1.2.6　奈良ネットワークオペレーションセンター

奈良 NOC も一つのイーサセグメントから構成されている。AI3 プロジェクトと IMNet と cisco2.nara を介して BGP を利用し経路の交換を行っている。

1.2.7　サンフランシスコ ネットワークオペレーションセンター

藤沢 NOC から 1.5Mbps の専用線を利用し接続されているサンフランシスコ NOC は、一台のルータから構成されている。AS 境界ルータとして、MCI と NSI と相互接続してお

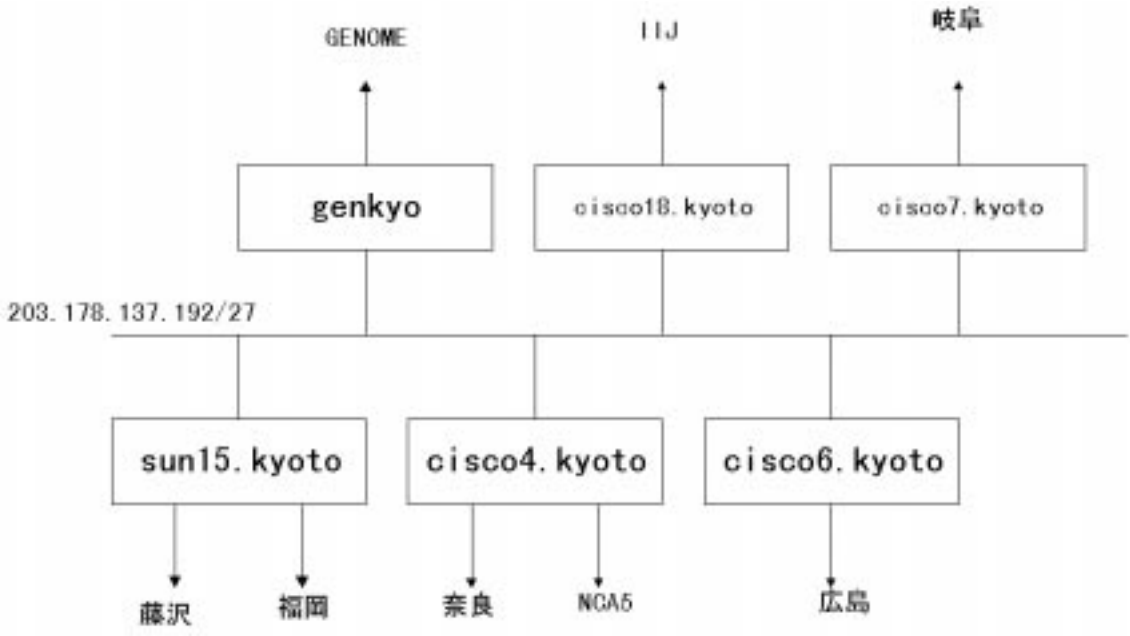

図 1.6: 京都 NOC トポロジー図

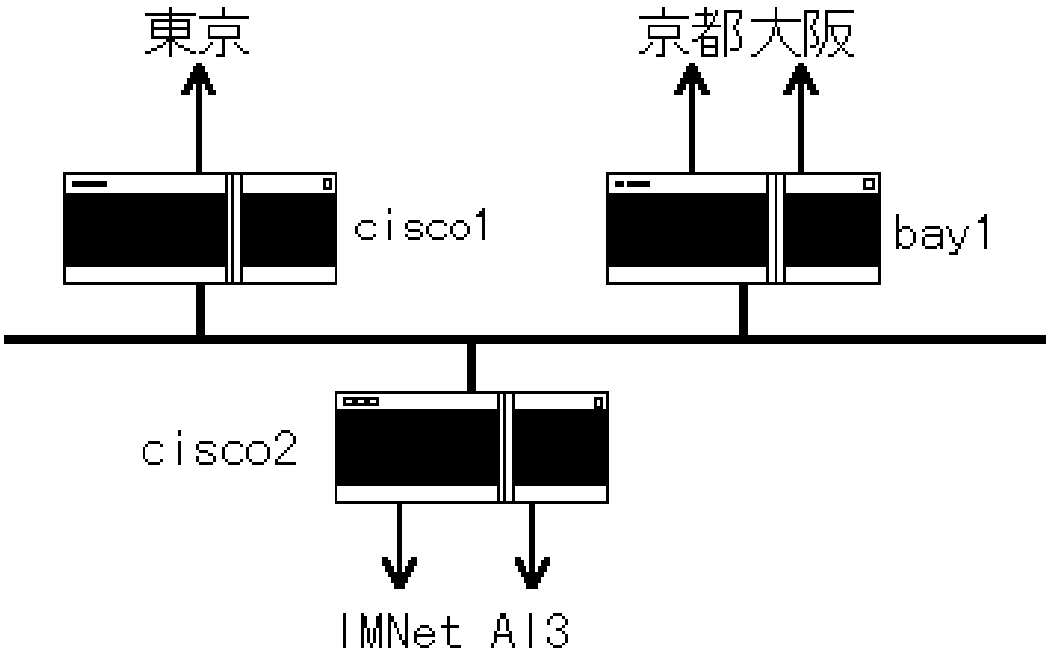

図 1.7: 奈良 NOC トポロジー図

り経路の交換を行っている。また、地理的に離れていることから、障害発生時に備え遠隔地からルータを操作できるように、コンソールにはモデムが接続されている。

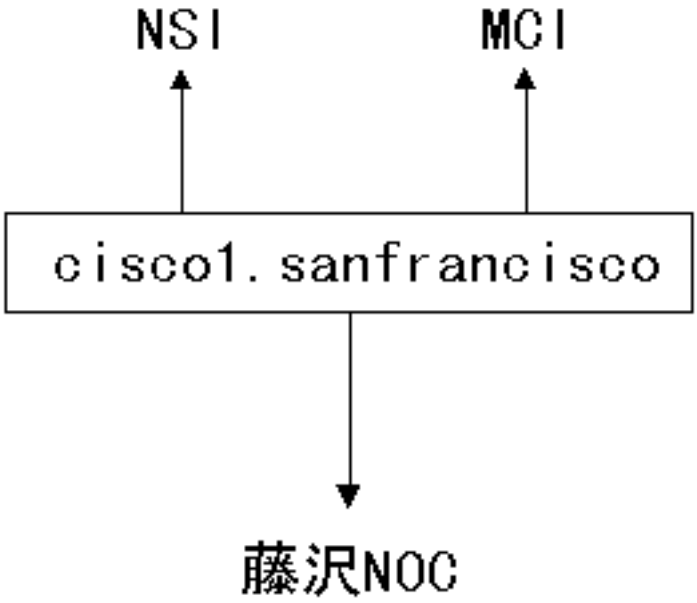

図 1.8: サンフランシスコ NOC トポロジー図

1.3　リネーム

1.3.1　名前空間利用方法に関するルール

