DNS & Mail

中村 素典(京都大学)

1998 年 12 月 15 日

InternetWeek 98 国立京都国際会館

(社)日本ネットワークインフォメーションセンター編

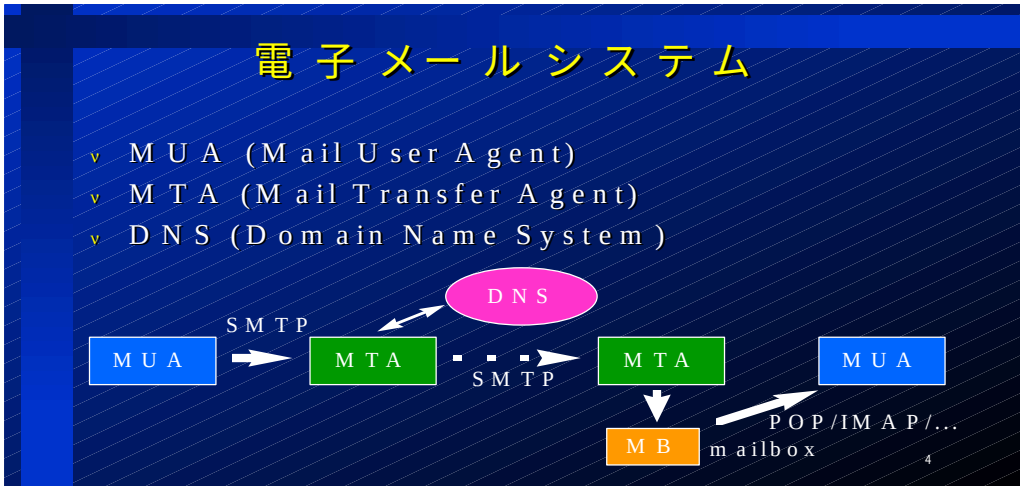
概要

電子メールはインターネットにおける最も重要な基本サービスの一つですが、メールサーバを構築するために必要となる基礎知識は多岐にわたります。特に、DNS(Domain Name System)**に大きく依存します。ここでは、メールサーバ管理者として必要になる** DNS**、メールシステムの設定、運用に関する実践的なトピックを中心に解説します。前半は** MTA**、電子メールの配送系と** DNS **との関連に中心を置いて説明し、後半部分ではメールシステムのデザインとして、バックアップサーバやファイアウォールなどについて説明します。最後に、最近、非常に問題となっている** SPAM **に関して、その対策などを説明します。**

目次

1. インターネットメールの基礎

電子メールシステム



電子メールに関与するシステムには、おおまかにみるとこのような構成要素があります。ユーザに一番近いところにあるものが MUA で、その MUA の間で配送を司っているものが MTA です。MTA は、現在のインターネットの世界では、DNS を参照しながら配送を行っています。さらに mailbox から MUA に対してメールを渡す部分に、POP や IMAP などが出てきます。

MUA (Mail User Agent)

・ユーザ・アプリケーション

メールを読む

メールを書く

メールを保存/検索する

・UNIX

ucbmail, RMAIL, mush, MH (mh-e), mew,....

・Windows

OutLook, Netscape Mail, Eudora,....

MTA (Mail Transfer Agent)

・メールの受信

・配信先の決定

・メールの配信

リモートへ、ローカルへ、発信者へ（エラー）

・Store and Forward

一旦受信した後、次のホストへの転送を試みる

まずメールを受信し、次にどこに送るかを決定してそこへ投げる、という一連の動作をするのが MTA の仕事ということになります。メールの送り先は大きく分けると、「リモート、ローカル、発信者」の 3 つになります。インターネット電子メール配送のポイントは、Store and Forward ということで、エラーがあるということがその発信者がメールを出す瞬間には必ずしも分かるとは限らないということにあります。

MTA Programs

・sendmail　　http://www.sendmail.org/
・qmail　　http://www.qmail.org/
・SMAIL (GNU)
・MMDF (Multi-channel Memo Distribution, CSNET)
・exim　　http://www.exim.org/
・VMail　　http://wzv.win.tue.nl/vmail/
・LSMTP　　http://www.lsoft.com/LSMTP.html
・PP (X.400)

これらは主な MTA のプログラムですが、これ以降では「sendmail」あるいは「qmail」に関して説明します。

インターネットでのメールの送受信

・SMTP - Simple Mail Transfer Protocol
　RFC821(S)
・TCP のポート 25 番
・ほとんどの MTA は SMTP の実装をもつ
　DNS との連携機能をもつ

インターネットでのメール送受信の基本知識として、まず SMTP というものが使われるということがポイントとなります。SMTP というのは Simple Mail Transfer Protocol で、RFC821 で定義されています。この「(S)」というのはスタンダードということで、RFC はドラフトスタンダード、プロポーズドスタンダード、スタンダードなどと、その規格がどこまで実際に利用されているかによりレベルが設けられています。(S)というのが一番最終的に行き着くレベルとして広く使われている、つまりメールをやりとりするためには、この規格に従わないといけないというものになります。

TCP ポートの 25 番を使っていますので、たとえば telnet コマンドで 25 番ポートをアクセスすると、SMTP でお話ができますよ、ということになります。最近、インターネットというのが基本的な環境になっていますので、ほとんどの MTA は、SMTP の実装をその中に含んでいます。

SMTP の様子

実際に SMTP がどのようなものか、ということが示されています。データのやりとり、電子メールをやりとりするというのは、基本的にテキストで表現されているデータをやりとりするという形になっていますが、そこに単にデータそのものではなく、メールアドレスと呼ばれる情報が伴うわけです。

アンダーラインが引いてありますのが、クライアントから、つまり送る側から渡すデータで、アンダーラインのないものは、サーバ側からクライアント側に、つまりメールを受信する方からメールを送ろうとしている方に対して送り返すメッセージという形になります。

まず最初にサーバがグリーティングメッセージでこういうものを出し、それに対して、送る側が自分のホスト名を名乗る。これは冗長な情報のようにも見えますが、規格ではこういう HELO というのが必要になっています。

さらに、これが重要なんですが、自分の、発信者のメールアドレスをまずサーバに対して告げる。サーバはそのアドレスを受け入れましたよ、ということで ok と入れます。

その次に受信者、メールを受け取って欲しい人のメールアドレスをここで渡すわけです。そうすると、それに対しても ok と言われるので、さらに DATA というコマンドを送ってそのあとに、電子メールの本体が送られるという形になります。

電子メール自体いろんな表現形式がありますが、その最後を示す文字として、「.」のみの行というのを渡すことになっています。「.」のみの行というのが含まれたメールを渡すと問題が出そうですが、もともとのメールに含まれている「.」に関しては、さらにもうひとつ「.」をつけて 2 つにします。受け取った側で、最初の「.」を 1 つ捨てるという形にしています。

最後に転送が終了したら QUIT で両者の接続を切る、という流れが SMTP の基本的な通信になります。

インターネットでメールをやりとりする場合は、その MTA の間で、あるいは MUA から MTA に最初にメールを渡すときにも、このようなデータのやりとりが行われています。

インターネットにおけるメールの送信先の決定方法

・あて先メールアドレスのホスト名を抽出

user@host

・ホスト名から IP アドレスを取得

host → 12.34.56.78

/etc/hosts

NIS (YP)

DNS (Domain Name System)

ユーザはまずメールの頭で宛先を指定しますが、その宛先から実際にどのホストに送るのかを決定します。ユーザが web **や** ftp **のサービスを受けようとするときにも、ホスト名を指定する必要があり、**URL **の中にもホスト名が埋まっていたりしますが、メールの場合でも、だいたい同じような感じになっています。**

基本的には、user@host **という形で電子メールのアドレスを書きますが、この電子メールのアドレスの、@マークより後の部分、**host **という部分をもとにして、その電子メールの送り先、つまり先ほどの** SMTP **の接続先を決定するという形になります。そうしますと、そのホスト名から何らかの形で** IP **アドレスを導き出す仕組みが必要になります。このために** /etc/hosts **や** NIS **を使ういう時代がありましたが、ローカルな環境ではまだ使われています。インターネットのようなグローバルな環境では「**DNS**」というものを使って、ホスト名から** IP **アドレスへの変換を行うということになるわけです。**

DNS (Domain Name System)

・広域分散ディレクトリ・サービス

分散配置

分散管理

・ホスト名 → IP アドレス

・メールアドレス → ホスト名 → IP アドレス

同じドメイン空間を共用している

DNS **というのは、「広域分散ディレクトリ・サービス」とありますように、インターネットの中で各組織が個別に管理しているデータを、統一的にアクセスできるようなシステムです。そのシステムにより、ホスト名から** IP **アドレスが分かるような仕組みが提供されているわけですが、さらにメールでは、**MX **と呼ばれる情報を用いて、メールアドレスの@マークから後ろの部分を、**MX **の情報に従ってホスト名に変えて、さらに** IP **アドレスに変換するというように、処理がもう** 1 **段階加わっています。**

用語

ここで使っている用語を整理しておきます。

・配信

ローカルに配信 → メールボックス

リモートに配信 → 別の MTA へ渡す

ここでは「配信」という言葉を、受け取ったメールをどこかに渡すということのために使っています。

・転送: リモートに配信

・受理 (たぶん一般的な用語ではない): ローカルに配信

どこかに渡す対象として、自分のところのメールボックスに入れるという場合と、他の MTA に渡すという場合があるわけですが、それぞれに対して、配信というのを細かく分けています。たとえば「転送」というのは、リモートに配信することで、ローカルに配信するのは「受理」というように呼びます。

・受信: リモートから配信

どこに渡すかということにこだわらずに、他からメールをもらうという動作に関して「受信」という言葉を使っています。

メールアドレス

・発信者/受信者情報として利用

・ユーザ部 @ ドメイン部

motonori@wide.ad.jp

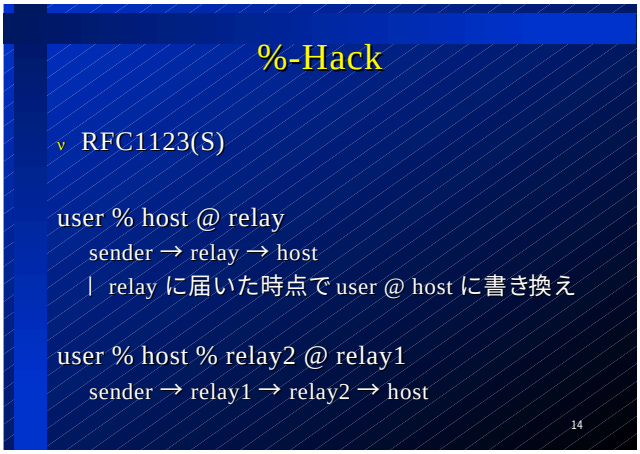
・その他の形式

%-Hack

Route Address

UUCP addressing

メールアドレスは、発信者あるいは受信者の識別情報として利用されるものです。最近では、こういう全世界共通の形式を使っています。ユーザ部というのがあり、@マークで区切って、ドメイン部というのが後ろにあるという形式、これをドメイン形式と呼んだりすることがありますが、これが一般的です。最近では、このような形の電子メールさえ扱えればよいという気がしますが、昔の時代は他にもいろいろな形式がありました。



%-Hack というのがありますが、これはドメイン形式が始まったあとで使われているもので、どこかに中継して欲しいホストを明示的に指定したい場合に使います。

@マークというものがメールアドレスに必ず 1 つである、ということは定かではないのですが、実際にはそのような形態で表現されています。そうしますと、電子メールのソフトウェアは、@マークより右側を注目して処理をします。ローカルに配信するようなとき、つまり@マークより右側が、そのホスト自身の中であるということになった場合に、初めてその@マークより左側を注目して処理をするわけです。

%-Hack というのは、@マークより左側のユーザの部分で、user%host という形式で別のホストへの転送を指示しているというものです。当然、@relay で対応するホストには、その%マークを@マークに読み変えて、次のホストに届けるという機能が組み込まれている必要があるわけです。

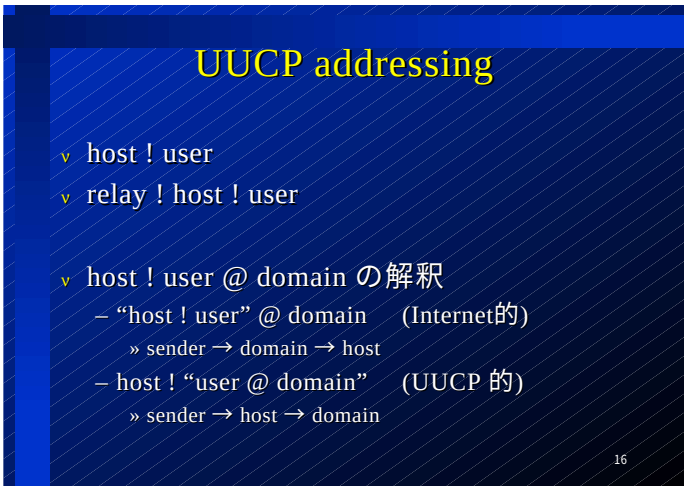
インターネットが昔、インターオペラビリティーがそれほど高くなく、どこに対してもワンホップでメールが送れるような状況ではなかった頃には、このようなアドレスを使って遠いところにメールを送っていた時期がありました。

最近ではそのようなことはなく、逆に SPAM のために悪用されたりする可能性がありますので、こういうものは逆に使えないようにしようという流れになってきているようです。

%-Hack は、正式には RFC などで定義されているものではなく、実際に、こういう使われ方をしていますよ、という程度の記述が載っているだけですが、Route Address に関しては、RFC822 とかいうアドレス形式が使えますよ、という形で書かれています。

Route Address がやっていることは、先ほどの%-Hack の場合と同じようなことで、@relay:というのがあったら、まずそこに送って、次にそこに届いたものは user@host というところに転送されます。これは%-Hack とは異なり、インターネットの電子メールのドメイン形式とは形式が大きく違いますので、これはこういう処理をちゃんとするような仕組みを、すべてのインターネットの MTA が持っている必要があるというわけです。

最後は UUCP 形式ですが、これは今となっては、あまりもう使われていません。「!」で区切る形式ですが、ネットニュースなどではヘッダのところに、似たようなものがありますが、これは UUCP addressing のときの名残りです。



コメント形式

・Full Name <user@domain>

・user@domain (Full Name)

・user(User Name)@domain(Company Name)

( ) のコメントはどこに入ってもよい

これは処理には関係ありませんが、電子メールのヘッダを処理するようなプログラム、メーリングリスト処理のようなものを書いたりするときにアドレス処理ルーチンが必要になりますが、そのときにどのような形の電子メールアドレスあるいはコメント形式を注意しないといけないかということが問題になります。RFC822 を見て頂ければ、どういう形式が許されているかというのが分かります。

ドメイン部

・Fully Qualified Domain Name
インターネットドメイン形式の完全なホスト/ドメイン名

・Fully Qualified Mail Address
user@mailhost.wide.ad.jp
user@mailhost **ではないことを意味する**

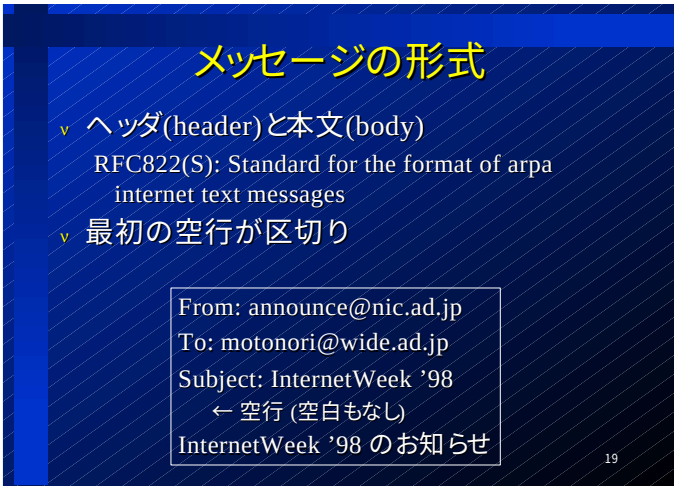
・Not Qualified Mail Address
user

・Generic Address
user@wide.ad.jp

"Fully Qualified Domain Name"**という表現が、電子メールやホスト名の説明のところに出てきたりします。こういうものは日本語訳が難しい表現ですが、たとえば「インターネットドメイン形式の完全なホスト/ドメイン名」という訳になります。**
user@mailhost **というのは実際にインターネットの形式では、正しくは、後ろに** wide.ad.jp **などがつきますが、省略できる場合があります。省略しない、最後の** jp **などのトップレベルまでしっかり書いたものを** FQDN **と呼びます。**
これは、メールアドレスに関してだけではなくて、他の ftp **や** web **などのサービス、あるいは単に組織内で使う便宜上の名前の書き方などでも出てくる一般的な話ですが、電子メールに限っていいますと、**Fully Qualified Mail Address **ということになります。**
つまり、@マークより後ろが、Fully Qualified **されているもの、ということです。@マークより後ろがあるというものは、**Qualify **されているといいますが、逆に** Qualify **されていないメールアドレスというのは、@マークより後ろがないものという形になります。このあたりは英語の文献などを読むときの必要な知識になると思います。**
Generic Address **というのは、ホスト名の部分が省略されているものです。**

電子メールのメッセージ

メッセージの形式



電子メールのメッセージの形式の説明をします。先ほどの SMTP の例でデータと呼んでいた部分は、電子メールの場合ではメッセージと呼ばれます。メッセージは、最初にくるヘッダとその後にくる本文の 2 つの部分から構成されます。一番最初に現れる空行、つまり改行だけの行がヘッダと本文の間の区切りになります。ヘッダの部分は配送の処理に関与することは少ないのですが、MUA が最初にメールを出すときに作った情報がヘッダに収められます。ユーザが書いた内容が本文に含まれるという形になります。

発信者と受信者

・発信者 (Sender)：1 人

　ヘッダの発信者は複数のことも：意味上の発信者

・受信者 (Recipient)：1 人または複数人

次に発信者と受信者ですが、これは自明だと思います。発信者というのは通常 1 人です。ただ、ヘッダに関しては意味的な発信者というのは書くことができますので、たとえば From という部分に複数の人を書くことも一応、許されています。それに対して受信者というのは、1 つのメールが複数の受信者に対して送られることがありますので、受信者というのは複数指定されることがあります。

ヘッダとエンベロープ

・封書に似ている

・エンベロープ (envelope)

　投函した人/届け先

　封書の表書きの送り主/宛先：実際に事務作業を行なう人

　配送の際に書き換えられていく

・RFC821(S): Simple Mail Transfer Protocol

　エンベロープはコマンドで指定

・UUCP

　エンベロープは rmail のコマンド・ラインに指定

ヘッダとエンベロープという概念について、整理しておきます。最初の SMTP のところにメールアドレスというのがありましたが、そこで渡しているものはエンベロープと呼ばれ

ます。それに対して先ほどのメッセージ形式のところで、アドレス、ヘッダというものがでて来ました。このように電子メールのアドレスには、ヘッダに出てくるアドレスとエンベロープの 2 種類がある、ということが MTA を考えるときの重要なポイントとなります。ヘッダとエンベロープの関係は、実際の郵便の封書に似ています。封筒の中に手紙が入っていて、内側の手紙の最初に意味的な受取人と発信者が書いてあります。さらにそれが封筒に入れられて、封筒の表書きには配送処理のために必要な実際の発信者と受信者が書いてあります。つまり、エンベロープというのは、実際の配送処理に必要なアドレスをやりとりするものです。エンベロープを使って実際の配送処理が行われますので、たとえば途中で書き換えられることがあります。たとえば実際の郵政省メールでも、転居の場合など、受取人のところにシールが貼られて、送付先が書き換えられていきます。それに対して、内側に入っている手紙の受信者という部分は書き換えられないということです。

・ヘッダ (header)
　本文を書いた人/読んで欲しい人
　内封された書面の送り主/宛て先
　基本的に書き換えられない
・ヘッダとエンベロープの送信者/受信者
　一緒の場合:　個人宛て
　異なる場合:　メーリングリストなど