WIDE インターネットに接続されるホスト、及びルータは次のルールにしたがってホスト名をつけなければならない。各ホスト/各ルータは、それぞれのベンダを表す文字列に続けて、そのホスト/ルータを区別するための数字を用いてホスト名がつけられる。表 1.1それぞれのベンダ名とベンダを示す文字列の対応表を示す (順不同)。

また、ホスト名だけでなくドメインを示す名前空間においても利用方法に関するルールを定めた。ネットワークを表すネットワークアドレスには、”net” サブドメインを利用することとし、各 NOC のフルスペルをそれぞれの NOC のサブドメインとした。表 1.2に挙げるサブドメインが存在する。

また、NOC 用サブドメインの他に、必要性が認められたプロジェクト用にも表 1.3のようにサブドメインが割り当てられている。

また、各プロジェクトメンバーの自宅にあるネットワークが、WIDE インターネットに直接接続されていて、WIDE のアドレス空間からアドレスが割り当てられている場合、自

表 1.1: ベンダ名とベンダを示す文字列の対応表

Sun Microsystems	sun
Cisco Systems	cisco
Bay Networks, Inc.	bay
ソニー	sony
Telebit	netblazer
Proteon	proteon
ヤマハ	yamaha
IBM	ibm
富士通	fujitsu

表 1.2: NOC サブドメイン一覧

sapporo.wide.ad.jp	sendai.wide.ad.jp
hachioji.wide.ad.jp	tokyo.wide.ad.jp
nezu.wide.ad.jp	otemachi.wide.ad.jp
aomi.wide.ad.jp	fujisawa.wide.ad.jp
gifu.wide.ad.jp	hamamatsu.wide.ad.jp
komatsu.wide.ad.jp	kyoto.wide.ad.jp
nara.wide.ad.jp	osaka.wide.ad.jp
hiroshima.wide.ad.jp	fukuoka.wide.ad.jp
sanfrancisco.wide.ad.jp	

表 1.3: プロジェクトサブドメイン一覧

ai3.wide.ad.jp	expo96.wide.ad.jp
iaa.wide.ad.jp	kirc.wide.ad.jp
nspixp.wide.ad.jp	sfc.wide.ad.jp
v6.wide.ad.jp	camp.wide.ad.jp

宅が存在する場所の地名からサブドメインを割り当てることが認められる場合がある。

これらのルールに従い、各ホスト/ルータにホスト名を割り当てる。例外としては jp-gate や sh や endo および ns など特定の認められたものだけがサブドメインを持たずに WIDE.AD.JP ドメイン直下に置かれている。その他のサービスホスト用の名前に関しては、それぞれが置かれている NOC のサブドメインの下に登録されることになった。

これらのリネームは、広域ネットワークの管理運用に関して、その運用コストを下げるために行われた。ホスト名やサブドメイン名の割り当てに一定のルールを設けることにより、ネットワークの運用者が容易に障害が発生しているホストやルータにを識別できるようになる。しかし、クラッカーなどのシステム破壊者に帯して、ホスト/ルータに関する情報を提供していることにもなり、クラッキングの足掛かりにされる可能性がある点が問題となる。

1.3.2　リネームにおける注意点

今回のリネームを進めていく上においていくつかの注意すべきことが前もって考慮されていたが、実際に行なうことで他にもいくつかの予期せぬ注意点が得られた。

基本的にはリネームする時に旧名を新名の別名として残すことで大丈夫であるが、注意点は大きく分けて、ネームサーバでの逆引き結果が変わることによる影響、ホスト名自体が変わることによる影響、旧名が別名になる影響、の 3 つに分類される。

ネームサーバでの逆引き結果が変わることによる影響は逆引きのホスト名によってなんらかのホスト認証を行なっているシステムに発生し、例えばニュースサーバや IRC サーバなどサーバホスト同士がコネクションを張る時のホスト認証で問題となりうる。また、ユーザサイドでは rlogin や RPOP などのホスト認証に影響がある。これらは設定ファイルに新旧両方の名前を登録しておくことで移行中も問題を避けられる場合が多い。

ホスト名自体が変わることによる影響としては、例えばメールにおいて sendmail.cf などの設定で対応しないと、旧名もしくは新名だけでしかメールを受けとることができなくなりエラーを起こすことがある。また直接問題を引き起こすわけではないが、自分のホスト名から自分の属すドメイン名を得ることで同じドメインに属す他ホストについてはホスト名だけに省略して出力をする syslogd や traceroute などの例もある。

旧名が別名になることによる影響としては、ネームサーバでの設定においてネームサーバやメールサーバの指定は正式名を使わなければならないと定められているため、指定に旧名を用いていた場合は新しい名前を指定するように対応しなければならない。

1.4　リナンバリング

現在、インターネットの全体の流れとして、巨大なアドレス空間であるクラス A、クラス B を保有する組織に対して、本当に必要とするアドレス空間を再割当するかわりに、本来所有しているアドレスを返却するという活動が進められている。それにより、実際に利

表 1.4: 203.178.136.0

0-31	BGP Speaker	32
32-63	Service 系ホスト	32
64-151	NOC 間接続用 Point-to-Point Network	30
152-159	NOC 間接続用 Point-to-Point Network(予約分)	30
160	東京 NOC 1	27
192	東京 NOC 2	27
224	大手町 NOC 2	29
232	大手町 NOC 3	29
240	東京 NOC—大手町 NOC 2	29
248	大手町 NOC 1	30
252	予約分	30