SMTP では、"Mail From"や"Recieved To"などのコマンドでアドレスが渡されましたが, 他のシステムにおいてもヘッダとは区別して受け渡されるようになっています。先ほどの説明のとおり、ヘッダというのは実際に本文を書いた人、あるいは読んで欲しい人、という意味的なものを保持するもので、基本的に書き換えられないということです。電子メールの場合でも普通の郵便でもそうですが、エンベロープとヘッダの情報というのは、たいていの場合は一緒です。個人宛のメールのやりとりを考えた場合、そのヘッダの情報とエンベロープの情報は一致していますが、たとえばメーリングリストなどのように異なる場合もあります。

エンベロープはいつ作られるか

・ヘッダから抽出される
　送信する MUA が行う
　最初に処理を行なう MTA が行う
・エンベロープは配信処理で書き換えられる
　転送
　メーリング・リスト

エンベロープはいつつくられるのか、という疑問が出てきます。ユーザは普通、MUA に対

して、受け取って欲しいアドレスを書くわけですが、MUA で記述するメールアドレスは、ヘッダに書き込まれるアドレスという形になります。そして、MUA が MTA にメールを渡すときに、自動的にヘッダの内容をエンベロープにコピーする、という形でエンベロープが作られます。それに基づいて、MTA はエンベロープに対して必要であれば書き換えを行いながらメールを転送していくわけです。
基本的に最初のヘッダに書かれたアドレスは書き換えられないということになっており、書き換えの対象となるのはエンベロープ、つまり SMTP などでコマンドとして送られる封書の表書きに相当するもの、という形になります。

返信に利用するアドレス

・配信エラー通知の返送(自動)
　エンベロープの発信者
　Errors-To: ヘッダ　　；エンベロープの概念がないシステム用（まだあるの？）
・内容への返事(人が介在)
　ヘッダの発信者　　From:, Reply-To:, (To:, Cc:)

返信というものには少なくとも 2 通りあります。たとえばメールを受け取った人が返事を書く場合、さらにメールを送ったけれども途中でアドレスが間違っていて、Mailer Daemon とか、Postmaster とかからメールが送り返されてきた、という状況もあるわけです。これも当然、メールの返信に相当します。
実はそれぞれ見ているところが違います。配信エラーの場合では、エンベロープに書かれている発信者に対してメールが送り返される形になっています。MTA がずっとバケツリレーしている状況では、本文あるいはヘッダのところは基本的に参照しないということになっており、エンベロープでやりとりされる発信者の情報に対して返事を出す、エラーメールを送り返すという形になります。ただし、エンベロープが正しく使えなかった時代というのがあったようで、そのときに、Errors-To というヘッダを使って、メールのエラーが発生した場合のメールの返信先を記述できるような仕組みがつくられました。ただ、インターネットで使うものに関しては SMTP を使っている限り、そういうことはあり得ないので、Errors-To というのは、意味を持たないようになってきています。パソコン通信などの、SMTP とは違う別のメールシステムに対してメールを渡すときに、こういうヘッダがいまだに必要な場合というのがあります。
それに対して普通に返事を出すときは、エンベロープではなくて From や Reply-To、場合によっては Cc などに書かれているアドレスを使って返事をするということで、この場合はヘッダの発信者に対してメールを送り返すわけです。
ですから、こういうところの違いを認識して、うまくヘッダとエンベロープの設定を使いわけるということが必要になってきます。

メールボックスから MUA へ

・ローカル・メールボックス: UNIX など

・POP

・IMAP

メールがメールボックスに届いた後ですが、UNIX の場合はいきなりメールボックスからメールを取ってくるというのもありますし、POP や IMAP などを使う場合もあります。

メールの配信の 3 つのポイント

まず、メールの配信が具体的にどうなっているのか、あるいはメールのサーバを設定するときにどういうところに注意を払えばいいのか、という説明をします。

ポイントは 3 つありますが、1 番目は受信です。自分が注目しているある MTA に対して、メールを送ってきて欲しいわけですが、ただ単にメールサーバを立ち上げただけで届くわけではなくて、DNS に対して、そのホストのアドレスを登録する、つまりこれから使おうと思っているメールアドレスの@マークより右側の部分を DNS に登録することが必要になります。まずメールサーバを立ち上げると同時に、DNS に対してその情報を設定することで他からメールを初めて受け取ることができることになります。

2 番目は受理ということですが、どのメールアドレスで届いたものが自分宛かというのを判定するというのが、さらに別の処理として必要になるわけです。

自分宛ではない場合には、再び DNS を参照してそこにメールを送り出すという形になるわけですので、3 番目として、転送ということが必要になります。

メールを送ってもらうための設定

どうやって送り先を教えるか

・Internet

SMTP による直接配信

→ DNS に配信先を定義

・バケツリレー・システム

UUCP など(JUNET 時代)

→ 経路上の(すべての)ホストに配信先を設定

mailconf の活躍

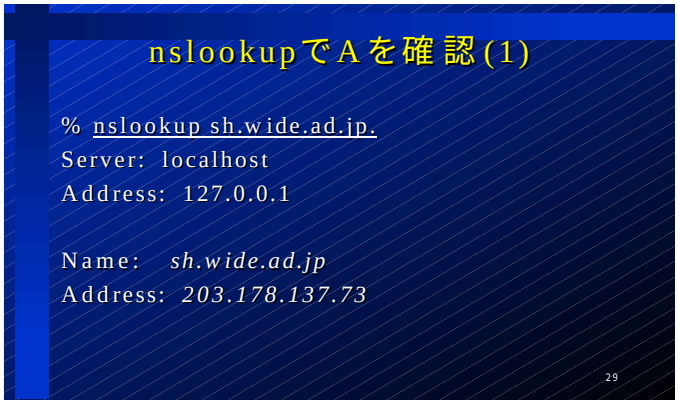
sendmail.cf 作成ツール

受信、メールを送ってもらうための設定ですが、これは自分が設定したメールサーバだけではなく全世界の他の MTA に対して、このメールアドレスはこのホストに送るということを教えるということが必要になり、これは主に DNS 設定のお話になります。
昔の UUCP でつくられていた、たとえば JUNET というのがありましたが、そのときは自分の周辺や隣接ホストだけでなく上流まですべてのホストに対して、このメールアドレスを見たら、こっちの方にメールを送ってね、というのを設定する必要がありました。そのときに mailconf のような sendmail.cf の作成ツールというもののが活躍したわけです。
今はそういうことは必要ではなく、DNS に対して設定をしておけば、正しくメールを受け取ることができます。

メール配信時に参照されるDNS のレコード

・A (Address) RR (Resource Record)
　ホスト名から IP アドレスを取得
・MX (Mail eXchanger) RR
　メールアドレスから配信先ホスト名を取得
・CNAME (Canonical NAME) RR
　ホストの別名を取得

メール配信時に参照される DNS のレコードとして、3 つのものがあります。A レコード、A というのは Address の頭文字です。ARR、RR というのは Resource Record の省略形ですが、ARR、A レコードといったりします。さらに MX レコードと CNAME レコードというものが、メールの配送の際に関与する、主な DNS のレコードとなります。



A というのは Address のレコードですが、nslookup などを使ってホスト名を指定してアドレスを引いてみると、このような結果が返ってきます。DNS を正しく設定していれば、世界各地の計算機の名前を指定すると、それに対応する IP アドレスが得られるわけです。自分の今立ち上げようとしているメールサーバに関しても、こういう形で名前からアドレスの対応づけが得られれば、それに対してメールが送られてくる、ということになります。自分の組織の中だけで引けるというファイアウォールの場合もありますが、インターネットからちゃんと見えるかどうかを確認するということも重要になります。

複数の IP アドレスを持つホスト

mail.x.co.jp　　IN　A　12.34.56.78

　　IN　A　12.34.54.32

・最初のアドレスへの配信に失敗した場合、順に全てのアドレスを試みる (実装依存)

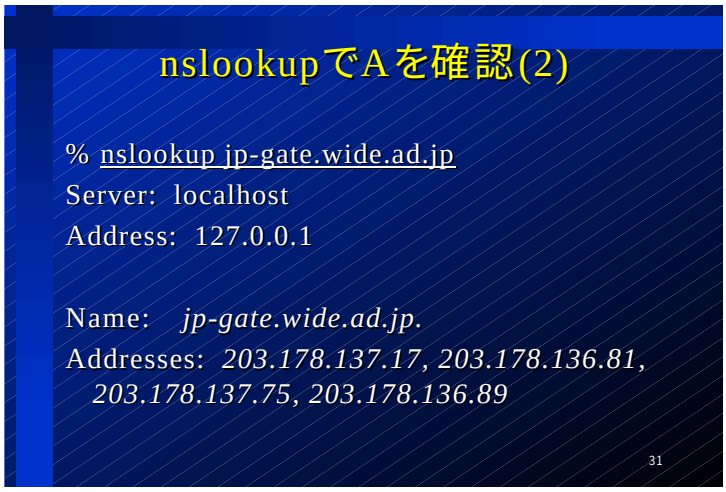
・DNS のラウンドロビン機能により、検索で得られる順序は毎回変わる

　負荷分散

　最初のアドレスしか試みない実装でも、そのうち届く(?)

DNS での IP アドレスの設定は、実際のところこのような感じで書くわけですが、そのときに、複数の IP アドレスを書くことができます。

そのときの nslookup はこのようになりますが、複数の IP アドレスを持っているホストがメールを受け取りたい場合には、ある 1 つのホスト名に対して、複数の IP アドレスを対応付けておくことになります。そうしますと、順番に IP アドレスをチェックして行き、アドレスが生きていれば、そこでメールを受け取ることができます。



Generic なメールアドレス

・ホスト名部分を持たない: ホストの改廃に依存しない

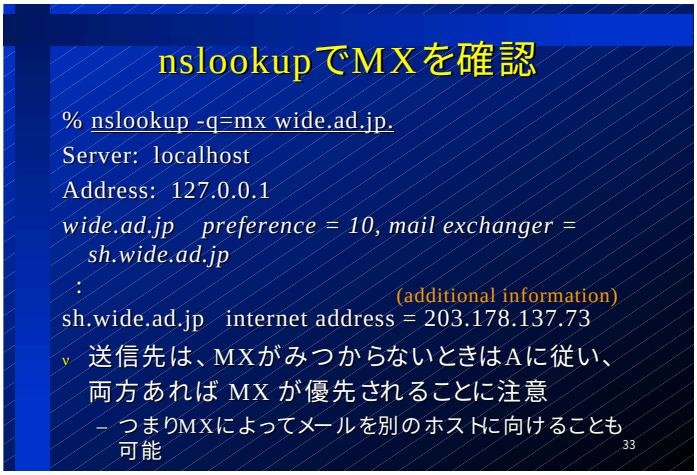
・MX (Mail eXchanger) RR を利用

・user@x.co.jp 宛てのメールは指定されたホストに送られてくる

　MX を引いて、得られた右辺のホスト名について、

　　さらに A を引き IP アドレスを取得

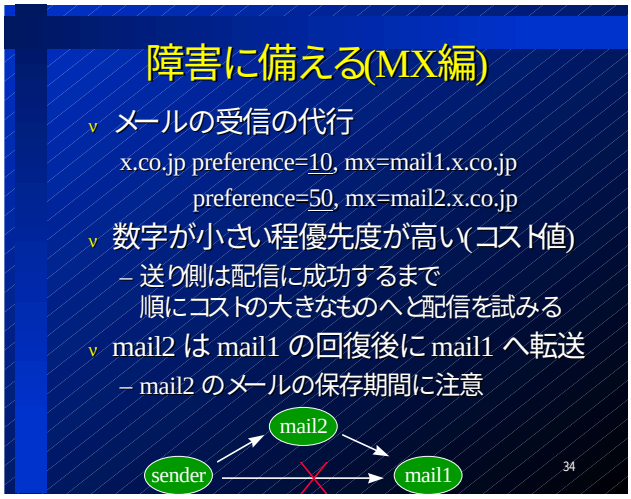
たとえば、x.co.jp というのは、x という名前の会社の Generic なアドレスです。アドレスとしてはこの前にホスト名が付くこともありますが、そのようなホスト名を隠して、会社のドメイン名だけでメールを受け取りたいというのが普通です。この場合、名前自体にアドレスを割り振るという方法もありますが、一般的には MX のレコードを定義するということになります。



たとえば wide.ad.jp という Generic なアドレスに対して-q=mx というオプションを指定して nslookup をかけると、mail exchanger=sh.wide.ad.jp という答えが返ってきます。

MX というのは mail exchanger の略で、MX レコードというのは、メールアドレスに対して、送って欲しいホスト名が定義されているものです。これはホスト名ですので、さらに IP アドレスが定義されているという形で、2 段階の参照関係になります。つまり、Generic なアドレスに対してまず MX としてホスト名が指定されており、そのホスト名に対して、さらに IP アドレスが定義されているというわけです。この 2 段階の参照をして初めて IP アドレスが得られて、メールを送るという処理に入ることができるわけです。

メールの送信では、MX のレコードが定義されているかを調べて、もし見つかればそれに対する IP アドレスを引きにいきます。もし MX が見つからないときは、A に従うという形になります。



障害に備える(MX 編)

インターネットでは、どこかで回線障害が発生して到達不能になっていることもありますし、当然、マシンが何らかの事故で落ちていたりするということもあります。その場合でも、メールの受け取り先として複数のホストを指定し、どれかに届けば目的地に届くということで、障害の発生に備えることができます。

sender というホストが mail1 にメールを送ろうとするときに、送り先を 2 つ登録しておくと、sender と mail1 の間に何らかの障害があったとしても mail2 経由で送ることができます。たとえばこの mail1 と mail2 が 1 つの会社の中にあり、インターネットとの接続点を、2 カ所のプロバイダから買っている、といった場合には、こういう形に当てはまるのではないかと思います。

そういうときに、まず最初に mail1 にメールを送って欲しいが、もしだめならば mail2 に送る、ということを DNS の MX を使って表現することが必要になります。

具体的には「preference」というものがポイントになります。たとえば MX を 2 つ指定した場合に、mail1 と mail2 に対する preference をそれぞれ 10 と 50 に設定します。preference というのは数字が小さいほど優先度が高いということになっていますので、まず数字の小さいところに対してトライして、だめならば大きい方に順に配信を試みるという形になります。

そうしますと sender としては、最初の MX が mail1 を向いているので mail1 に対して試みて、それでだめならば sender は mail2 に送り、mail2 は mail1 の復旧後に mail1 に送るという形になります。ただし、ひとつ注意する点としては、mail2 でどれだけの期間メールを保存するかということがあります。

メールは store and forward という形で 1 回受け取ったものを隣に送るのですが、送り先がずっと落ちていた場合には、そのメールがずっとたまってしまうわけです。そうすると、発信者からは、そのメールが届いたかどうかがよく分からないですし、mail2 の管理者も、メールのスプールがどんどん膨らんでいって困るわけです。ある一定の期間、たとえば 5 日とか 1 週間という期間、メールがずっと送れない状態が続いた場合は、そのメールを送り返すという仕組みが多くの MTA で実装されていますので、その保存期間を過ぎても mail1 が落ち続けていると、受け取るべきメールも送り返してしまいます。

たとえばお盆休みとか正月休みなどにメールサーバが止まりますよ、ということがあったりしますが、バックアップのメールサーバがある場合は、その期間を超えて保存できるようにメールの保存期間を設定するということが重要です

Lower MX の条件(メール・ループを防ぐための条件)

・MX RR の右辺にある自分の名前の認識

自分自身への接続の防止

sendmail -bt において $=w で確認

インタフェースのアドレス応じた名前は自動登録

qmail は IP アドレスで確認がおこなわれる

・自分の IP アドレスには接続にいかない

・自分の名前に対する MX RR のプレファレンス以上のコストの RR を捨てる

Lower MX 間でのピンポンの防止

先ほどの mail2 から mail1 に送るという状況で、mail2 がどういう動作をするべきかを考えてみます。mail2 では mail1 に対する MX が 2 つ得られるわけですが、このときに mail1 がだめだったとしますと 2 番目として mail2、自分自身が書かれています。本来の処理としては mail2 で貯めておいて欲しいわけですが、自分自身に対して SMTP を張りにいくと無限ループに陥ってしまいます。つまり、MX の名前が自分の名前でないことを判断することが重要になります。

first MX というふうにいいますが、最終目的地のメールサーバというのは、自分宛のメールが届いたらメールボックスに入れるだけですので、とくに難しいことはありませんが、最終目的地以外のサーバがその MX に書かれている場合は、lower MX の条件というのが出てきます。

その MX の右辺のところに出てくる名前が自分の名前かどうかを判断することで、おかしな挙動にならないように MTA としては工夫してあります。その工夫がちゃんと生きるかどうかというのは、さらに設定によるわけですので、そのへんの設定をしっかり間違わないようにしておくことが重要になります。

sendmail の場合はその名前を sendmail.cf に指定するということが必要です。qmail の場合は、さらに IP アドレスまで調べに行って自分のインターフェイスの IP アドレスと、MX の右辺のホストに対応する IP アドレスが等しいかどうかをチェックするようになっていますので、qmail の方ではエラーになることはありません。

今の話は 2 つあった場合の 2 番目の話ですが、さらに MX が 3 つ以上あった場合、自分がたとえば 2 番目の MX だったときに自分の名前を認識して 2 番目には送らないということが行えたとしても、さらに 3 番目があると思って 3 番目にメールを送ってしまうと、2 番目と 3 番目の間でピンポンしてしまうということになります。

MTA を一から書こうと思っている人は、このあたりを注意しないといけませんが、できあいの MTA に関しては、自分の名前が右辺に表れた場合は、それより preference の大きい MX はすべて捨てるという処理をしています。

負荷分散

x.co.jp preference=10, mx=mail1.x.co.jp.
preference=10, mx=mail2.x.co.jp.