用されていないアドレスを多くの組織に割り当てられるようになり、現行のバージョンのインターネットのアドレス割り当てに支障がでなくなる。

WIDE プロジェクトでは、クラス B のネットワーク１個とクラス C のネットワークを９個保有している。WIDE プロジェクトも保有するクラス B を返却し本当に必要とするアドレス空間だけを新たに割り当ててもらうリナンバリングを行うことにした。本節では、リナンバリングをするにあたり、再割当されたアドレス空間を効率的に利用する方法、及びリナンバリングに関して注意しなければならない点をまとめた。また、WIDE プロジェクトでは、リナンバリング後のアドレス空間としてクラス C のネットワーク８個の大きさに抑えた。

1.4.1 アドレスの割り当て方法

リナンバリングに伴う新しいアドレスの割り当て方法として、以下の点に気を付けた。アドレスからネットワークの種類を特定できるよう特定の種類のアドレスはまとめて割り当てたり、今後の拡張を見越して予めアドレス空間を予約したり、必要に応じて経路を集約できるようにした。

以下の表に、その割当した結果をまとめる。

1.4.2 リナンバリング時における注意点

リナンバリング実行時は、様々な問題がが発生する。本節では、作業時に明らかになった問題点をまとめ、注意するべき事項について述べる。

表 1.5: 203.178.137.0

0	大阪 NOC 2	28
16	藤沢 NOC バックボーン	29
24	京都 NOC バックボーン	29
32	福岡 NOC 1	28
48	福岡 NOC 2	28
64	藤沢 NOC 1	28
80	藤沢 NOC ボーダ	28
96	福岡 NOC 1	28
112	福岡 NOC 2	28
128	八王子 NOC	27
160	奈良 NOC 1	27
192	京都 NOC 1	27
224	京都 IAA 用	28
240	京都 NOC 2	28

表 1.6: 203.178.138.0

0	広島 NOC	28
16	仙台 NOC	28
32	札幌 NOC	28
48	浜松 NOC	28
64	小松 NOC	28
80	岐阜 NOC 1	28
96	岐阜 NOC 2	28
112	大阪 NOC 1	28
128	藤沢 NOC 2	28
144	サンフランシスコ NOC 1	29
152	川崎 (楠本)	29
160-219	6Bone 用	
220-223	藤沢 NOC—川崎	30
224-255	WISH 用	

表 1.7: 203.178.139.0

0	成城 (村井)	27
32	八王子 ISDN 1	28
48	八王子 ISDN 2	28
64	横浜 (中村)	29
72	サンフランシスコ NOC 2	29
80-255	藤沢 (研究用)	

表 1.8: 203.178.140.0

0-127	藤沢 (研究用)	
128-191	WISH 用	
192	根津 NOC	28
208	武蔵野 (NTT)	28
224	奈良 NOC 2	27

表 1.9: 203.178.141.0

0-63	藤沢 (研究用)	
64-159	高速ネットワーク (予約)	
160-255	藤沢 (研究用)	

表 1.10: 203.178.142.0

0-63	VIP	
64-159	藤沢 (研究用)	
160-239	予約	
240	青海 NOC	28
248	予約	30
252	大手町 NOC—青海 NOC	30

表 1.11: 203.178.143.0

0	ワークショップ 1	26
64	ワークショップ 2	26
128	ワークショップ 3	26
192	ワークショップ 4	26

まず、DNS を通じてサービスを受けるアプリケーションについて述べる。現在使われている多くのアプリケーションは、ネットワーク上でサービスを提供している計算機を指定するために、FQDN(fully qualified domain name) を用いる。したがって、このようなアプリケーションは DNS の変更を適切に行なうことによってサービスを提供する計算機の新しいアドレスエントリを自動的に取得するため、原理的な問題は発生しないと考えられる。しかし、サービスを提供している計算機を FQDN で指定している場合には、DNS 情報更新の伝播遅延が原因となり、サービスの提供を受けられないという問題が発生する。この問題を回避するため、今回はリナンバリング作業中のみ DNS のリフレッシュ時間を短縮した。リフレッシュ時間を短縮することは、DNS のトラフィックを増大させる問題を引き起こし、ネットワークに多大な負荷を掛けることになる。今回のリナンバリング作業時には、リフレッシュ時間を 15 分に設定した。この時間設定によってネットワークに極端な負荷がかかることはなかったと考えられるが、最適であるかどうかの判断をするには、理論的な検討が必要になる。

また、実際のアドレス更新作業時には逆引き用エントリを先に登録しておき、正引き用エントリは作業と並行して頻繁に更新を行なう方法を取った。

次に, DNS を参照しないアプリケーションについて述べる. syslog など、アドレスを起動時にのみ設定するクライアントは個別に再起動する必要があった。また、resolv.conf を参照しているアプリケーションは DNS サーバの提供するアドレスの設定変更に追随しなかった。このため、これらのアプリケーションについても個別に再起動する必要があった。

さらに sendmail や NTP などのように、計算機を指定するために IP アドレスを記述することがあるサービスに対しては、リナンバリングを行なった後もソフトウェアインターフェイスを用いることで、一定期間旧アドレスによるアクセスを可能とした。ソフトウェアインターフェイスに関しては次節で詳しく述べる。

1.5 ソフトウェアインターフェース

1.5.1 定義

ソフトウェアインターフェースとは、狭い意味では物理的に直接対応するインターフェースを伴わずにソフトウェア的に処理されるインターフェースを指すことが多く、仮想インターフェースなどと呼ばれることもある。実現方法としては、別個のインターフェースを作成して利用する方法と、物理的に直接対応するインターフェースの alias として実現する方法や、1 つの物理セグメント上に複数の論理セグメントを実現する論理インターフェースなどがあるが、ここではそれら alias や論理インターフェースを用いる方法などすべてを含めてソフトウェアインターフェースとして扱う。

1.5.2 経路制御

論理セグメントを伴う論理インターフェースは通常の arp によって解決されるが、そうでない場合のうち、そのホストの属する物理インターフェースに割り当てられた IP アドレス空間内の IP アドレスを持つソフトウェアインターフェースの場合は、自分自身への proxy arp を設定することで経路を確保することができる。一方、違う IP アドレス空間から割り当てられた IP アドレスを用いている場合は、経路情報をアナウンスすることによって他のホストからそのソフトウェアインターフェースまでの経路が確保される。ソフトウェアインターフェースによって仮想セグメントが作られている場合はその IP アドレス空間を、そうでない場合はその IP アドレスのホストルートを経路として流すことで実現される。

proxy arp 方式を使用することの利点は、他へ経路情報を流さなくてすむことや、その物理セグメントの IP アドレス空間内から IP アドレスを割り当てて使えることである。一方、ホストルートなど経路情報をアナウンスする方式の利点は、特定の物理セグメントに依存せずに自由に移動設定できることや、その IP アドレス自体を全く別の空間から自由に割り当てることができる点である。また、特定の物理インターフェースに依存せずに経路を確保できる。

ホストルートなどを流す時は数が多くなった場合の拡張性が気になるが、実際に使用される IP アドレスはその組織内や AS 内で閉じている場合が多いので、それぞれ組織外や AS 外へ出る時には経路集成されるために大丈夫である。

1.5.3 利用例

仮想ドメイン空間

仮想ドメイン空間を実現するためにソフトウェアインターフェースを用いることは多い。この場合、別個に用意したソフトウェアインターフェースもしくは既存のインターフェースへの alias を使って新たに IP アドレスを割り振り、その IP アドレスを異なるドメイン空

間に割り当てることで同一のホストに複数の仮想ドメイン空間を実現する。例えば、1 台のホストで複数ドメインの WWW サービスなどをするのに用いられる。特別なことをしない限り proxy arp 方式を使用すれば十分である。

単に複数ドメイン分のサービスを行なうならばネームサーバの別名設定を用いるだけでも十分であるが、その場合サーバ側においてクライアントがどの名前を指定してアクセスしてきたかを一般的には知ることができない。一方、ソフトウェアインターフェースによって個別の IP アドレスを割り振っておけば、サーバ側はクライアントによって指定されたこちら側のインターフェースの IP アドレスを見るだけで知ることができる。