・同じコストの場合は送り側が乱数で選ぶ

・最終的に一つのメールボックスへ

受信側に仕掛けが必要: 静的な配送定義など

普通は preference **を違うようにしておくことが多いのですが、同じにしておくと乱数で選びます。最終的に同じメールボックスに届くようにするには、受信側での対処が必要ですが、受信用のメールサーバを複数持っておきたいという場合は、**MX **をこのように設定することで負荷分散が実現できます。**

送ってもらったメールの受理

・届いたメールを自分宛てとして認識

ローカルに配信(受理)

「送られてきた = 自分あてである」ではない

・自分宛でないと判断した場合

転送先を探す

送ってもらったメールを受理するために、何をする必要があるかというと、つまりは、自分宛かどうかを判断することです。これはメールサーバの設定によって、いろいろなやり方がありますが、メールサーバを設定するときに最初に指定すべき項目の 1 **つであるわけです。**

受理するアドレスの設定

・Sendmail (CF)

ACCEPT_ADDRS **に定義**

・qmail

/var/qmail/control/locals **に定義**

たとえば sendmail **の場合は(**CF**)に** ACCEPT_ADDER **というところがあり、そこに書くことになります。**qmail **の場合は** locals **というところに、そのアドレスを書けば、それは自分のメールボックスに送るべきアドレスとなるわけです。自分の受け取りたいメールアドレスを、そのメールのソフトウェアに設定するだけで、それ以上の難しいことは何もないと思います。**

受信設定のまとめ

・相手に送り先を教えること

MX **レコードを定義**

・自分宛てだと解釈すること
　ローカルへの配信 (受理)
・別個に設定が必要
受信と送信という区別をするとすれば、ここまでが受信設定に対応します。何が重要だったかといいますと、まず DNS に対して自分のアドレスをちゃんと教えることと、さらに受理ということに関して、その名前がちゃんと自分のアドレスであることをソフトウェアに教えることです。。この 2 つは別の設定ですのでちゃんと区別する、いうことがポイントです。

メールの配信設定

配信方法のバリエーション

・DNS の MX RR 参照による配信
　MX を参照する MTA の準備
・ホスト名のみによる配送
・固定ルールによる配信
　DNS を参照する必要性の検討

次は 3 番目のメールの配信です。MTA からメールを送り出すときに何が重要かということですが、まず、DNS を参照してメールを出すということになります。ファイアウォールの中などインターネットと直接通信しないようなメールサーバの場合には、またいろいろと設定が変わってきますが、基本的な設定としては、DNS を参照するというところがポイントとなります。

DNS 参照のための基本設定

・/etc/resolv.conf
・サービススイッチファイル

UNIX の場合は、とにかく DNS が引けるようにならないといけないわけです。DNS を参照して送るためにはまず、resolv.conf というのが第 1 のポイントです。最近 Solaris などでは、もう少し複雑な設定ができるようになっており、サービススイッチファイルという別ファイルの設定も必要になっています。ホスト情報には、たとえば/etc/hosts や NIS などもありますが、これらをどのような順番で検索するかなどを設定できるようになっています。

/etc/resolv.conf

・ネームサーバの指定

nameserver 0.0.0.0　(localhost - 127.0.0.1 **と解釈される**)

nameserver 12.34.56.78

nameserver 12.34.56.79

　　3 **つまで** (MAXNS in resolv.h): **いくつでもタイムアウト時間は変わらない** (75s)

domain sub.x.co.jp

search sub1.x.co.jp sub2.x.co.jp x.co.jp

　アドレス補完の際に利用

まず resolv.conf **ですが、**nameserver **という記述でネームサーバの** IP **アドレスを指定します。アドレスを何個書けるかはコンパイル時の定数によって決まります。**

さらにアドレスの補完をさせたい場合、FQDN **ではなくドメインを一部省略した形でのメールの送受信をやりたい場合には** domain**、**search **を設定する必要があります。また** host1 **から** host1.x.co.jp **のように静的に定義するということも可能です。**

サービススイッチファイル

・Solaris

/etc/nsswitch.conf

　　hosts: files dns

・DEC

/etc/svc.conf

・その他

ServiceSwitchFile **オプション** (sendmail.cf)

デフォルト: /etc/service.switch

　hosts　　dns files nis

Solaris **の場合は**/etc/nsswitch.conf **で、その中に**"hosts: files dns"**というように** dns **を書いておかないと、**sendmail **からメールを送ろうとした場合にうまくいかないことがあります。これは付属の** resolver **を使うかどうかにも依存するので、**bind 8 **などを一から入れたりした場合などは自分の環境と使っているソフトウェアの組み合わせから整理して判断することが必要です。**

DEC **の場合は**/etc/svc.conf **というものになりますし、さらにサービススイッチファイルという考え方がだんだん広まってきており** sendmail **自体でも**/etc/service.switch **というものが設定できるようになってきています。たとえば**"hosts dns files nis"**と書けば、**DNS **で見つからなかったら次に** files**、つまり**/etc/hosts **とかを見に行くということができるようになっています。**

DNS の MX を参照する場合

・MX を参照する MTA

sendmail.mx

libresolv.a のリンク

MX 参照用 sendmail.cf

MX_SENDMAIL=yes (CF)

(実際には Wildcard MX 対策だけ)

→ アドレスの補完

MX を参照する MTA という話ですが、sendmail に関しては、たとえば Sun の場合では参照しないものと参照するものという 2 種類が提供され、sendmail.mx という名前のバイナリがついています。しかし、最近、OS についている sendmail というのはあまり信頼されておらず、sendmail のソースからコンパイルして使うことが多いようです。

固定ルールによる配信

・sendmail.cf に固定ルールを書く

mailconf

CF

STATIC_ROUTE_FILE

DNS を参照するのではなく、固定ルールによる配信をしたい場合、たとえばファイアウォールの中などで固定ルールを参照して配信したいという場合は、このように設定をすればいいわけです。

配信の動作確認

・アドレスの解釈が正しいか

sendmail -bv あるいは sendmail -bt の /parse

・MX が正常に検索できているか

sendmail -bt で /mx コマンド

・実際に送ることができるか

sendmail -v

配信に関しては、このようにして実際にメールが出ていくかどうかをチェックするということで、設定がうまくいっているかどうかを確認できます。sendmail の場合はテストをする方法がいろいろ用意されています。たとえば bv や-bt のところで/parse などを用いればアドレスの処理が正しくできているか、MX が引けるかどうかとかというようなことを、送ることなしにテストできます。さらに実際に送ってみるというのが何よりも確実ですので、sendmail の場合は-v などをつけて送るということになります。

配信設定のまとめ

・ホストが DNS **を参照できるようにする**

resolv.conf

サービス・スイッチ・ファイル

・メールアドレスごとの配信先を考える

DNS (MX) **を参照してそのまま配信する**

どのネームサーバを見るか (後述)

静的に配信先を設定する

配信設定のまとめとしては、ホストが DNS **を参照できるようにするというのが、インターネットでメールを送るときの一番のポイントです。そのときに重要なことは** resolv.conf **と、よく忘れがちなのがサービススイッチファイルの設定です。**

さらにもう少し複雑なことをしようとすると、たとえば、このドメインに関しては DNS **を見たいとかこの部分に関しては静的に送りたいなどという組み合わせが出てきます。そういう場合は、しっかりとその概念や仕様を整理して、設定に反映するということが重要になります。**

2. DNS の仕組みと管理

ドメインとゾーン、

DNS (Domain Name System)

・広域分散データベース

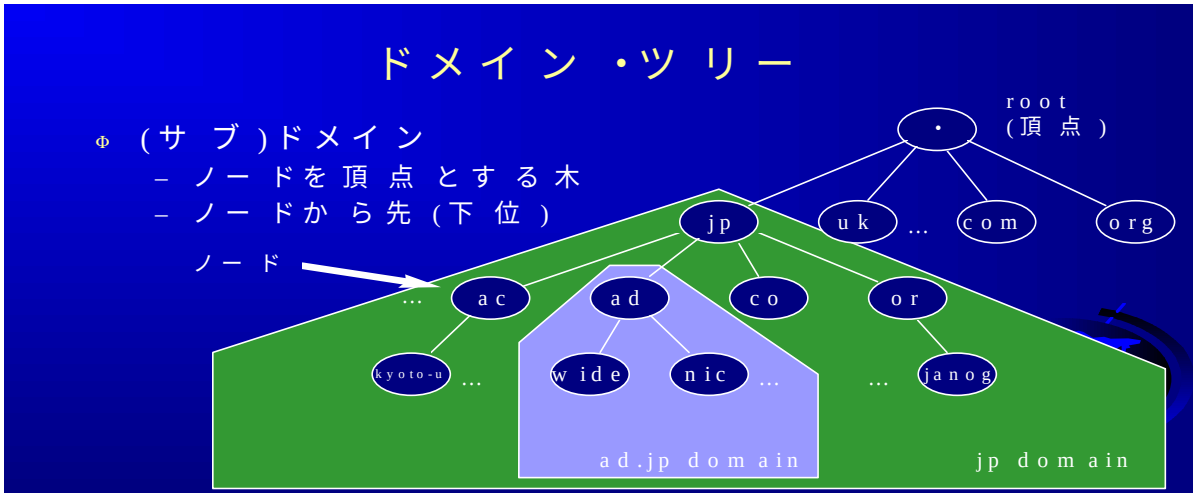
・ホスト名と IP アドレスの対応表

・組織ごとに自主管理

大昔は /etc/hosts で集中管理

DNS は広域的なひとつのデータベースとして見えますが、複数のサーバで分散的に管理しているというものです。昔は/etc/hosts という 1 つのファイルにホストを書いて定期的に ftp で配布することでネットワークにつながっているホストを扱っていたわけですが、それでは破綻をきたしてきたので DNS が出てきたという背景があります。

ドメイン・ツリー



ドメイン・ツリーというのがまずありますが、ドメイン名というのは、アルファベットの並びをピリオドで区切ってできています。それぞれの部分をドメインというわけです。たとえば日本の場合、jp ドメイン、ac ドメイン、kyoto-u ドメイン、wide などそれぞれのピリオドの区切りより左側、内側に入るところがドメインです。インターネット全域のドメインというのは root というところを頂点としています。その中で jp ドメインは、jp を頂点としたツリー、こういう領域が jp ドメインという形になるわけです。さらに ad ドメインというのが jp の下にあってこういう形で含まれているわけです。

分散管理と検索

・必要に応じてノード間の上下リンクで分割

ノードの下流へのリンク: Delegation(**権限委譲**)

TOP domain, 2nd(3rd)-level domain: NIC **が管理**

・単方向リンク(上から下へ)

上位へは root **まで戻ってから辿る**

全サーバは root **を知っている**

こういう形でドメインの階層関係があるわけですが、実際にネームサーバで管理しているものは、こういう感じとは正確には少し違います。

まず分散管理あるいは検索をするうえで重要なものとして、リンク関係というのがあるわけです。ドメインの先ほどの絵を見ると、jp **の下に** ad**、**ac**、**kyoto-u**、**wide **などがそれぞれつながり、それぞれのノードに対応するネームサーバが存在すると考えるのが直感的だと思いますが、正確には微妙に違うわけです。**

そういうリンクがある場合に、自由に関係ない人がいろいろ、そういう管理に割り込んできてもらうと困りますので、ちゃんと上下関係が対応づけられている。上下関係といっても、正確には上から下への関連づけです。つまり、この DNS **というシステムは、頂点から権限委譲、**Delegation **して、上から下に権威づけをしていくという形でできています。そのような上から下へのリンクになっていまして、検索のときも同じように、上から** jp**、**ad**、**wide **というように順番に行くという形になっています。**

このへんは、すべてソフトウェア、resolver **などの中で自動的に行われますので、ユーザが意識する必要はほとんどありません。**

ゾーンとドメイン

・必ずしもノード単位で分割管理する必要なし

・ゾーン

共同管理される隣接ノードの集合

必ずしもドメインとは一致しない: 1 **ゾーン内に複数サブドメインを定義可能**

データの管理単位

部所単位/地域単位に分割

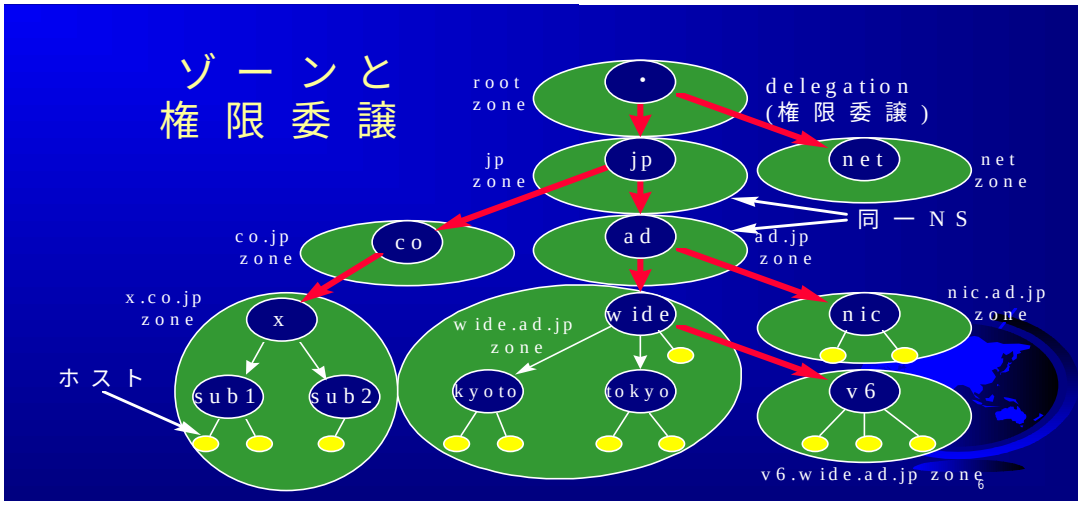
1 **つのネームサーバに対応**

末端ではドメインと一致

ドメインは、上から下に対して権威づけが行われているわけですが、必ずしもドメインのノードという管理で管理されているわけではない、ということです。ゾーンというのは、共同管理されている隣接ノードの集合です。先ほどのドメインツリーで出てくるノードの集まりではあるんですが、そのノードの隣接している部分の集合でゾーンというのがでてきています。たとえば、このゾーンはドメインと一致することもありますが、必ずしもド

メインと一致するわけではありません。
管理を考えたときに、そのドメインの中で一部分だけを管理したいとか、地域ごとに分けるなど、管理する単位を分割したいということがありますので、管理の単位に対応づけたものがゾーンであるということになります。

ゾーンと権限委譲



末端に関しては、普通はそのゾーンとドメインというのは一致します。ドメインというのは、あるノードを頂点とした、それより下の部分全体を指す言葉ですが、ゾーンというのはノードの集まりで、たとえば root のゾーンというのがあり、root だけによって構成されている部分になります。root の部分というのは、つまりトップレベルのドメインを管理しているサーバはどれですよ、というのを列挙しているデータになるわけですが、その部分だけを管理しているところがあるわけです。

そこから権限委譲が行われまして、たとえば日本であれば JPNIC とそれに協力している組織の持っているサーバが正しい情報を持っているということで、root のところからは、そのサーバを指し示している、という形になります。jp の中は、jp とさらにそのひとつ下のドメイン、ad や co などは全部、その JPNIC が管理していますが、ゾーン的には別になっています。

ゾーンが別であっても、「同一 NS」、同じネームサーバで複数のゾーンを管理することがあります。たとえば x.co.jp という組織は、co のゾーンの中から権限委譲を受けており、1 つのネームサーバですべてまかなっているとすれば、その部分に関しては x というドメインの中に入っているすべては x のゾーンの中で管理されているという形になります。また別の例として、wide というゾーンには v6 という実験関係のドメインがありますが、wide 全体の管理を離れて v6 のグループだけで管理したいということで v6 のゾーンとそれ以外の wide のゾーンという関係でドメインを分けて、権限委譲をしているという形になります。ですから、x.co.jp のゾーンは、x のネームサーバが受け持っていますし、wide は wide のネームサーバが受け持っています。ただし同じネームサーバが複数のゾーンを受け持っていることもあるということです。

以上のような形でドメインとゾーンは区別されています。

サーバの種類

・サービスの種類で分類

データ提供用 (検索もする) / 検索専用

・データ(ゾーン)の管理方法で分類

そこで編集(Primary) / **他からコピー**(Secondary)

・権限で分類

Authorized / Unauthorized

・サービス対象で分類

組織外向け / 組織内向け

情報を提供するサーバ以外にも、いろいろな他のサーバがありますので、サーバというものを分類してみます。

先ほどのゾーンを受け持っているサーバと、そうでないサーバという分類があります。1 **つのサーバだけですと障害が発生したときにデータが提供できなくなりますので、同じゾーンをサービスするサーバを複数用意しておくというのが、ネームサーバの管理においても重要なポイントになっきます。複数のサーバが存在するわけですが、それらは平等ではなく、その中の** 1 **つは** primary **と呼ばれ、それがそのゾーンのマスター的な管理をします。その他のものは、その** primary **からデータをコピーして、他のところにサービスします。このような管理の方法に関する分類というものがあります。さらにデータを提供するといった場合も、権限委譲というのがありましたが、それをちゃんと受けているかどうかという関係において区別がありますし、さらに組織外向けと組織内向けといった区別もあります。**

提供するデータ(ゾーン)の管理

・プライマリ(マスタ)・サーバ

データベース・ファイルの編集を行なう

・セカンダリ(スレーブ)・サーバ

プライマリのサービス・バックアップ

プライマリ・サーバからデータをコピー: 別のセカンダリからでも一応可

コピーのチェイン: コピー元サーバを複数指定可能

同時に到達不能にならない場所に配置

提供するデータ、ゾーンの管理についていいますと、サーバを区別する用語としてプライマリ(マスタ)・サーバというものがあります。これは bind 8 **のドキュメントにも書かれていたと思いますが、データベース・ファイルの編集をそこでするというものです。つまり管理という視点から見たときに、メインとなるサーバがプライマリのサーバということになります。それに対してプライマリからデータをコピーしてきて、バックアップという機能を果たすのが、セカンダリ(スレーブ)・サーバということになります。**

データは必ずしもプライマリからコピーしなくてはいけないというわけではなく、セカンダリから再び孫的な感じでチェインをすることもできるようにはなっています。これはまだインターネットの回線が細かった時代の話で、最近のインターネットの回線の環境では、あまりそういうことを考える必要もなくなってきたように思います。あるいは、コピー元のサーバは複数指定できますので、これで対応することも可能です。
さらにポイントとしては複数のサーバを配置するときには、ネットワーク的にどこかが故障したときなどに、同時に到達不能になったりしないような形で配置することが重要になります。

・検索要求は平等に来る
　プライマリ・セカンダリの区別はない
・ゾーンに対する区別
　一つのサーバで複数のゾーンを管理
　　ゾーン A に対してはプライマリ
　　ゾーン B に対してはセカンダリ
　サーバ個体に対する区別ではない

プライマリとセカンダリということで、メインとサブというような表現を使っていますが、これは管理上での違いです。プライマリとセカンダリのサーバ自身の間では、そういう区別はあるわけですが、検索を要求しているところから見た場合は、プライマリ、セカンダリというような区別は見えませんので、検索要求としては平等に行われます。
プライマリ、セカンダリというのは、サーバに対する概念ではなくて、ゾーンに対する概念です。ですから、複数のゾーンを 1 つのネームサーバで管理しているときに、ゾーン A に対しては、そのネームサーバはプライマリかもしれないけれど、ゾーン B という別のゾーンに対しては、そのネームサーバはセカンダリかもしれません。つまり複数のゾーンを 1 つのネームサーバが管理している場合には、プライマリ、セカンダリというのはゾーンごとにありますから、サーバがプライマリであるというような表現概念はないということです。

データの提供に関する権限

・Authorized Server

データをインターネットに提供

上位ゾーンからのリンク(権限委譲)がある

・Unauthorized Server

手元の恒常的キャッシュ

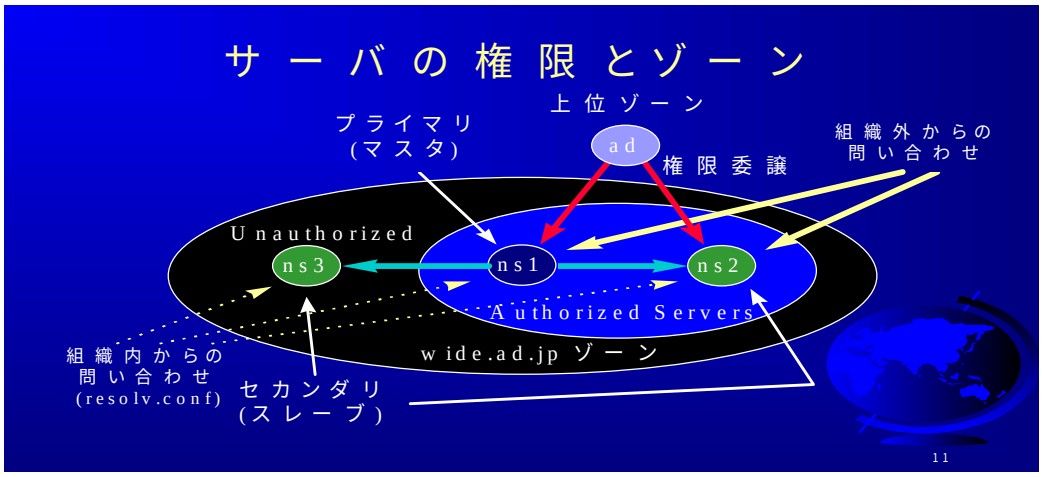
データを近隣クライアントに提供

上位ゾーンからのリンク(権限委譲)がない

・ゾーンに対する区別

プライマリ、セカンダリは、外から見たときには平等に見えるということですが、平等というのは権限委譲という点に関してです。そういう権限委譲の関係を見て検索を行うわけですので、プライマリ、セカンダリというような概念とは別に、権限があるかどうかということが独立な概念として出てきます。

サーバの権限とゾーン



プライマリとセカンダリというのは、ゾーンを管理しているサーバの間だけでの関係になります。ここではサーバが ns1、ns2、ns3 **と** 3 **つあり、そのうちの** ns1 **をマスター、プライマリとして定義しています。そうしますと、そこからデータをコピーしてくるのはセカンダリということになります。**

これはサーバの間での関係づけでしかないわけですが、それに対してその上位ゾーンからの権限委譲をというものがあります。たとえばこれは wide.ad.jp **のゾーンのサンプルになっていますが、**wide.ad.jp **というゾーンを管理しているネームサーバは** ns1 **と** ns2 **であるということが、上の** ad **のゾーンで定義されることにより、この** ns1 **と** ns2 **が、**Authorized Servers**、権限を持っているサーバであるということになります。そうしますと、**ns3 **は** ns1 **からデータをもらっているセカンダリのサーバではあるのですが、権限委譲がないので、**ns3 **に対して外からの検索が行われることはありません。**