落ちないインターフェース

WIDE の場合、複数の物理インターフェースを抱えるルータが同時になんらかのサービスホストを兼ねている場合が多く、ある 1 つのインターフェースががなんらかの事情で落ちている時にそれらのサービスに影響がでることがあった。そのサービスホストへアクセスするためにネームサーバをひいた時、すべての物理インターフェースの IP アドレスが登録されていれば、たまたま落ちているインターフェースに当たった場合はサービスを受けられないかもしくはタイムアウトするまで待たされる結果となる。また、どれか 1 つのインターフェースの IP アドレスだけ登録されている場合、そのインターフェースが落ちてしまうと他のインターフェースからそのサービスホストへ到達可能にもかかわらずサービスを受けることができない。

これらの問題点を解決する 1 つの方法として、ソフトウェアインターフェースを用いた落ちないインターフェースを設定するという手がある。そこには各物理インターフェースとは無関係な IP アドレスを割り当て、同時にそのホストルートを各物理インターフェースから経路アナウンスするという形をとることで上記の問題を解決することができる。

落ちないインターフェースを用いるのにふさわしい例の 1 つとして BGP スピーカがある。IBGP のコネクションを交換するインターフェースとして特定の物理インターフェースに依存しているとそれが落ちた時に機能しないため、落ちないインターフェースを用いるのがふさわしい。また、WIDE においては 203.178.136.0/27 の空間にこの BGP 用インターフェースの IP アドレスを集めることで運用上管理を容易にしている。

WIDE のプライマリネームサーバである jp-gate も同時にルータとして複数の物理インターフェースを抱えているが、今回落ちないインターフェースとして 203.178.136.63 及びその名前 ns.wide.ad.jp を新たに割り当て、従来からのルータとしての jp-gate とネームサーバとしての ns.wide.ad.jp を分離することで、特定の物理インターフェースに依存しない運用体制を確立した。これらのサービス用 IP アドレスの空間として 203.178.136.32/27 が用意されている。

サービスホストとその移設

一般のサービスホストについてもソフトウェアインターフェース用いることの利点は多い。例えば上記のようにその物理的ホストとサービスホストを分離することで、物理的ホストが落ちてしまった時などにソフトウェアインターフェースごと他のホストへ移設することで、サービスを続行することができる。

もしもネームサーバにおいて別名を用いることでサービスホスト名を使用していた場合は、別のホストへサービスを移設した時にネームサーバの別名定義も変更しなければならない上、ネームサーバのデータがインターネット上へ伝わるには遅延が生じるためその間サービスが止まってしまうという問題もある。また、長時間動き続けるデーモン類は起動して一度ネームサーバへひきにいくと同じホスト分については二度とネームサーバへ再度ひきにいくことをしない実装のものも多いが、サービスホストと IP アドレスが固定に結びつけられている場合はこれらのものをわざわざ配慮する必要がなくなる。

その他にサービスホスト専用に IP アドレスが割り当てられることの利点は、その IP アドレスの逆引きをそのサービスホスト名自身に設定することができることである。これはそのサービスホスト自体から他へコネクションを張るようなタイプのサービスの場合、その相手側においてのホスト名認証においてそのサービスホスト名を用いることができるようになる。例えばニュースサーバや IRC サーバ同士のコネクション認証がこれに該当し、もしもニュースサーバをしている物理ホストを変更したりしても相手側に設定ファイルの変更をお願いする必要などがなくなる。

サービスのバックアップ

上記において、サービスホストのトラブル時に他へ影響を与えずに別のホストへとサービスを移設することができることを書いたが、サービスによってはさらに考えを進めてサービスの自動バックアップが実現できる。これは同時に複数のホストにおいて同じ IP アドレスのソフトウェアインターフェースを持たせ、例えばそれぞれからのホストルートの経路アナウンスメトリックに差をつけることで通常はメインのホストがサービスを引き受け、それが落ちた時には別のホストがバックアップしてサービスを継続できる。

さらに状況によってはネットワーク的に離れた位置で同じ IP アドレスを持つ複数のサービスホストを設置することで最寄りのところからサービスを受けるようにできる場合もある。これらはまさにホストルート方式によってエニーキャストのようなしくみを実現して提供していることになる。

これらをソフトウェアインターフェースを用いずに行なうにはそのサービスホスト名に対してそれぞれの物理ホストの IP アドレスをネームサーバにおいて登録しておかなければならないが、実際の利用において複数の IP アドレスのうちどれかが落ちていればタイムアウトを待たなければならないなど問題が多い。

リナンバリング

リナンバリングの移行時において新旧両 IP アドレスを持つことを実現するには、1 つの物理セグメント上に複数の論理セグメントを実現できる論理インターフェースを用いるのが好ましい。しかし、そのような論理インターフェースは特定の OS でしかサポートしていないことが多く、また、特に新旧両 IP アドレスを保持しておかなければならないようなホストは特定のものに限られるため、そのような場合には個別にソフトウェアインターフェースを設定することで解決することができる。

セキュリティ

サービス毎に専用のソフトウェアインターフェースを割り当てる他の利点としては部分的にセキュリティ面でも向上に役立てることができる。これはサービスなどをしているホストが特に侵入などでねらわれやすいことから、例えばそのサービス用インターフェースにおいては telnetd など他のデーモン類を立ち上げないことで実現する。つまり、インターフェース毎に立ち上げるデーモン類を必要最低限なものに制限する。

1.5.4　実装例

WIDE バックボーン上のサービスホストではまだ主力をつとめる SunOS4 の場合、インターフェースに対する alias をすることができず、また、gated の実装では IFF_LOOPBACK であるインターフェースについている IP アドレスを他へ経路アナウンスするのを抑制しているため、自分から経路をアナウンスするには IFF_BROADCAST もしくは IFF_POINTOPOINT のソフトウェアインターフェースを作って行なう必要があった。特に IP アドレス空間節約でホストルートを用いるために IFF_POINTOPOINT のソフトウェアインターフェースを実際に作成して運用している。cisco では IBGP 用に Loopback0 インターフェースにその IP アドレスをつけることで落ちないインターフェースとして運用している。

1.5.5　サーバ側での対応

サービス毎もしくはドメイン毎にソフトウェアインターフェースを割り当てて運用する場合、サーバ側ソフトにおいて対応をしていないと有効に活用することができない。例えば WWW サーバの多くはクライアントからどのインターフェースを指定されたかによって動作を変えられるようになっている。

sendmail では DaemonPortOptions にてデーモンがコネクションを待つインターフェースの IP アドレスを指定することができるようになっている。これによって仮想ドメイン毎に別個の sendmail を立ち上げることができたり、セキュリティ上特定のインターフェースでだけ待ち受けることが実現できる。ただし現時点の 8.8.5 ではサーバから他へコネクショ

ンを張りに行く時に自分側のアドレスを DaemonPortOptions で指定したものへ設定していないので、運用上問題が発生する場合は自分でパッチを当てて運用する方が好ましい。

ネームサーバの場合も基本的には相手が指定してきたインターフェースをソースアドレスとして返事をするだけなので問題はないが、バルクでデータを転送するのを正式なセカンダリサーバだけにフィルタリングを設定して制限している場合は、そのセカンダリサーバ側がデータをとりに行く時のソースアドレスに注意が必要となる。フィルタリングの設定で対応するか、もしくは簡単なパッチを当ててサービスホスト用の IP アドレスがソースとなるようにする。

ニュースサーバの場合も同様で、対応していないプログラムを使う場合は相手への記事転送のパケットのソースアドレスが常にサービスホスト用になるような簡単なパッチを当てて運用しておくことで、特定の物理ホストにしばられることなく運用することができるようになる。

一般に、相手先へコネクトをする部分のところで前もって自分側のアドレスを指定するようにするだけでそれらは簡単に実現できる。また、種々のテスト用として自分の発信アドレスを自由に指定できるような telnet を用意しておくと便利である。