このような authoraized と unauthoraized の関係がありますが、unauthoraized なものも、セカンダリのネームサーバという位置づけにはなりますが、セカンダリといういい方はせずに、キャッシュサーバと呼んでいます。
管理する側としては、プライマリとセカンダリといった違い、あるいは上からの権限づけの部分というあたりを区別して、それぞれのサーバの位置づけを把握すればよいと思います。

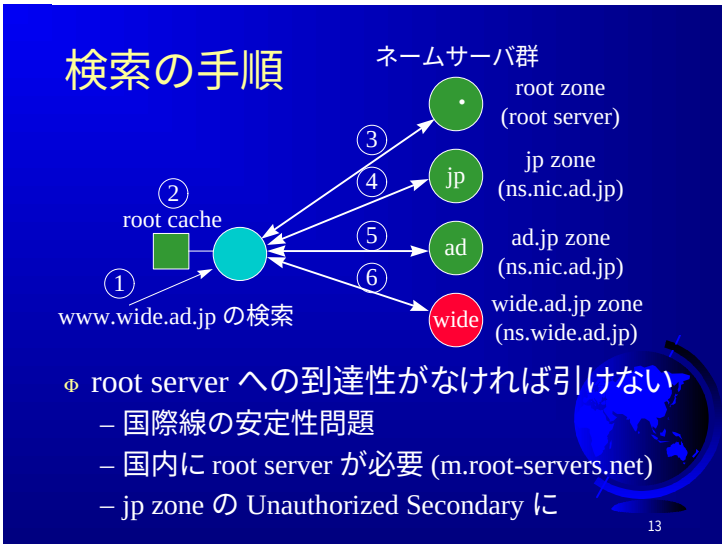
検索専用

・キャッシュサーバ
　一度検索したデータをしばらく記憶: ２度目以降は Unauthoritative Answer として応答
　プライマリでもセカンダリでもない: どのゾーンに対しても
・参考：ネガティブ・キャッシュ
　該当レコードが存在しなかったことを保持(全サーバ)

検索専用と書いてありますが、何もデータを持っていないし権限委譲も受けていないサーバというのを、キャッシュサーバといいます。ネームサーバの仕組みとして、1 回検索したデータというのは一定期間覚えておきます。そうしますと、組織から外への検索は 1 回だけですむことになるわけです。そのような無駄な DNS の検索を削減するために、キャッシュサーバを立ち上げることもあります。
参考として、ネガティブキャッシュというのもあり、存在しないメールアドレスとか存在しない URL などを引きに行ったりした場合にも、何度も同じもの、存在しないものを引きに行ってしまうと、当然、同じ検索が何回も行われることになります。これを抑えるようなネガティブキャッシュという仕組みも用意されています。
また何かのゾーンを管理している普通のサーバの場合でもキャッシュサーバ同様に、1 回検索したデータや存在しなかったデータもしばらく覚えておくというような機能を持っています。

検索の手順

たとえば www.wide.ad.jp というのを検索しようと思った場合には、まず 1 番目に外部から検索の要求が来ます。検索というのは権限委譲の矢印に従って行われますので。まず最初に root が必要になります。それぞれのネームサーバには root キャッシュというものを設定しておくことになっていますので、2 番目として root キャッシュを調べます。そして 3 番目として root を受け持



っているサーバに聞きに行きます。今度はこれに対して、jp のサーバがどこにあるかを聞くわけです。そうすると root のサーバは、jp サーバを教えてくれます。そうすると 4 番目として、jp サーバは ad を返してくれます。それによって 5 番目として、ad のサーバが wide を返してくれます。今度は wide のサーバに対して、www がどれかを聞きに行くわけです。このような形で、順番にドメインの階層構造をたどって下りてくることによって、検索が実現されということになります。そうしますと www.wide.ad.jp というのを検索しようと思うと、手元のネームサーバは当然のことながら、root と jp と ad と wide という、一連のネームサーバすべてに到達できないといけないということになります。検索の仕方によっては、最初に聞いたサーバが他のところも全部調べて答えを返してくれるというようなこともありますが、root とか jp とか、上の階層のドメインというのは、必ず検索が最初に飛んできますから、次はどこというような情報だけを返すサーバとして設定されていることが多いわけです。そうすると、すべてのサーバが見えていないといけないということになるわけですが、以前は、root サーバというのは海外にしかなかったわけで、そうすると、海外のリンクが事故とかで落ちていたときに、root サーバに到達できないので、その下のさまざまなドメインが検索できないという状況が発生することがありました。
最近は、m という root サーバが国内にもできましたので、それで海外線事故で落ちていても大丈夫という状況になっています。

DNS Servers

・Berkeley Internet Name Domain (BIND) Server

bind 4.9.7

bind 8.1.2

できるだけ最新版を

セキュリティ、パフォーマンス、信頼性、新機能

http://www.isc.org/bind.html

・Windows NT のネームサーバなど

信頼性は大丈夫(?)

実際にどういうサーバが使われているのかということですが、普通は BIND というものが使われていると思います。この最新版としては、4.9.7 あるいは 8.1.2 が使われています。bind 8 の方が機能的にどんどん拡張されて、いろんな機能が使えるようになってきていますので、最近はもう 4.9.x などを使うような理由は特になくなってきています。入手するには、この URL のところを見て頂ければいいと思います。Windoes NT など他の OS にもネームサーバがありますが、ちょっと不安があるとか、うまくいかないとかという噂もいろいろ聞いたりしますので、インターネットで安定性を求めるなら、UNIX 上で普通の bind を動かした方がよいのではないかと思います。

サーバの設定

サーバの設定ファイル

・/etc/named.boot (bind 4)

・/etc/named.conf (bind 8)

named-bootconf.pl **でフォーマット変換**

named.boot **から** named.conf **に**

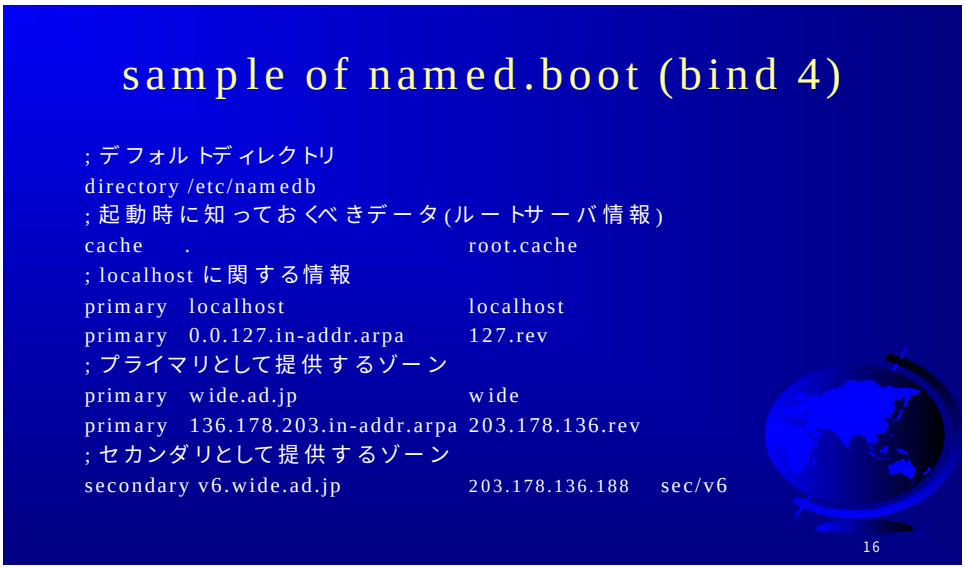
bind 8 **に添付**

BIND **では** '; ' **がコメントの開始**

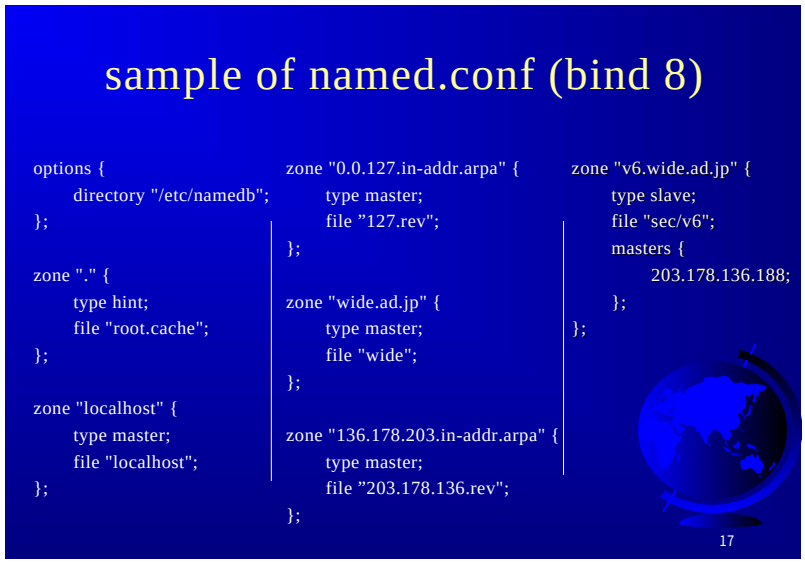
サーバの設定ファイルですが、ネームサーバに関していいますと、/etc/named.boot **とか** /etc/named.conf **などというファイルがあります。これはネームサーバのバージョンが変わったことによってファイルの形式が変わっていますので注意してください。**

bind 8 **には、**named-bootconf.pl **という古い** bind 4 **の形式から、**bind 8 **の形式に変換してくれるツールがついていますので、**bind 8 **をインストールしようという人には、わりと便利なツールです。**

BIND **では、「**;**」がコメントの開始であるといった基本的な情報があり、それに基づいて、こういうふうにコメントを入れてありますが、**named.boot **の書き方としては、これが典型的な書き方になります。**



これはbind 8のnamed.confの例です。すべてのネームサーバはルートサーバの情報を持っていないといけないということがありましたが、このようなtype hintという記述、ヒント情報として、ルートサーバの情報が入ったファイルを指定することになります。そうしますとここに書かれている情報に基づいて、まず最初のルートサーバのところに検索が飛ぶというわけです。



それ以外の自分のサーバで管理しているゾーンを、適当に定義していくわけですが、まずローカルホストというそのホスト自身を表す名前、そしてIPアドレスというものがあります。これらはすべての計算機で使われるものですから、すべてのネームサーバに定義しておくべきです。定義していなくても他のサーバから引いてくることはできますが、無駄なトラフィックが発生してしまいます。

まず、localhostというゾーンと、0.0.127.in-addr.arpaという、逆引き用のゾーンがあります。逆引きとは、IPアドレスからホスト名を検索するという逆方向の参照のことです。このようなlocalに関連するものに加えて、さらにwideに関するゾーン、あるいは、そのwideの持っているIPアドレスに関する逆引きのためのゾーンが定義してあります。

先ほど説明したネームサーバのそれぞれの種類や関係付けを整理して、それをそのままnamed.confに反映するという形になっています。

root cache

・ルートサーバに関する情報

　ルートサーバさえ知れば全てを検索可能

・ftp://ftp.rs.internic.net/domain/named.root

・13番目が日本で稼働開始(1997/8)

　m.root-servers.net

・Firewallの内側では

　内部向けroot serverを用意

　Forwardersでがんばる

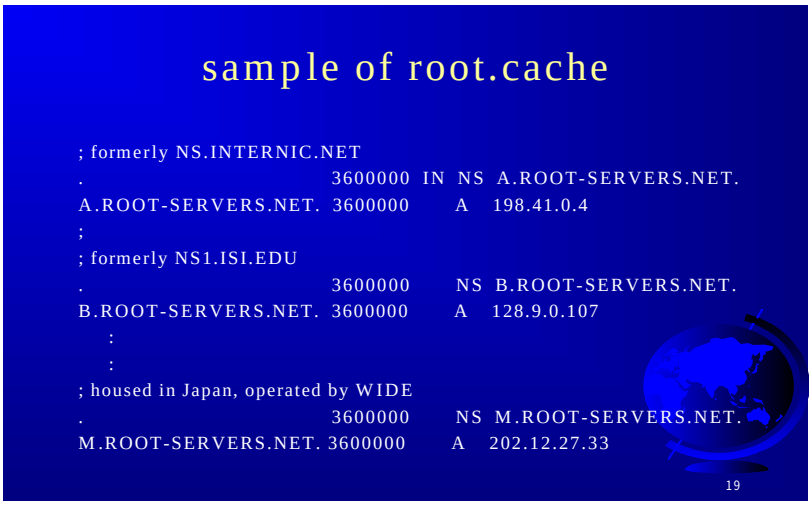
繰り返しになりますがルートキャッシュというのがまず重要です。バインドのパッケージの中にも入っていますが、最新のものはこのURLの場所に置かれています。13番目のネームサーバが日本で去年の8月に稼働を開始しました。

インターネットを直接参照できる環境であれば、それだけを設定すればよいのですが、

Firewall の中でネームサーバを立ち上げるときには自分の Firewall の内側で、同じようなドメイン階層を作っておかないといけません。そうすると内側向け用のルートサーバなどが必要になりますし、内側向け用のルートキャッシュのようなヒントファイルを用意するということになります。あるいは、その Firewall の中から外のデータも見たい fowarders という設定をすることで、内側から外のサーバを参照して見ることができます。

sample of root.cache

ルートキャッシュの例ですが、名前、数字があって、それからインターネットドメインという意味の IN というのがありますが、省略されている場合もあります。その次に、NS や A というのがあり、次に IP アドレスとか、あるいはサーバの名前などが書かれています。ネームサーバのデータというのは基本的に行単位のデータになります。



forwarders

・組織内から外部のアドレスの問い合わせ

外部のネームサーバに問い合わせを転送: socks 対応 firewall などの場合

slave とともに指定

forwarders 12.34.56.79 (内外両側からアクセス可能なサーバ)

slave (options forward-only - 4.9.3 or later)

・キャッシュの有効利用

データを特定のサーバに集約

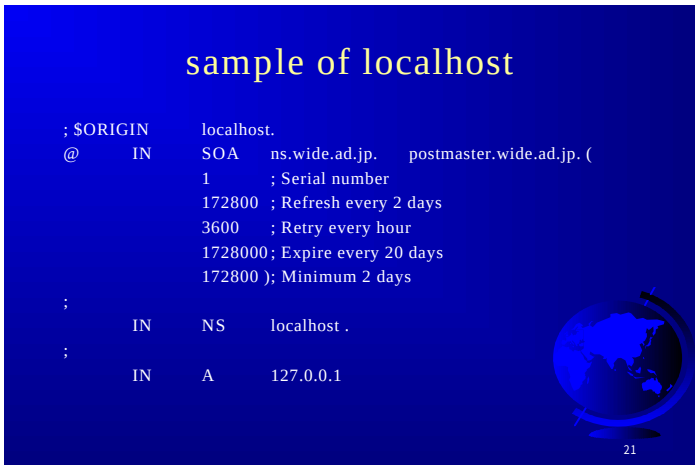
トラフィックを抑える: 回線が細いときなど

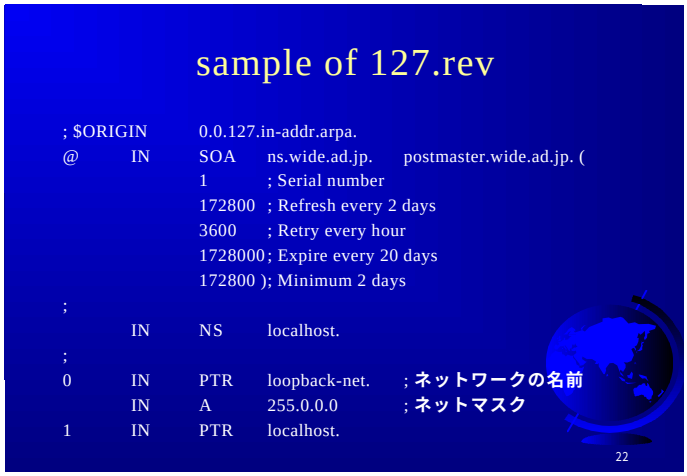
forwarders というのは名前のとおり forward するものという意味です。データの検索要求が送られてきたら、自分が知っていれば、それをそのまま返せばいいわけですが、自分が知らないデータに関しては forwarders が指定されていれば、ルートサーバなどに行く前に、指定されているサーバに検索に行くというような動作をします。

たとえば socks 対応の firewall、socks4 など、外の IP アドレスが分からないとサービスができない、使えないというような場合には、こういう方法をとることになります。

さらに bind 4 の表現では slave、bind 8 の場合は、options forward-only というような記述になりますが、これを使いますと、先ほどのルートキャッシュやネームサーバへの問い合

わせをすべてやめて、forwarders に指定されたところに聞きに行くという動作になります。firewall の場合ですと、問い合わせはある特定のサーバにしか行ってはいけないという設定をすることがあると思いますが、そのときにこの slave というのを指定します。
これを指定しないと、もし forwarders ではなくてネームサーバの情報を知っていたときには、そちらに聞きに行こうとしてしまいますので、そのへんを注意しないといけません。
このへんも、どういう状況のときに、どういう設定をしないといけないかというのは、ドキュメントをみて頂いたらよいかと思います。
キャッシュの有効利用というのは、データを特定のサーバに集めて無駄なトラフィックを抑える、回線が細いときなどに有効、ということです。

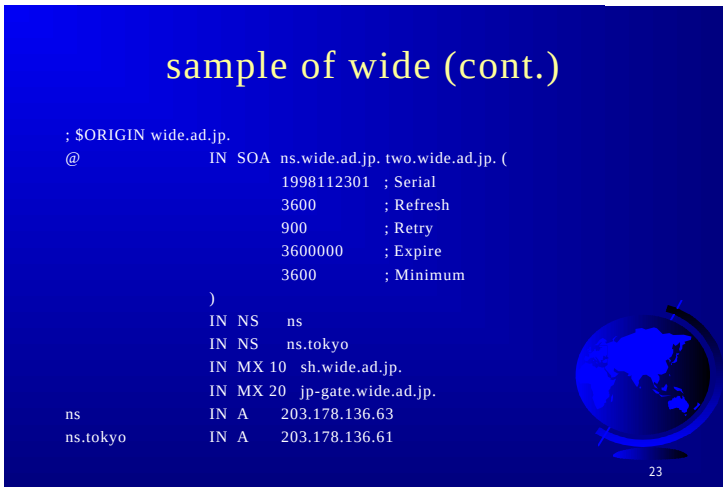


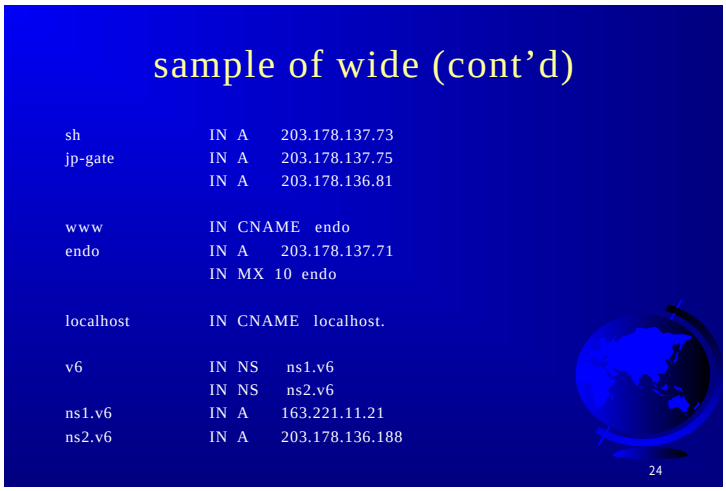


ネームサーバの中の設定の話になってきますが、このようなデータを、いろいろ書いていくことになります。localhost に関しては、あまり他のサーバとの連携というのが発生しませんので、このような形になるという例として見ておくだけでよいと思います。

sample of wide

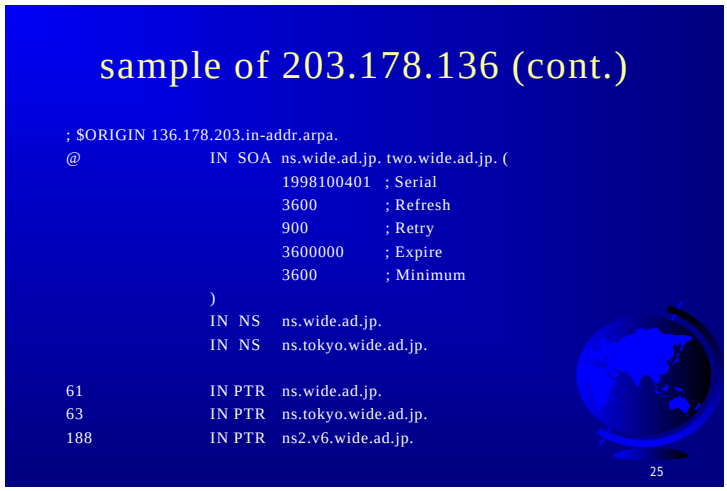
サンプルとして wide のデータを書いてありますが、一般的にネームサーバにデータを設定する場合は、このような感じの内容を設定することになります。ゾーンごとに管理するという形になっていますが、この場合のゾーンというのは、コメントとして$ORIGIN wide.ad.jp.と書かれていますように、wide.ad.jp のゾーンということです。このファイルがどのゾーンかというのは、先ほどの named.conf などのファイルに記述されていますので、参照される方であるこのファイルの中には、明示的に記述されることは少ないです。その中でこういう感じで、IN、インターネットドメインで、SOA とか NS とか MX とか A とか、そういう種類のレコードを、順番に並べて記述していくという形になります。





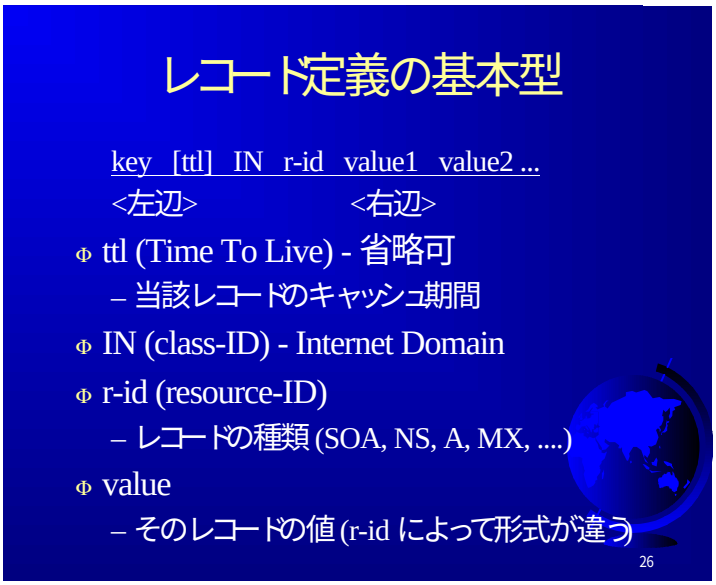
sample of 203.178.136

逆引きのレコード、ゾーンというのは、このような感じになっています。IP アドレスを逆引きするときは、in-adder.arpa というドメインがインターネットで定義されていますので、その下に、IP アドレスを逆順に並べて引くことによって、IP アドレスから名前が得られるという形になっています。当然、その IP アドレスの数字の関係も、先ほどの権限委譲によってサーバの間で連携づけられています。



レコード詳説

レコード定義の基本型



レコードの定義は 1 行ごとのデータの並びで構成されています。キャッシュサーバでデータを拾ったときに、どれだけの期間それを覚えておかないといけないかということをデータを提供する側が決めることができるようにその時間を ttl として指定します。ttl というのは省略可能で、ここに指定することもできますし、別のところでデフォルトとして指定することもできます。

IN の後に r-id(resource-ID)というのが来ます。ここで、SOA、NA、A、MX というのが出てくるわけですが、その後ろに value というのがあり、そのレコードに対応づけられている値が来ます。

レコード定義の基礎知識

・同一 key に対する定義の連続
　後続の定義の key は省略可
・$ORIGIN <domain>
　デフォルトのドメイン名の指定
　初期デフォルトは named.{boot,conf} で指示されるもの
・$INCLUDE <filename> [<domain>]
　ファイルの挿入
・FQDN 表記のホスト名の末尾には . を

基本は最初と最後のところになります。key は省略できるということで、先ほどの wide の例を見て頂いたらよいと思います。

基本的に IN の左側に何も書いていないときは、最初に定義があるところまで戻ってそれを採用するという形になっていますので、この IN NS は、@マークに対する定義という形になるということです。

また最後のピリオドがあるところとないところがありますので、そこに関しても注意してください。

SOA (Start Of Authority) RR

SOA、これは Start Of Authority の略です。ここには、パラメータをたくさん書き並べないといけないのですが、このあとに出てくるのは、まずネームサーバです。プライマリとなっているネームサーバの名前があり、その次に管理者のメールアドレス、そのゾーンに対して何か問題があるときの連絡先となるメールアドレスをここに書きます。

ただしメールアドレスは、@マークをピリオドに変えるという規則になっていますから、たとえば motonori@wide.ad.jp の場合は、motonori.wide.ad.jp となります。

SOA パラメータ

・Serial

Sec-NS のデータ更新判定用

・Refresh (秒)

Sec-NS の Serial チェック間隔

・Retry (秒)

Refresh 経過後のチェック間隔

・Expire (秒)

サービス停止までのチェック不能期間

サービス停止後に nslookup をすると...

*** ns.provider.ad.jp can't find x.co.jp.: Server failed

・Minimum TTL (time to live) (秒)

ゾーン内に定義される全レコードのデフォルト・キャッシュ期間

(キャッシュする全 NS に対して効果を持つ)

Serial というのはサーバが複数ある場合に、プライマリのデータが新しくなっていることをセカンダリが判断するためのデータです。データを編集して新しくするたびに、この Serial の数字を増やすということが決められています。

そのときにセカンダリのサーバがプライマリのサーバを見に来る間隔を Refresh で定義します。

次の Retry というのは、Refresh の間隔での更新時にプライマリのサーバと通信できなかったという場合に、この Retry の間隔でチェックを繰り返すというというものです。この場合は 3600 ですから、1 時間ごとにチェックしに行くわけです。

Expire というのはデータの有効期間を指定するもので、自分の持っているデータが Expire

で指定した期間、1 回もプライマリとの Serial のチェックが成功しなかった場合にはこのデータを無効にして、サービスをやめてもよいというものです。従ってこの場合は、20 日間を過ぎてもプライマリとの間でチェックが成立しなければセカンダリのサーバはこのゾーンに関するサービスをやめてしまいます。
最後の TTL、Minimum TTL というのはキャッシュをした場合の保存時間を規定するものです。この場合は 2d、つまり 2 日になっていますので、1 回拾ったデータは 2 日間そのサーバに残り続けて、問い合わせに対しては、キャッシュされているデータがそのまま返されます。その変更が 2 日後にようやく反映されるという形になってしまいますので、頻繁にデータを更新したいという場合、たとえばホスト名に対して IP アドレスを付け替えたとか、あるアドレスに対して MX のホストを変更したという場合には、この値を 1 時間とか 30 分とか、短めに設定しておくことが必要になります。

Serial について

・Secondary の Primary との同期のため
　内容を変更したら必ず Serial を増加させる
・32 ビット
・. による混乱に注意(使わない方がよい?)
　1.01 = 100001 ("." は "000" と同値)
・1997122501 など日付を使うと明瞭
　一日 100 回更新で 4294 年まで
・上限なし(ループ状):RFC1912(I)
　1 に戻すことが可能
　2147483647(7fffffff)以内を 2 回足す

Serial に関して補足します。これはセカンダリがプライマリとの同期を確認するために使うものです。この数字としては 32 ビットの割り当てがありますので、たとえば 1997122501 というような長い数字列で設定できます。1997122501 というのは 1997 年 12 月 25 日の 1 回目の変更という意味で、こういうやり方をしているところも多いと思います。32 ビットありますので、4294 年ぐらいまで同じ方法で使えます。これはコンパイル時の設定で変えられるのですが、「.」というのは、数字の 0 を 3 つ押し込めるのと同じ動作をすることになっています。「.」をつけて、「.」から下を、その日何回更新したかということにしようと思ってしまいますと、この 3 桁分が無駄になってしまいますので、「.」はあまり使わない方がよいと思います。
32 ビットの数字で 4294 年まで行けるとしても、その後はどうするのかという話がありますが、ネームサーバの Serial に関しては、行き着いたらそこで終わりではなくて、また下から戻って来られるというループ状の Serial の並びが設定してありますので、少しずつ増やしながら永遠に使い続けることができるようになっています。

そうすると Serial が気に入らないということで、また小さい Serial に戻したいということも、一応、仕様的に可能なわけです。少しずつ足していくのは大変なわけですが、7fffffff 以内の数字を 2 回足すことによって、うまく調整すれば 1 に戻すことができるようになっています。それ以上の変化は、逆に Serial が戻ったという判断になってしまいますので、一応、その付近で小さな前後に関しては、Serial がちょっと戻った場合は、プライマリの方がデータが古いという判断になりますし、ある程度大きな数字を足すと、ぐるっと回して元に戻してくることができるという形になっています。

データの再読み込み

・データ更新後、named に SIGHUP を送る

```
# ndc reload
```

・bind 8 以降の場合は、BIND_NOTIFY 機能により、

Secondary に更新要求が送られる(Serial が増加している場合)

Secondary も bind 8 以降が必要

データ変更した場合は、ネームサーバのプロセスに SIGHUP を送って新しいデータを読み込ませることが必要です。

さらに bind 8 以降では BIND_NOTIFY というのを設定してコンパイルしておきますと、プライマリに SIGHUP を送るとセカンダリのネームサーバに対してもデータが更新されたという情報が送られて、セカンダリがプライマリに対してデータを引きに来るという形になっています。

Secondary での手動更新

・FORCED_RELOAD 機能

SIGHUP を受けるとシリアルをチェック

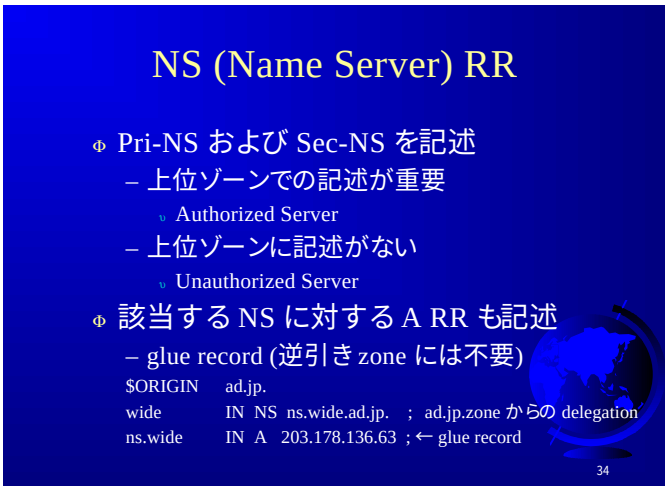
・バックアップファイルを消してから named の再起動

named-xfer で転送がおこなわれる

```
# mv mydomain.zone mydomain.zone.bak
# ndc restart
```

bind 8 を使っていない場合はこのような方法を使って手動でデータを更新するということが必要になるかもしれません。

NS (Name Server) RR



ここでポイントとして出てくるレコードというのは、NS、A、MX、CNAME です。ネームサーバ間の関連で、delegation というものがありましたが、そこに関連する内容として、NS レコード、NS RR というものがあります。これは上位のゾーンから、下のドメインが管理しているネームサーバを指示するためのレコードです。ですからデータを検索していくときに、まず参照されるレコードが NS レコードということになります。従って上位ゾーンに NS のレコードが記述されているサーバが Authorized Server という形になりますし、上位ゾーンを見ても、そのサーバに対する NS レコードが書かれていない場合は、その下のゾーンのサーバは Unauthorized Server という形になりますので、NS レコードを調べることによって、あるサーバが Authorized かどうかというのが分かる仕掛けになっています。

NS のレコードの横には計算機の名前を書くことになっていますが、名前が分かっても IP アドレスは分かりませんので、さらにその IP アドレスを書いておくことになります。

上の例では、wide のゾーンを管理しているネームサーバは、ns.wide.ad.jp.と記述されています。"ns.wide'には「.」はついていないので、ad.jp という ORIGIN が自動的に補われるわけですが、その IP アドレスを一緒に書いてあります。このような、本来必要なさそうだけれどもないと困るという、上から下へのゾーンの対応づけをするために必要な A レコードのことを、glue record(糊づけするためのレコード)とよびます。ですから、ゾーンの delegation をするときには、このように NS のレコードと、そのネームサーバの IP アドレスに対応する A レコードというペアを書いておくことが必要になります。

lame (不完全な) NS

lame NS、不完全なネームサーバというのは、上のゾーンのネームサーバでは NS のレコードが定義してあるにも関わらず、指定されたネームサーバが実際には、その対応するゾーンを管理していなかった場合、またはそのネームサーバ自身がアクセス不能だった場合におきる現象です。当然これは正しい状態ではないわけですが、設定次第ではそういう状況は発生します。そうすると当然メールが落ちることになってしま

いますので、こういう状況にならないような注意が必要です。

A (Address) RR

・A RR

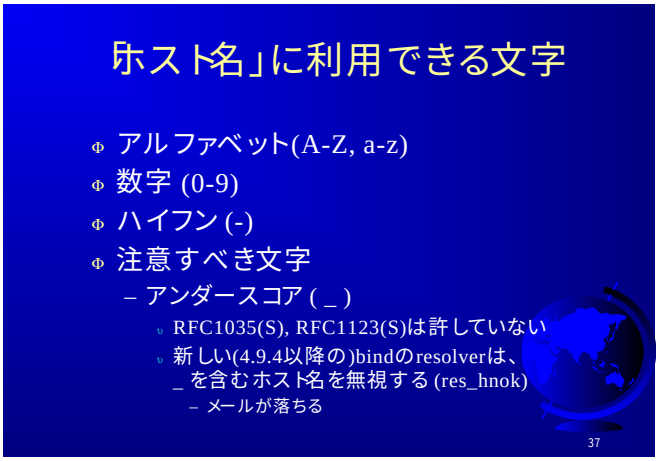
ホスト名から IP アドレスのマッピング

```
$ORIGIN         wide.ad.jp.
sh              IN A 203.178.137.73
```

A レコードというのは基本的なアドレスを定義するためのレコードです。

「ホスト名」に利用できる文字

RFC1035 や 1123 などで決まっているわけですが、ホスト名として利用できる文字には名前付けの規則があります。ここに使っていい文字はアルファベット、数字、ハイフンからなる文字列という形になっています。たとえばよく使いたくなるアンダースコア(_)は RFC 上は使ってはいけないという文字になっています。新しい bind の resolver は、この文字が含まれているデータというのは捨ててしまう仕様になっていますので、ネームサーバに登録してあっても、そのデータは存在しないことになってしまいます。こういう名前のホスト名を、メールアドレスに持つ、あるいは MX の先にこういうホスト名が指定してあると、メールが落ちてしまうことになりますので注意が必要です。



MX (Mail eXchanger) RR

MX レコードの書き方ですが、MX で始まり、最初に数字でプレファレンスを書きその後ろにそのアドレスに対応づけられているメールサーバを書きます。ここで最後のピリオドが必ず必要なことに注意してください。MX レコードで最後のピリオドを省略してしまいますと、その後ろに、さらにそのゾーンの名前が補われますので、sh.wide.ad.jp.wide.ad.jp のような定義になってしまいます。

MX のプレファレンス

・DNS の MX RR に指定するコスト値

・コスト最小

Primary MX / Primary Mail Server

First MX / First Mail Server

・コスト準最小

Secondary MX / Secondary Mail Server

・コスト最小以外

Lower MX (優先度が低いという意味)

先頭の@マークというのは、そのゾーン自体に対する定義ですので、たとえばここに wide.ad.jp.というように書いたのと等しくなります。つまり user@wide.ad.jp という宛先、メールアドレスに対して対応づけられている MX レコードという形になります。

MX RR の右辺と CNAME

・MX RR の右辺に CNAME の左辺となる名前を書くべきではない

・Lower MX が MX RR の右辺にある自分の名前を認識できないと問題

回避策が講じられていれば動くけど*

named が警告を出す

これはセカンダリ MX、Lower MX のときの注意です。MX レコードの右辺が、自分の名前ではないとき、つまり MX レコードが複数定義されていて、それの 2 番目、3 番目に対応するメールサーバが、正しく自分の名前を認識していないと、メールがループしたりエラーになるということがありましたが、そのときにメールサーバが自分の名前を正しく認識するためには、ネームサーバ側でも正しい設定をしておく必要があるわけです。あるホストを示す名前は、いろいろと別名定義ができるわけですが、メールサーバとしては、普通は正式な名称が 1 つ決まっていますので、基本的にメールの送り先を示す MX レコードでは、右辺のところに別名ではなく本名を書くということが重要です。

Wildcard MX

・ *.x.co.jp.　IN MX 10　mail.x.co.jp.

・Firewall がある場合(直接通信できない)

外: 個々のレコードを外部に見せたくないが、ホスト宛のメールアドレスを利用したい

内: 外界をひとつのレコード定義で代表; root に Wildcard MX を定義し、GW に集める

・nohost.x.co.jp や host.nosubdom.x.co.jp にマッチ

無駄なメールが飛ぶ

MX に関して、ワイルドカード MX というものがあります。ネームサーバの中で、「*」印を指定することによって、どのような名前で検索が来ても、そのデータを返すという設定ができます。たとえば Firewall がある場合は内側の情報を外に見せたくないというはずですが、内側のいろいろなドメインに対するメールを受け取りたいという場合にワイルドカード MX を定義すれば内部のドメイン名を公開しなくても可能になります。情報隠蔽という立場からすると、ワイルドカード MX は非常に便利ですが、当然、存在しないドメインやホスト名、また間違ったメールアドレスの場合でもメールがそのゲートウェイまで届いてしまうという欠点もあります。

・specific なレコードが存在すると参照されない

```
ns.x.co.jp.          IN A   12.34.56.78
*.x.co.jp.IN MX 10 mail.x.co.jp.
ns.x.co.jp.          IN MX 10 mail.x.co.jp. (必要)
```

サブドメインが存在する場合も同様

ワイルドカード MX を使う場合の注意点ですが、specific なレコードが存在すると参照されないという問題があります。正確にマッチするレコードがあった場合は、そちらに関連して対応づけられているデータの方がワイルドカード MX よりも優先されます。その場合はこのように、ns.x.co.jp.に対する MX を明示的に書く必要があります。これは当然、サブドメインが存在する場合でも同様です。

Wildcard MX の弊害

・存在しないアドレスにもメールが飛ぶ

送信時に存在しないアドレスであることが不明

・存在しないアドレスに補完される

user@mail.x.co.jp.x.co.jp

回避するには sendmail.cf で

ResolverOptions に HasWildcardMX を定義

・配信先に対応する MX RR が引けない

配信先ホスト名の最後に必ず . を補う

→ どうしても必要な場合にのみ利用する

ワイルドカード MX は Firewall の場合には便利なのですが、むやみに使うと問題があるということで「弊害」といっています。まず 1 つ目は、存在しないアドレスに対してメールが飛んでしまうということ、もう 1 つは、存在しないアドレスに補完されてしまうということです。これは MTA、sendmail などでも、補完をするという設定をしておいたときに、同じことが起きるますが、正しいメールアドレスを書いているはずなのに、mail.x.co.jp.x.co.jp というような補完が行われたりします。この場合は sendmail に対して、ワイルドカード MX がある環境で使っているということを指示するために、ResolverOptions というところに HasWildcardMX を設定することが必要になります。

CNAME (Canonical NAME) RR

・ホストの別名定義

```
$ORIGIN     wide.ad.jp.
archie      IN CNAME sun3.tokyo.wide.ad.jp.
```

末尾の . に注意

CNAME チェインはできるだけ避ける

同一 key に別の種類のレコードを定義しない

同一 key に複数の CNAME は定義しない

・NS, MX の右辺に CNAME で定義される名前を使わない

CNAME は別名を定義するために使用します。たとえば archie というサービス名を wide ドメインに定義するときに、IP アドレスを直接振るのではなく、他のホスト名へのエイリアスとして定義したいときなどに使います。

しかし MX や NS の右辺には CNAME で定義される名前は使えないことになっています。ネームサーバを bind 8 にした場合には、そういう定義が見つかった場合には、エラーが報告されます。

CNAME のチェイン

・CNAME RR の右辺がさらに別の CNAME RR の左辺

```
alias1      IN  CNAME  alias2
alias2      IN  CNAME  real-name
```

・RFC1034(S)

チェインの定義は非推奨(should not)

実装では辿れること(should)

sendmail では 10 回までたどる (MAXCNAMEDEPTH)

named は 8 回までたどる (MAXCNAMES)

CNAME、エイリアスのチェインというのも当然できますが、あまり複雑なことは避けた方がよいと思います。

メールアドレスと CNAME

・エンベロープのエイリアスは本名に書き換えられるべき(RFC1123(S))

・多くの(古い)sendmail **はヘッダも本名に書き換える**

どのアドレス宛に届いたのかがわからなくなる

経路上の sendmail.cf **の設定次第**

・書き換えられたくないときは、MX **か** A **を**

IETF **は** CNAME **による書き換えをしない方に動いている**

DontExpandCnames **オプション** (8.7 **以降**)

古い sendmail **あるいは** sendmail.cf **を使っている場合に、そのヘッダというのは、基本的に書き換えないという話を、最初の方にしたと思いますが、**CNAME **が見つかった場合は、本名に書き換えるべきであるという決まりがあり、書き換えが行われることがありますので注意が必要です。**

ただし RFC **で定義されてるのは、ヘッダではなくエンベロープの書き換えなので、その点からは** sendmail **の実装は正しくないという話もあるわけですが、古い** sendmail **を経由してしまいますと、その** CNAME **で定義されている名前をメールアドレスに使っている場合に、書き換えが行われてしまうことがあります。これは自分の管理外で起きることですので、**CNAME **の使用をやめるという方法しか今のところ手段がありません。**

メール配信時のDNS **検索手順**

(1) CNAME **を解決**

CNAME **でなくなるまでチェインをたどる: 上限あり(無限ループ防止)**

(2) MX **で検索**

複数あれば、preference **でソート**

preference **が同じ時はさいころを振る**

MX **が見つかった時、**A **も同時に** Additional Information **として返される** (DNS **の仕様)**

(3) A **で検索**

MX **が得られなかった時**

個々の MX **に対して**

(Additional Info.**で** A **が得られなかった時)**

・A **しか定義されていなければ、検索処理は** 2 **回必要** (MX **と** A)

ホストにも MX **を定義すべき: 通信トラフィックの削減**

メール配信時の DNS **検索手順ですが、まず** CNAME **を解決してから** MX **を引きます。ここで重要なのは** MX **を引いて見つかったら、対応する** A **レコードをさらに見つけるということですが、**DNS **の場合は** MX **を引いた場合に** additional inmormation **として、**MX **の右辺に対応するホスト名の** A **レコードも返ってくるという効率化を行っていますので、**MX **で見つかった場合は、同時に** MX **のホストに対応する** IP **アドレスも得られるということに**

なっています。もし MX でアドレスが見つからなかった場合は、さらに A レコードで引くことになります。

ホストにも MX レコードを

・障害に備えるため

A レコードだけでは Secondary MX が指定できない

異なるホストに対する IP アドレスを定義した

A レコード(仮想ホスト): 障害対策としては弱い/負荷分散にしかならない

・DNS 検索の効率化

そのホストしか受信しなくても定義すべき: DNS 検索が一度で完了する

つまり A レコードだけで見つかるデータであっても、さらに MX レコードを、そのホスト自身に対して向けておくことで、DNS の検索に関する通信時間、あるいはトラフィックを削減できるというのが、ここでのポイントになります。

アドレスの補完

メールアドレスの補完

・MX RR と A RR を用いる

ワイルドカード MX 問題に注意

・/etc/resolv.conf を参照

domain sub.x.co.jp

search sub.x.co.jp x.co.jp co.jp と同値

遡って 3 階層分調べる (MAXDFLSRCH)

最短は 2 レベル (LOCALDOMAINPARTS)

JP domain の実状にあわない

RFC1535(I) では暗黙の遡りを禁止: 新しい bind の resolver は遡らない

メールアドレスの補完に関してですが、まずワイルドカード MX 問題に注意が必要です。また、resolve.conf で補完のサーチ順を指定しておくと FQDN でないメールアドレスの補完をしてくれるということです。

search sub1.x.co.jp sub2.x.co.jp x.co.jp

・LOCALDOMAIN 環境変数によるユーザ設定

最大 6 ドメイン (MAXDNSRCH)

・検索の順序

nic.ad.jp

nic.ad.jp.sub.x.co.jp

nic.ad.jp.x.co.jp

nic.ad.jp.co.jp

RFC1535(I)より前は nic.ad.jp を最後に検索

そのときの補完順序はこのようになります。うまく検索が行われるかどうかにより、このような補完機能を有効に使えばよいと思います。

PTR (domain name PoinTeR) RR

・IP アドレスからホスト名へのマッピング

いわゆる逆引き

```
$ORIGIN         137.178.203.in-addr.arpa.
73              IN PTR sh.wide.ad.jp.
```

PTR レコード検索によるサービス制限

引けないホストからのアクセス拒否

ドメイン名の確認

・うそつき問題

アドレス→ホスト名 の単方向だと騙れる

引き直しでチェック

逆引きレコードというのは、PTR のレコードで定義します。SPAM と関連して最近では、逆引きがされていないホストからは、信頼できないという理由でメールを受け取らないという設定をしているところもあります。ですからメールが正しく届くようにするためには、逆引きも正しく設定するということが必要になってきています。アドレスからホスト名への対応というのは、適当な設定を書いておくと全く関係ないドメインのホスト名をアドレスに対応付けることができます。これは、セキュリティ的に問題があるということで「うそつき問題」というようによばれていましたが、最近のきちんとしたコードでは、このような点にも注意を払っており､得られたドメインからさらにもう一度 IP アドレスを引いて、IP アドレスが一致しなければ、その逆引きは信じないということになっています。

例: nslookup で逆引きの確認

・ホストの IP アドレスが 1.2.3.4 のとき

```
% nslookup -q=ptr 4.3.2.1.in-addr.arpa.
```

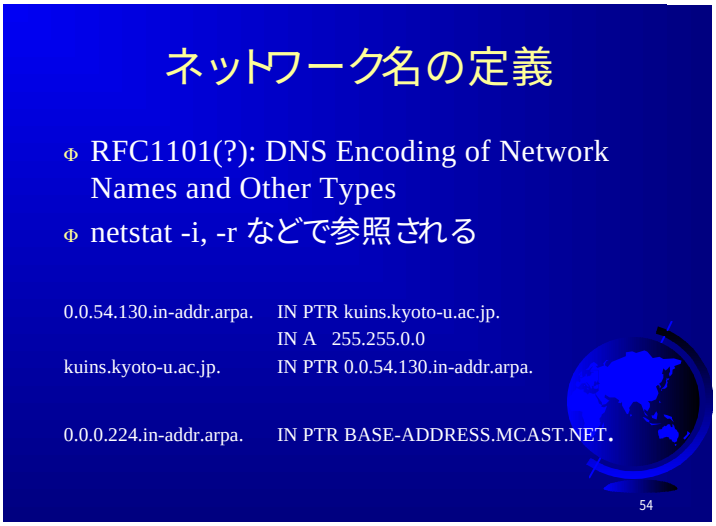
・新しい (4.8.3 以降) nslookup では次の指定も可能

```
% nslookup 1.2.3.4
```

逆引きの確認の具体例はこのようになります。

ネットワーク名の定義

これは、netstat -i **または**-r **を実行すると表示されるネットワークアドレスからネットワーク名を逆引きで表示するための設定です。**



その他のレコード

・HINFO, TXT, WKS

HINFO **は必ず** 2 **つ以上のパラメータを書く!**

・NULL, MB, MG, MR, MINFO (experimental)

RFC1035(S)

・AFSDB, ISDN, RP, RT, X25

RFC1183(E)

・PX

RFC1664(E)

localhost/127.in-addr.arpa zone

・すべてのネームサーバに設定すべき

root server **まで問い合わせるのは無駄**

$ORIGIN my.domain.jp.

localhost IN CNAME localhost.

・引き直しの際の不整合の防止

127.0.0.1 → localhost.my.domain.jp **にならないように**

CIDR と逆引き管理

・class less なアドレスの割り当て

192.0.2.0/25　　組織 A に

192.0.2.128/26 -　　組織 B に

・逆引きゾーンの管理単位問題

オクテット(8 ビット)単位の権限委譲との不整合

・解決策

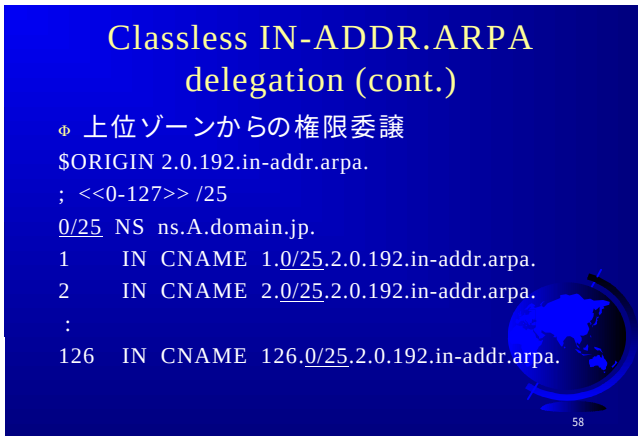
CNAME で散らす: RFC2317(BCP)

Classless IN-ADDR.ARPA delegation

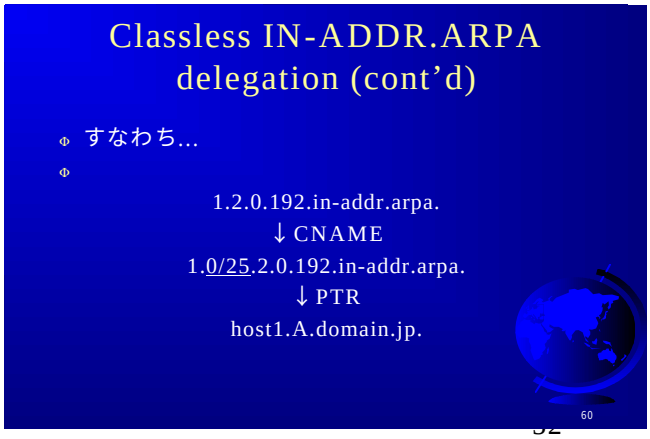
NS で散らす

最近の IP アドレスは、ブロックを細かくして割り当てられていますが、そのような環境で、逆引きをうまく設定する方法が RFC に書かれています。

具体的にはこのようにして、CNAME のレコードで 1 レベル階層を作って、そこでうまく割り振るということをしています。



たとえば 192.0.2.1 を逆引きをしたとき、まず最初に CNAME で飛ばされて、直接上から delegation されているゾーンでの定義が得られます。そこからさらにポインタでホスト名を得るという形で、CIDER 環境でうまく逆引きを実現しています。

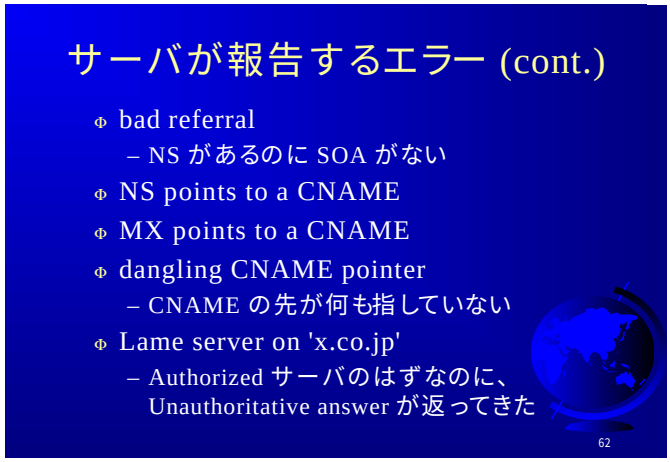


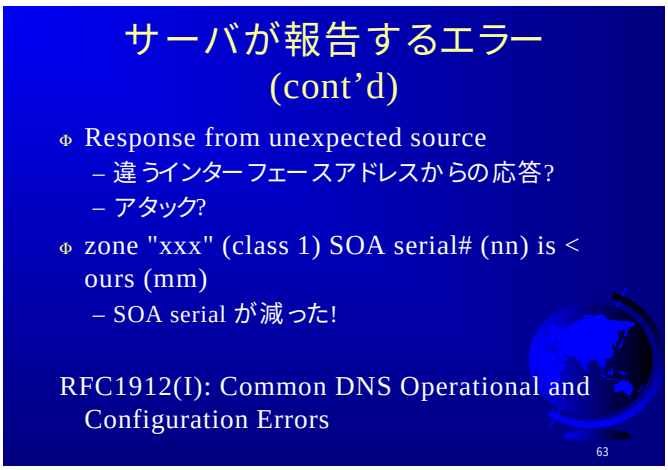
よくある設定の誤り

このあたりは、最近の新しいネームサーバでは、あまり問題になることはないと思います。昔の 4.8.3 など古いネームサーバを使ったり、あるいは 2 つのネームサーバがあったときに、それぞれがあるゾーンのプライマリで逆がセカンダリという関係がお互いにあるというような状況のときに起きた問題です。

サーバが報告するエラー

これまでに紹介したネームサーバが報告するエラーをまとめたものです。
最近のネームサーバは、設定に関して間違った状態になっていると警告してくれますので、これらもログをみて注意しておくことが必要です。





DNS の今後

・Dynamic Update

レコード単位のデータ更新

・Incremental Zone Transfer (IXFR)

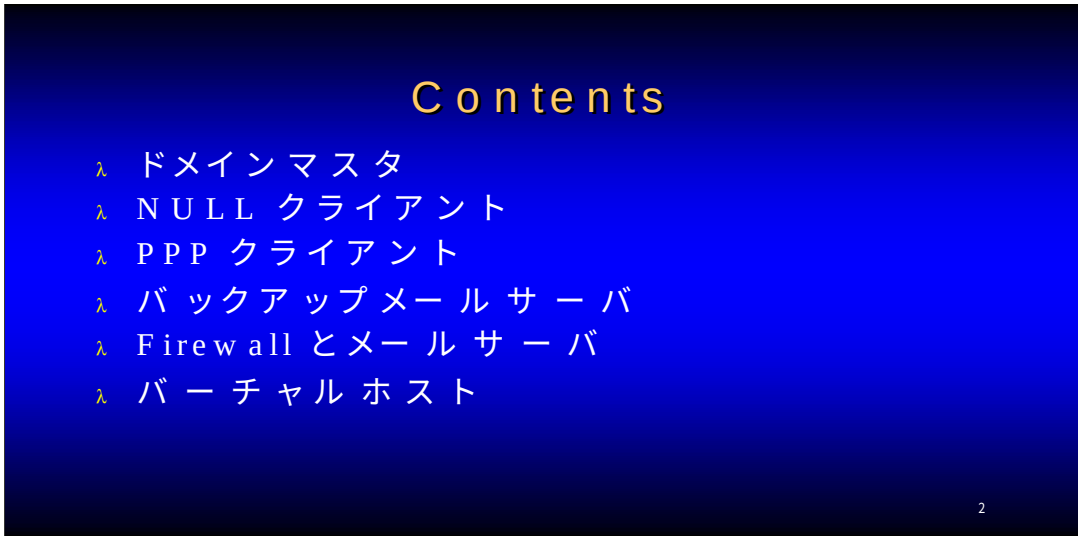
トラフィックの削減と更新速度の向上

・Security Extention

SIG RR, NXT RR

bind 8 に関しては機能拡張がさらに続き、便利な機能が入っていくでしょうから、そのへんも注目しておくとよいと思います。

3. メールシステムのデザイン



これまで基本的な用語、技術の話をしてきましたが、それをどういうふうに組み合わせていくかという話になります。組織ごとの設定というのは非常にバリエーションに富んでおり大変なので、一応、基本的なところ、典型的なところを説明していきます。応用については、たとえば cf の中に多くのサンプルが入っていますので、あとは参考にして下さい。ホストが存在して、そのホスト宛のメールを受け取るとい事に関しては、今まで説明した基本的な設定をすれば、とくに難しいことはないはずですので、それには触れず、更に、付加的な機能をつけていこうとした時、どういうことが必要かということを説明します。

ドメインマスタ

- user@mail.x.co.jp の他に user@x.co.jp も受理する。
- メールの発信時に user@x.co.jp を発信者アドレスにする。
- 計算機を特定すべきメール(root などからのメール)は、user@mail.x.co.jp が望ましい

メールアドレスに関して、たとえば user@mail.x.co.jp は、ホストの名前に対応づけられているメールアドレスです。それに対して、さらに user@x.co.jp というメールも受けたいという場合に、このホスト名を省略したもの（generic なアドレス）を受け取る役割をするメールサーバのことをドメインマスタと呼びます。(最近は、あまり呼ばないのかもしれませんが……)UUCP 等でやっているときは、ドメインマスタは、さらに、下流（ 他のホスト、クライアント）に対しても仕事・役割を受け持っているので、重要な位置づけにありましたが、最近は、単に、generic な形式のアドレスを受け取る普通のホストのような役割でしかなくなってきています。

generic な形式のメールを受け取るということは、出すときもこういう形式のアドレスを使いたいというわけですから、発信の時にも generic なアドレスを使うという設定が必要にな

ってきます。
人間が使うということに関して考えると、generic な形式の要求が出てくるわけですが、計算機が、自動処理（計算機がクーロンで動いていて、何か仕事をして、その結果を送ってくるような）をするような時には、どの計算機からのメールかがよく分からなくなってしまいますので、当然、計算機をしっかり特定すべきメールアドレスは必要になります。そういう例外も考える必要があります。

ドメインマスタの設定

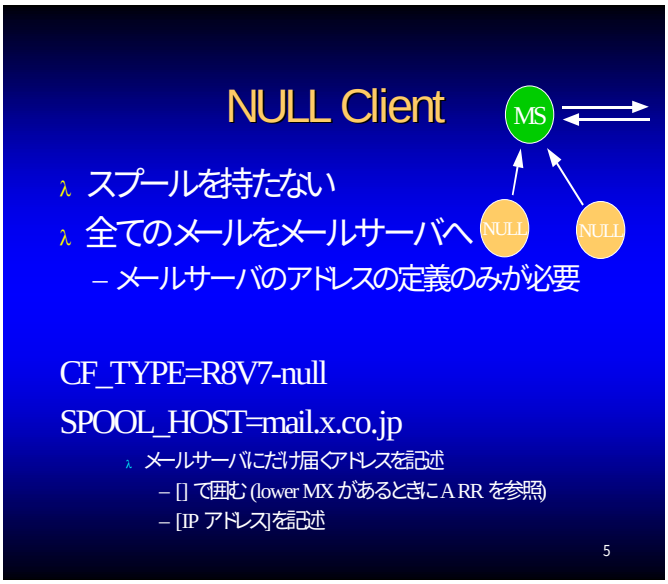
sendmail の場合(CF の場合)は次のように書きます。

- ACCEPT_ADDRS='x.co.jp'
  - 受理すべきアドレス部 (これがポイント!)
- FROM_ADDRESS='x.co.jp'
  - 送信時のデフォルトのドメイン部
  - 管理用アドレスにはホスト名が付与される

    root, daemon, postmaster,...
- 複数ドメイン宛を受理
  - ACCEPT_ADDRS='sub1.co.jp sub2.x.co.jp'

qmail の場合は localnames 等に設定します。 つまり、受信するための設定と、発信するときの設定があり、それぞれ設定が必要であるというポイントさえ押さえておいていただければいいと思います。
複数のドメインを受けたい場合、「sub1.co.jp sub2.x.co.jp」と列挙すれば、どちらのドメインにも属するような形でメールを受け取ることができるようになります。これは、あとで出てくるバーチャルドメインの一番安直なやり方にも相当します。

NULL Client



計算機がどんどん増えていき、それに従って、すべての計算機でメールを使いたいが、それぞれの計算機に対してメールの設定をするのが大変だという場合に、NULL Client という、簡易な設定の方法があります。

これは、スプール（自分のところにメールを貯めること）をしない、メールボックスを持たないというものです。つまり、NULL Client は全てのメールを、他のサーバに投げるということはせずに、必ずメールサーバ(MS)に送るという機能だけを持っています。CF では、NULL という特別な設定がありますので、それを使います。

NullClient の場合、あとで出てきます SPAM 関連の設定は、非常に手抜きになっています。CF で SPAM を拒否する設定をこれと組み合わせると、すべてのメールを拒否してしまうという問題があるらしい事が判明しています。ファイアウォールの中で非常に簡単に、メールのサーバの設定をする時には便利な設定です。

PPP クライアント

PPP クライアントは、メールサーバを管理している側の話ではなくて、家庭、等で、UNIX ワークステーション、等のサーバから、ダイアルアップで接続するような時に、どういうふうにしたら良いかという事です。

ダイアルアップ環境なので、つねに接続されているわけではないというのがひとつ、あります。そうすると、自分専用のモデムを持っている人は IP アドレスが固定的に決まっているかもしれないので別ですが、普通はプロバイダに電話でダイアルアップし、その度 IP アドレスが自動割り当てで決まりますから、DNS 的なホスト名、つまり IP アドレスに対応づけられている名前も毎回変わります。そういう名前は、インターネットから見えるから良いのですが、逆に、自分の家でダイアルアップする側（クライアント側）のホストに付けている名前（内部的ホスト名）がインターネット側に流れ出してしまう事になります。とくに、メールアドレスに自宅のホスト名が出てしまうと、返事を返してもらえない事、等ありますので、注意が必要です。

発信者のアドレスは、契約しているプロバイダから自分に割り当てられているメールアドレスを使うことになります。自分のメールアドレス(ドメイン形式の user@domain のユーザのところ)が、自分の家で使っているワークステーションのユーザ名と一緒の場合は問題

ないのですが、違っていると変換する作業、操作が入ってきて面倒になります。
受信は、POP (popclient など)で拾ってくればいいというふうに考えると、送信のときにうまくやることを考えないといけなくなります。
ダイアルアップで接続して、コネクションを張ろうと思った時に、毎回接続するような設定になっていると電話代もかさみますので、それを防ぐことも考えないといけません。

PPP クライアントの設定

・ DIRECT_DELIVER_DOAINS=none　　(プロバイダのメールサーバが発信用に使える時)
・ DEFAULT_RELAY=mail.provider.ne.jp(プロバイダのメールサーバが発信用に使える時)

とします。 mail.provider.ne.jp は、プロバイダーのサーバです。そうすると、自分から直接メールを出さずに、プロバイダのメールサーバに一旦メールを投げて、そこから送るという形になります。つまり、先ほどの NULL Client と同じような形になります。
プロバイダによっては、認証等の関係から最初に POP しなければならず、こういう方法が使えないところもあるかもしれませんが、その時はまた工夫が必要になります。たとえば直接投げる方法もひとつの解かもしれませんが、メールサーバが使える場合は、上のように書きます。
次に、発信者のアドレスですが、ユーザ名が等しい場合は、po.provider.ne.jp を補うことですみます。

・ FROM_ADDRESS=po.provider.ne.jp

すぐに発信しない工夫ですが、これはウィンドウズ等のクライアントからメールをもらって、さらにそれを転送するという形態のメールサーバを考えています。直接発信する場合でも、結局は一緒です。SMTP_MAILER_FLAG_ADD で e という FLAG を出し、さらに、CON_EXP に True と書いて設定します。これによって、SMTP で送ろうとしているメールに関しては、一つ目は、すぐに実行しなくなります。

・ CON_EXP=True　　（すぐに発信しない工夫）

_c SMTP_MAILER_FLAG_ADD=e （すぐに発信しない工夫）

expencive なメーラ(最初の処理に対してコストが高いメーラ)は、個々に送らずにまとめたいので、一つ目は、すぐに処理せずにキュー(mqueue)に貯めます。キューに貯まったメールは、たとえば 30 分ごとに処理をするといった設定もできますし、それも無駄だという場合は、手動で sendmail -q を実行すれば、まとめて配信されます。手動でキューの処理をさせたい場合は、sendmail デーモンを-bd だけで、キューの処理間隔を指定せずに起動します。

高度な PPP クライアント

・発信者アドレスの書き換え

ローカルのユーザ名と契約ユーザ名の変換

userdb, usertable **の利用**

・契約していないアドレスからの発信の抑制

check_compat **ルールセットの利用**

・自動ダイアルアップ時のタイムアウト

O DialDelay=15s

実際には、これでうまくいくと思いますが、ネームサーバへのクエリーが飛んでしまう可能性が残るかもしれませんので、ネームサーバのクエリーだけでダイアルアップ接続が起こらないようにする為の工夫がさらに必要かもしれません。

さらに、ローカルのユーザ名と契約のユーザ名が違う場合は、senddmail **の場合、**userdb **や** sertable **を使用して、外に出て行く時にユーザ名を書き換えるようにするような工夫が必要になります。**

それ以外のメール、例えばルートからのメールや、自分が意図して発信したもの以外のメールは外に出ていくとまずいので、(SPAM **にも関連しますが)**check_compat **でエラーにして、外に出さないような設定も必要になります。**

自動ダイアルアップで接続する時、たとえば sendmail -q **と手動でメールを送り出そうとした最初のコネクションの時に、モデムがつながりネゴ後** IP **が上がるまでに、時間がかかりますので、**sendmail **のバイナリ自体の機能で、** DialDelay=15s **というように設定できるようになっています。**

バックアップメールサーバ

バックアップメールサーバの役割。

・ 1st-MX **の障害発生時に代わりに受信**

- 2nd-MX **として動作**

これはメールを送ってもらうところの設定の話です。DNS **のところで出てきましたが、**MX **レコードに複数のホストを列挙して書くことによって、たとえば１番コストの低い(**preference **の数字の小さい)サーバに最初にトライし、それがだめだったら、よりコストの大きいサーバに対して順番にトライしていくのですが、そのときに、２番目、３番目のメールサーバは、特殊な設定をしなければ、単に** 1st-MX(**最初に受け取るべきメールサーバ)に対して、メールを転送してくれるだけです。そこで、**2１つ目のホストがトラブルを起こした時に、メールをきちんと届けて欲しい場合、(１つ目のメールスプールに貯まっているメールは駄目ですが)**2nd-MX **からも直接送ってもらうことができれば、**1st-MX **が止まっても、メールを受け取る事ができます。**

・ 可能なら 1st-MX **と独立に直接配信(**2nd-MX **から直接配信する方法)**

- 1st-MX と aliases ファイルを共有
- 同じアドレスを受理する方法

全てのユーザ　　ACCEPT_ADDRS=

特定のユーザ　　SECONDARY_*=

指定したもの以外は、1st-MX の回復を待つ

1st-MX が止まっていても 2nd-MX から直接配信する方法です。aliases ファイルで転送先を定義して 1st-MX とそのファイルを共有するのです。共有の方法としては、ＮＩＳ、ＲＣＰ、Ｒディスト、等を使う方法があります。

全てのユーザに対しては、1st-MX と 2nd-MX の２つのメールサーバの設定内容をまったく一緒にして、その aliases を共有したうえで、ACCEPT_ADDERS に、該当メールアドレスを設定します。 たとえば wide のドメインマスタであれば wide.ad.jp と書くと、受け取ったメールのうち、 ACCEPT_ADDERS に列挙したアドレスに対しては自分宛だと思うので、さらに aliases を見て処理するということになります。

ただ、片一方のメールサーバでメーリングリストも運営しているとまったく一緒にはできないことがあります。その場合には、ユーザ個人宛のものは直接投げよ、メーリングリストに関しては 1st-MX が回復するまで待って 1st-MX に送れ、という設定が必要になってきます。そのような場合、CF では SECONDARY の設定を使うことで、直接送ってもいいユーザを特定でき、その人達だけに関しては 1st-MX を経由せずに直接投げることができます。

aliases を共有する方法を次に示します。

- NIS などの共有手段を利用する
- rdist して newaliases する方法もある
- ローカル aliases と共有用 aliases に分離

sendmail バージョン８(R8)は複数ファイルの aliases が扱える

OA/etc/aliases, nis: mail.aliases

⇨ ローカルな aliases ファイルと、共有している aliases ファイルを別々に管理することができる。

Firewall とネームサーバ

・組織外向けネームサーバ

Wildcard MX を定義する場合

```
 $ORIGIN x.co.jp.
*   IN   MX   10 ext-mail.x.co.jp.
```

内部のホストの存在を見せない

存在するメールアドレスをすべて定義

Wildcard MX を使わない方法

エラーになるべきメールが GW までやってこない

Firewall があると、 DNS との関連についても十分注意をしなければならなくなってき

ます、組織の外に対してサービスをするネームサーバと、組織の内側に向けてサービスをするネームサーバという使い分けが必要になってきます。
外向けの場合は、Wildcard MX を定義して、内側のホストとか、ドメインの存在を見せないという方法と、存在するメールアドレスをすべて定義するという方法があります。どちらを選択するかは、ポリシーの問題ですので、ご自由にどうぞ、と言うしかありません。

・組織内向けネームサーバ
　組織外のアドレスを内部に見せない方法
　　あらゆるサービスはすべて proxy 経由
　　DNS タイムアウトを避けるため
　　組織内に root サーバを設置
　組織外のアドレスを forwarders で拾ってくる方法
　　内部から外部に直接接続できる場合
　　socks

組織内向けは、組織の外のアドレスを見せるか見せないかによって設定が変わってきます。組織外のアドレスを内部に見せない場合は、組織の内側に、そういうドメイン階層をつくることになります。これをいい加減にしてしまうと、ルートに行こうと思っても名前が引けなくて時間がかかったり、それによって、テンポラリーフェイリアのような、メールが出て行かない状況になる可能性がありますから、注意が必要です。組織外のアドレスが見えないと困る場合は、最初の DNS で説明しましたように、forwarders の設定で拾ってくる必要があります。

Firewall とメールサーバ(1)

メールの環境は、DNS の環境がきちんとあった上で、機能するという関係になります。メールサーバの台数については、コストを考える必要があります。メールサーバをたくさん用意してしまうと管理が大変だと思ったり、逆にメールサーバを分けたほうが設定がすっきりするので、そのほうが楽になるかなと思ったりしますが、どちらを選択するかは個人の趣味によるという気がします。とりあえず、１つでやるか２つでやるかという方法がありますので、それぞれに関して考えてみたいと思います。

ネームサーバとメールサーバの構成

・メールサーバを 1 台でまかなう

a. 内向け zone を持った外部検索用ネームサーバを参照する方法

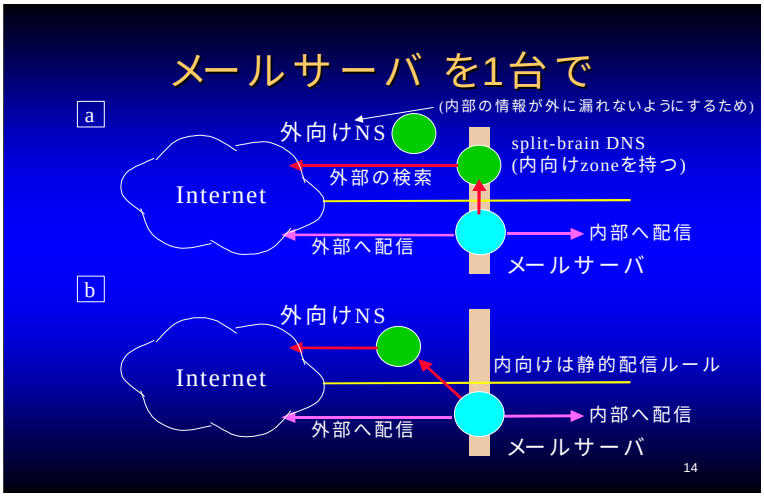
split-brain DNS

b. 内部は固定ルールで配信する方法

まず、１台でやる方法ですが、方法としては２つあります。内向けゾーンを持った外部検索用ネームサーバを参照する方法、つまり、メールサーバを１台でやろうとすると、そのメールサーバが、外向けの配送も、内向けの配送も両方きちんとできなければならないので、両方に対してメールアドレスから接続正規 IP アドレスが分からなければならない事になります。それと、内側に関してはそれほど宛先が多くない、サブドメインがいくつかしかないという場合に、内側は固定ルールにしてしまうという方法です。

a.の場合、split-brain DNS という形の、１つのネームサーバ（メールサーバが参照するためのネームサーバ）を、わざわざ用意することになるのですが、そのネームサーバは、内側の zone を持っており、かつ外側の zone も検索できる形のネームサーバになります。そうしますと、ネームサーバの、authorization あるいは delegation が、外向けの情報と内向けの情報という、２つの情報が混ざってしまうという問題が出てしまいます。内側の zone がさらにサブドメインに対して delegation されている場合に、ネームサーバが、きちんと下流の zone を見に行ってくれるかどうかという課題があり、そういったところを整理しなければなりません。

メールサーバを１台で



内側のゾーンは、セカンダリになるという形で切り抜けたとしますと、形態としては、上図 a.になるわけです。その場合、ネームサーバ（緑色）は、当然、２つ必要になって、１つ目の外向けのネームサーバ(NS)は、内側のデータが引けない、外に対してサービスをするためのもの。それともう１つ、メールサーバが検索するためのネームサーバ(メールサーバと同一計算機内)を別に用意する。このような形でやりますと、一応、１つのメールサーバで、内向きも外向きも DNS の情報を見て投げることができます。当然、Firewall(水色)がありますから、他の直接的なメールの通信はできない。必ず、このメールサーバを経由しますから、このメールサーバが外向けのアドレスを見た場合は、外の DNS の情報が拾えて、内側に関しては、a.左側ネームサーバがセカンダリとして、キャッシュしているデータを使うという形になります。

b.のように、内側は DNS を見なくても良いと考えると、内側をスタティックにして、a.右側の DNS サーバはいらなくなりますから、さらにシンプルにすることができます。

メールサーバ１台の場合は、DNS の設定が面倒臭いとか、静的な設定をしておくのが面倒臭いといったことがあるのですが、メールサーバを２台にするという方法を考えれば、状況はわりとすっきりします。　メールサーバを２台にした場合は、当然、外向けの DNS サーバを参照するメールサーバと、内向きの DNS サーバを参照する（つまり内向きの配信に責任を持つ）サーバとの２つに役割が分かれます。それぞれに対して、方式がａとｂとあります。

Firewall とメールサーバ(2)

メールサーバを 2 台

- **外部 DNS を参照するメールサーバ**
- **内部 DNS を参照するメールサーバ**

方式 a

両者間は静的経路設定

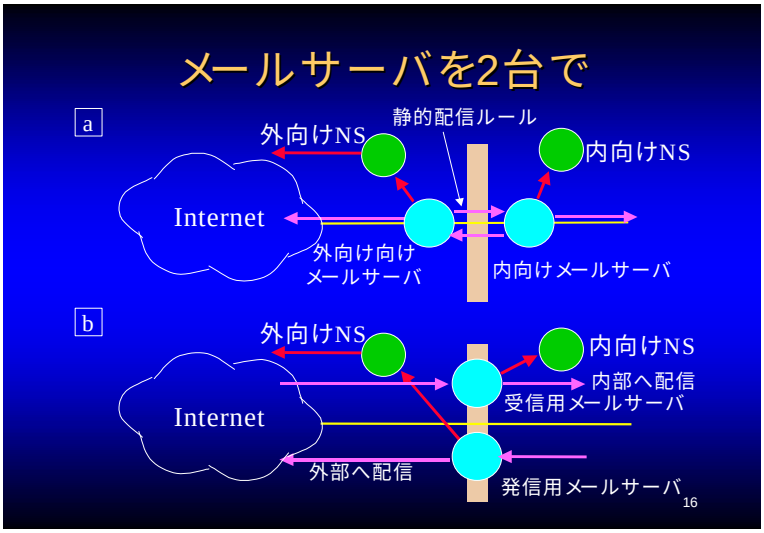
方式 b

受信専用メールサーバ

発信専用メールサーバ

この２つのメールサーバの間をどうするかということに関して方式が２つあります。

メールサーバを２台で



ａの方法は、Firewall の両側にメールサーバを置くわけです。この Firewall の間はお互いのメールサーバの間のトラフィックだけしか通過できない設定になっていると、内側のほうは内側の DNS を見ながら内側にメールを投げるし、外向きのやつは外側の DNS を見てメールサーバに投げるというような形で連携させることによって実現できます。

それに対してｂの方法は、メールサーバを２台とも Firewall 上に置いてしまい、内側からやってきたメールは発信専用のメールサーバに投げて、発信専用のメールサーバは DNS を見ながら外に投げる。逆に、外から来たメールは、MX を受信専用のメールサーバに向けておくことで、完全に役割分担をさせてしまうことで実現します。

ｂのほうが、SPAM の対策も簡単になるという気もします。

内部向けメールサーバの設定

・外部への静的配信ルール

DIRECT_DELIVER_DOMAINS=x.co.jp

DEFAULT_RELAY=external.x.co.jp

外部向けメールサーバ

ほとんど sendmail **の話になってしまいますが、たとえば** external.x.co.jp **という名前のメールサーバが外向けのメールサーバだとしますと、そこにメールを投げるという設定をすればいい。つまり、自分の組織内宛ではないメールが** DIRECT_DELIVER_DOMAINS **に関係していて、直接投げるドメインは、**x.co.jp **の内側と指定します。だから、メールアドレスに** x.co.jp **がつくものは直接投げる、それ以外は** DEFAULT_RELAY **に投げる、という形になります。これによって、外側向けのメールは外部に投げる形になります。**

外部向けメールサーバの設定

一方、外向けのメールサーバは逆になります。

・ 内部への静的配信ルール

STATIC_ROUTE_FILE=x.static

x.static **の内容**:

GW [12.34.56.78]

(internal.x.co.jp)

DOM x.co.jp

メールサーバ宛てのメールは受理可能

基本は DNS **を見て投げたら良いのですが、内側向け（**x.co.jp **が含まれている）メールに関しては、**internal.x.co.jp **というメールサーバに投げなさい、という設定をすることになります。そうすると、たとえば** cn **の場合は静的配信ルールの設定ファイルの中に、**x.static **と指定します。この内容は、ドメイン(DOM)が** x.co.jp **のメールに関してはゲートウェイ(GW)に投げなさい(直接、**internal.x.co.jp **と書いてもいいのですが)と書いた場合は、その外向きのネームサーバで、その** IP **アドレスが引けなければならないという条件がついてきますので、普通（**internal.x.co.jp **というアドレスは、外向けネームサーバには見えないはずですから）、ここには** IP **アドレスで直接書くという形になります。このメールサーバは、自分宛のメールは自分で受け取ることになっていますが、特殊な設定は何もしていませんので、**ser@external.x.co.jp **等の宛先のメールは自分で受け取る、受理するという形になります。**

ネームサーバを１台で

・ NS, MS **とも** 1 **台でなんとかするには...**

a. NS **で内側に** first MX **を向けておく**

inner-host　IN　MX　10　inner-host

IN　MX　20　gw

外部からは 1st-MX と直接通信できず、

タイムアウトが発生

― 送信側のストレスになるのでよくない

b. GW で内側へは A RR を参照して配信

inner-host　IN　A　12.34.56.78

IN　MX　10　gw

２台でやるのが一番すっきりしている方法なのですが、そんなに計算機をたくさん置けないので、１台でなんとかしたいという要望もあると思います。ネームサーバもメールサーバも、１台でなんとかする方法を考えてみます。これもいくつかやり方があります。

a.で MX を複数（２つ）定義しておきます。MX の first（プライマリの、つまり数字の小さいほう）は内側のホストに向けておく。プライマリではない（MX の数字が大きい、２番目のもの）方にゲートウェイ（Firewall のメールサーバ）を書いておく。そう書くと、IP アドレスは、当然、外から両方見えることが前提になるので、これは外から内側の情報は見えてもいいという立場になります。こう設定し、ゲートウェイ(GW)というホストの気持ちになって考えると、とにかく１番目を最初に好みますから、inner-host に対して、自分は直接通信ができ、自分が受け取ったメールは目的ホストに送れる。外から見た場合を考えると、まず、外から MX を引いて inner-host に投げようと思うのですが、Firewall が途中にありますので、直接通信ができずに、ここでタイムアウトが発生する。そうすると、１番目に失敗したので、次に２番目に送ろうということでゲートウェイに送る。ゲートウェイは Firewall のところにあり、直接見えるということで、ゲートウェイに送られる。そうすると、ゲートウェイ（GW）が中継して inner-host に渡すので、メールをやりとりすることもできます。

ただ、Firewall の設定にもよりますが、直接通信できないホストに対して、何も返事しない Firewall の設定であれば、タイムアウトをずっと待つことになります。そうすると、ＴＣＰの標準的な設定、実装であればタイムアウトに 75 秒位時間がかかりますので、その分メールが届くのが遅くなる。メーリングリスト等で、こういう設定をしている参加者がたくさんいると、タイムアウトのたびにメールが遅くるかもしれないので、あまりよくない。Firewall で、アンリーチをいきなり返す設定になっていれば、コネクションはすぐに切れるので、すぐに次に進むから良いと思いますが、それでも無駄なトラフィックが飛んでしまいますから、それもあまりおすすめではありません。

そこで、同じような構成で、ずいぶんましな方法がｂ.になります。これは、inner-host（外から見えるアドレス）に対して、Ａレコードに加えてゲートウェイ（GW）に対する MX だけを書いておく。

c. 内側を別の枝にマップ

inner.domain.jp → inner.domain.jp.local

sendmail.cf でアドレス変換をおこなう

STATIC_ROUTE_FILE の MAP 行 (CF)

ｃは、内側を、別の枝にマップするというやり方です。これも、実際にやっているところがあるようです。たとえば inner.domain.jp というメールアドレスで、ゲートウェイにメールが飛んできたとしますと、そこで内側向けのアドレスに関しては、inner.domain.jp.local のようにさらに下に文字列を補ってゾーンのデータを定義することによって、別のドメインを参照するようにできます。結局、ネームサーバに対してスタティックでいっぱい定義するのと同じ事を、DNS にまとめるという方法で実現することができます。sendmail.cf では、ドメイン名の対応づけ、うしろに.local 等を追加するのは、STATIC_ROUTE_FILE の MAP 行を使うとできるようになっています。

d. 1 台に複数のデーモンを起動する

外側/内側の IP アドレスにバインドさせる

外向け named と 内向け named

listen-on, query-source, transfer-source (bind8.1.2)

外向け sendmail と内向け sendmail

O DaemonPortOptions=Address=12.34.56.78

いわゆる virtual host の設定

ｄ.の方法ですが、１台に複数のデーモンを立ち上げる方法を使っても、同じようなことができます。これは、サービスをするサーバは、物理的な、計算機という面で見ると１台ですが、その上で動いているサービスとしては２つ（ネームサーバが２つとメールサーバが２つ）の形になるわけです。この場合、外側のインターフェイスに対してやってきたリクエスト（query）に対しては、外向けのネームサーバが答えて、内側のインターフェイスのアドレスに対してやってきたリクエストに対しては、内向けのネームサーバが答えるということができれば、１台の計算機の上で、複数のサーバを起こすことができるわけです。bind 8 の場合、 IP アドレスを限定するためのオプション（configuration ファイルの listen-on や query-source や transfer-source）がありますので、それを用いて外側のインターフェイスの IP アドレスを、列挙しておきます。当然、configuration ファイルは２つ用意して、それぞれ起動するときに configuration ファイルを分けて指定するのですが、これをすることによって、１台の上で、複数のネームサーバを起動することができます。同じようなことで、sendmail も複数起こすことができます。この場合はメールキュー(mq)のディレクトリを変えておく等が必要になるという気がしますが、それを注意すれば可能です、sendmail の場合、DaemonPortOptions という設定行がありますから、例えばそこに、Address=12.34.56.78 と書いておくと、そのアドレスに対してサービスを待ち受けるデーモ

ンが起動されます。そうすると、いわゆる virtual host の設定になるので、内向けのインターフェイスと外向けのインターフェイスに対して、複数の IP アドレスが定義されており、それぞれに対して sendmail を起こすという事をすれば、１台で、virtual host のような設定もできます。ここまで行えば、複数の計算機に分けてやっているのと同じようなサービスが１台で実現できます。

・ a, b 方式の問題
- 内外の直接の通信は不可なのに
- 外から内部ホストの情報が見える

bind8 の allow-query だけでは役不足

ａやｂの方法では、内側の情報が外に見えてしまう、あるいは直接通信ができないホストなのに外に見えているから無駄なタイムアウトが発生してしまう、という問題が出てきてしまいますが、そのへんを避けたいと考える人は多いと思います。
たとえば、１台のネームサーバでがんばるときに、ネームサーバが問い合わせを受け付けて返事をする時の、クライアント側の IP アドレスの制限がでてきます。つまり、最近の、そして少し前のネームサーバには、どこから来たリクエストだけには答えよう、ということができる仕組みが入っていますが、１つのネームサーバに、外から来る query に対して答えてあげるデータと、内側から来たときに答えてあげるデータを区別して覚えさせるということはできません。片一方だけしかできないので、それを組み合わせて１つのサーバでがんばるというところはまだできていないようです。

b 方式の具体的設定

・ゲートウェイから内部への配信

静的経路定義

A RR を見る

1st-MX が自分だった場合の挙動の変更

TRY_NULL_MX_LIST=True (CF)

O TryNullMXList=True (sendmail.cf)

・正しく設定できていないと

local configuration error になる

ゲートウェイ（GW）の設定を考えたときに、標準的な設定だったら、MX を見て、自分が MX の先頭になっているので、自分が受け取るべきかなと思うわけです。ところが、最初に、メールの設定の基本事項を３つ説明しました。送ってもらうための設定と、受理するための設定と、自分が送り出すための設定です。MX が自分を向いているということは、自分に送ってもらう設定に関しては、うまくいっているのですが、自分が受理する設定にはなっていないわけです。これは inner-host に向けるメールですからうまくいかない。そこで、

これもメールサーバ（MTA）の実装に依存する話ですが、MX が自分だった場合に、自分だと思い続けるというのと、MX は見なかったことにしてAを次に見るという、2通りあります。Aを見るという設定をすると、特別な設定をすることもなくきちんと inner-host にリレーをしてくれます。それをするための設定として sendmail の場合は、TryNullMXList を設定することで、（自分が MX の first ではあるが）自分が受理しない場合にAを見るという動作になります。それをしない場合、あるいは MTA がそういう機能を持っていない場合は、静的な経路定義をしないといけません。

ゲートウェイ内クライアント

- **なんでも** GW **へ**

  DIRECT_DELIVER_DOMAINS=none

  DEFAULT_RELAY=internal.x.co.jp

- **内部については直接配送**

  DIRECT_DELIVER_DOMAINS=x.co.jp

  DEFAULT_RELAY=internal.x.co.jp

- qmail **の場合は、**control/smtproutes **に定義**

先ほどの NULL Client **とほとんど一緒なのすが、それをゲートウェイ内クライアントという標準的な設定でやるための方法です。先ほどの、内側と外側のメールサーバを関連づけるときの話と同じ感じで、**DIRECT_DELIVER_DOMAINS **に直接投げるドメインを書いておき、** DEFAULT_RELAY **にそれ以外の投げるところを書いておけば良いわけです。**

ゲートウェイ内クライアントで、Firewall **があって、必ずゲートウェイを通過しないといけない場合は、直接送るところは何もなしだよ、と言うわけですし、**Firewall **のホスト同士は直接送ってもいいよ、という場合は、直接送るべきホストの範囲を指定します。**

「qmail **の場合は」というのがいきなり出てきますが、**qmail **の場合はすべて** control **の下のファイルで設定するわけですが、**smtproutes **というところに、こういうドメインの場合は、このホストに送りなさいというものを設定することになっています。**

バーチャル・ホスト

バーチャル・ホストの話が出てきましたが、もう少し詳しく説明します。他のやり方に関しても、一緒にまとめて説明をします。

・ 1 台のホストで複数アドレスを利用

a) **ユーザ空間の共有**

USERTABLE_MAPS='domain1=hash:/etc/map1 ¥
domain2=hash:/etc/map2'

b) **ユーザ空間の分離**(1)

- 1 **つのホストに複数の** IP **アドレス**
- **アドレスごとに** sendmail **をバインド**

O DaemonPortOptions=Address=1.2.3.4

- chroot **で環境も分離すると良い**

ユーザ空間を共有する方法。これが最初の accept adders **のところで複数のドメインを指定する方法で、ユーザ空間を共有している場合は、曲がりなりにバーチャル・ホストができると説明してきましたが、もう少し複雑な方法として、**USERTABLE_MAPS **を使うことができます。この場合は、エリアスを、それぞれのドメインごとにバラバラに定義することができます。ですから、たとえば** domain1 **に関しては、この** map **を見て、送り先、転送先を見なさいと定義されます。このファイルに、**domain1 **に関する転送先が書いてありますので、それを見ます。**domain2 **に関してはこれを見ましょう、という形で、エリアスにあたるものを分離するという方法です。**

ユーザ空間が共有していてもいいという場合は、a)**で良いのですが、ユーザ空間を分離したいという要求に対しては、b)の方法です。先ほど出てきました、1 つのホストに複数の** IP **アドレスを設定する等バーチャル・ホストを実現する方法です。ユーザ空間を分けたいわけですから、**etc **のパスワード等は、別々に管理したいので、**chroot **等で環境を分けて、それぞれ別々に** etc **のパスワード等を管理するところまでがんばる必要が出てきます。**

c) **ユーザ空間の分離**(2)

- sendmail.cf **でがんばる**

アドレスごとに local mailer **を切り替え**

- /etc/passwd **とは別のデータベースを利用**
- POP **等専用サービス**

ドメイン名を含んだ識別子でユーザ認証

もうひとつの方法は、OS **的にがんばるのではなくて、**c）sendmail **側でがんばるという方法です。**local mailer(binmail **や、**mail.local **のこと)は、**sendmail **から渡されたメールをメールボックスに入れる役割をするプログラムなのですが、それが参照するパスワードファイルを別々に用意しておきます。別々に用意したパスワードファイルを、手を加えた** local

mailer が参照できるようにして、それで、たとえば POP を、スプールを別々に分ける。たとえば、その場合に、ドメイン名までをユーザ名として、POP の認証のときに渡すことで、そのスプールを、ドメインごとに切り換えるという、統一的な手の入れ方をすることで、バーチャル・ホストをつくる方法もあります。
１つのホストでがんばる場合には、sendmail 的にあまり手を入れない chroot でがんばる方法と、local mailer に手を入れてがんばる方法がありますので、このへんでがんばりましょう、ということになるかと思います。

システムデザインのまとめ

- どのアドレスを受理するか
- アドレスごとの配信方法の選択
  - 配信先を静的に定義
  - ネームサーバ (MX) を参照

これらをしっかり整理する

ポイントは以上の通りで、どのアドレスを受理してどういう転送をするかの組み合わせにしかすぎないわけです。これらをしっかり整理し、ネームサーバとの連携をしっかり押さえることが必要になります。

付録：以外と知られていない（？） sendmail テクニック

付録ということで、sendmail を使う場合に、意外と知られていないテクニックをまとめてみましたので、簡単に、紹介しておきたいと思います。

- 送信待ちメールをひとつのホストに集める
- 巨大なメッセージを拒否する
- 発信者アドレスによって処理を変える
- SMTP に失敗したら UUCP で送る
- メーラの処理順序の指定
- 個人レベルの ML 設定

送信待ちメールをひとつのホストに集めるには

１つ目は、送信待ちのメールをひとつのホストに集める方法です。

- FallBackMX オプションを利用する
  - DNS が引けなかった場合
  - すべての MX にメールを送ることができなかった場合

  に指定したホストにメールを転送する
- mqueue の管理等が楽になる

- 最長保存期間の調整

・ 経路のトラブルに気がつきやすくなる

組織の中に、sendmail が動いているホストがたくさんある場合に、宛先が、ネットワークトラブルでアンリーチだった時に、メールがそれぞれのメールサーバのキューに貯まります。そういったメールを確認して、ネットワークのトラブルをモニタするのに使う為に、１カ所に集めたいときは、FallBackMX オプションを設定して、DNS が引けなかったり、メールを SMTP で送ろうと思ったけれど送れなかったという場合に、最終的に送るべきホストを指定することができます。FallBackMX にホスト名を書いておくと、最終的に、そのホストにメールが貯まるので、そのホストで、mqueue を管理すればいいし、保存期間が短いと思った場合は、そのホストだけ保存期間を長くする事という管理が楽になる手法があります。

巨大なメッセージを拒否する

最近、マイクロソフトの OS がはびこっていて、ワードやパワーポイントの、巨大なファイルを送ってくる人が絶えないのですが、(業務的に必要なものいは別ですが)巨大なファイルは受け取りませんという具合にメッセージサイズで拒否することができます。

①・ MaxMessageSize オプションを利用する場合

- 受信時にサイズを超過したメッセージを拒否
- ESMTP の場合は、MAIL FROM の時点でサイズが通知されるので、そこで拒否

②・ メーラ定義に M= を指定する場合

- 一旦メッセージを受信してしまう
- 送信直前にサイズをチェックする

メーラごとに許容サイズが異なる可能性があるため

sendmail では、①MaxMessageSize オプショで設定する方法と、②メーラの M=に設定する方法と、２つあります。

①は、メールを受け取る前に判断します。ESMTP で、メールを送るときに、メールサイズが通知されるので、相手が ESMTP 以外のときは受け取ってしまうのですが、ESMTP の場合は、メールを受け取ろうとした瞬間に拒否できます。

②でチェックしますと、メールを一旦受け取ってから、それに対してサイズを見て、エラーを返すというやり方になります。宛先ごとにメールサイズの上限があり、特定の宛先には、一定以上の大きさのメールを送りたくない場合に、こういう設定が使えます。

発信者アドレスによって処理を変える

少し高度なテクニックなので簡単に説明しますが、長期出張等に出かけている際に、エラーメールや、管理者宛のメールを捨てたり、他のメールボックスに送っておきたいという時の設定です。

・ **エラーメールや** SPAM **の分類を** sendmail.cf **レベルで実現**

```
CT root news postmaster MAILER-DAEMON uucp cron
S0
 :                                          エンベロープ発信者
R $*                                  $: $1 $| $>3 $&f
R motonori $| <@>       $: trash              <> のとき
R motonori $| $=T<@$*>          $: trash            クラス T で検査
R $* $| $*                            $: $1
 :
```

宛先は私の場合ですと、motonori **宛で届くので、宛先で判断することはできません。それで、**sendmail **で配信処理のところで、発信者の情報を拾いだしてチェックし処理をすると上のように書くことができます。たとえば１行目の人たちから送られてきたメールは** trash **という名前のメールボックスに送るようにする事ができます。**

SMTP に失敗したらUUCP で送る (3.1Wpatch)

オリジナルの sendmail **ではなくて、**3.1W **の** patch **をバイナリに当てた場合の機能になりますので、注意してください。**

・ 複数のメーラを順に起動する機能を利用

```
S0
:
R $*<@x.co.jp>$*        $# smtp $@ x.co.jp $: $1<@x.co.jp>$2
(スペース)              $# uucp $@ uucp-x $: $1<@x.co.jp>$2
:
```

SMTP **と** UUCP **両方のリンクを持っていて、**SMTP **の回線（インターネットの回線）が止まっている場合に** UUCP **で送る方法です。**

メーラの処理順序の指定 (3.1Wpatch)

メーラの処理順序の指定です。

・ 大規模 ML **などで、**SMTP **より前に** local **への配信（自分あて、アーカイブ）を先に処理させたい場合**

メーラ定義に %= **で優先度（コスト）を指定**

Mlocal …, %=0

Msmtp …, %=10 (local mailer **の後に処理される)**

メーリングリスト等、処理に時間がかかる仕事を sendmail **にさせている時に、本来、自分宛はローカルメーラとかメールボックスに入れられのですが、いつもメーリングリストが処理した後で届くので、一番不利だと思っている人もいるかと思うのですが、そういう時に、自分宛のメールを先に処理する方法です。**

個人レベルでのML 設定

バーチャルドメインに似ているのですが、個人レベルでメーリングリストを設定する方法です。

・ CF **の** localdeliver **機能を利用**

user@host **だけでなく、**user+opt@host **も利用可能**

さらに、opt@user.host **もサポート**

ユーザは .forward **のほかに** .forward+opt **や** .forward+default **が利用可能**

したがって、.forward+ML **を設定することで、**ML@user.host **を運用可能**

参考： Samples/virt-domain+.def

cf **の機能を利用します。たとえば、**user+opt@host **という形のメールアドレス機能を有効にすると使えるようになります。これはさらに、読み換えの設定をすると、**opt@user.host

という形でメールアドレスが指定できますので、DNS の WildcardMX と組み合わせることによって、ユーザごとにアドレスをつくることができて、ユーザごとのメーリングリストをつくることができるという機能もあります。

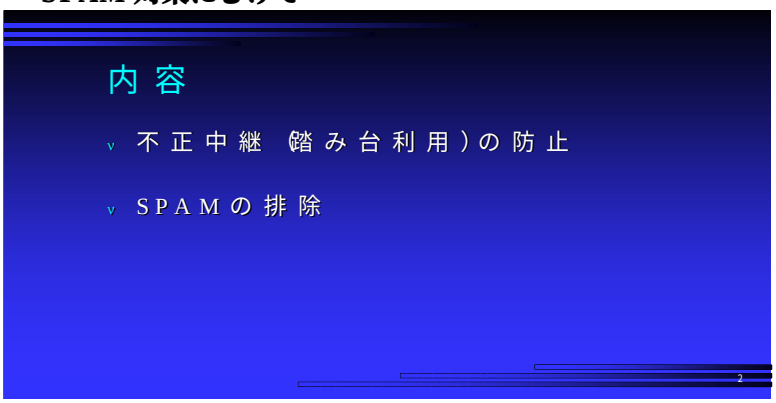
sendmail の重要なオプション

sendmail の重要なオプションがいくつかあります。sendmail を使うことのある人は見ておいていただけたら良いと思います。

- EightBitMode=pass8
  MIME でない 8bit データをそのまま通過させる
- SendMimeErrors
  RFC1894 - DSN (Delivery Status Notification)
  に従ったエラー通知を返すかどうか
- ConnectionCacheSize
  接続したままにする SMTP コネクションの数
  run queue の時に役立つ
- PostMasterCopy
  エラーの発生したメールのヘッダのみを postmaster にも送る
  致命的なトラブルを未然に防ぐため
- DoubleBounceAddress
  エラーメールを発信者に送り返せなかった場合の送り先
  本文も届いてしまうことに注意

補足をしておきたいと思います。MX レコードとAレコードを併記して、Firewall をはさんだときに、内側を見せてしまってなんとかするという方法ですが、プライベートアドレスを内側に使っていて、それを DNS で外に見せるということは混乱が起こりそうなので、やめたほうがいいと思います。ですから先ほどの、望ましくない理由がさらに増えた事になりますが、注意してください。

４．メールの配信制限　　～ SPAM 対策にむけて ～



SPAM の説明に行きます。SPAM 関係の話題としては、リレーの防止と、実際に SPAM を受け取ることに関する防止や排除の２つの内容に分類できると思います。おおまかには分類しているつもりですが、混ざっているところもあるかもしれませんので、そういう前提で読んで戴きたいと思います。

従来のメールシステムの問題

- メールを送り届けることが第１目標だった
- 中継やソースルーティングにも寛大
  - ネットワークの相互接続のため
    ゲートウェイを明示的に指定
    user%domain@gateway
    @gateway:user@domain
  - 不安定・不完全な経路制御への補助
    適当な途中のホストをゲートウェイに指定
  - 細い回線の補助
    他組織によるバックアップ MX

昔はインターネットがつながるということに生きがいを感じていた人達がいっぱいいて、いろんなつながり方ができましたが、その時に、何とかしてメールが相手に届くといいということで、いろいろな方法が許されていたし、節度を持ってみんな皆が使っていました。前半のメールアドレス形式の説明の中に、%-Hack や、ルート表記とかで明示的にゲートウェイを指定して、こういう指定をしないと、ちゃんとメールがやりとりできないという時代もありました。さらに、色んなインターネットのリンクが不安定な場合に、使えないリンクを避けてメールを送るという時にも、%-Hack とかが使われていました。さらに、細い回線の補助ということで、いろんな組織にバックアップ MX をお願いする形で、勝手に MX にされていることもたまにありました。わりと寛大な時代があり、つまり、どういうメールでも、右から左に通すということができるのが、自然な形だったわけです。

着信・配信制限への要求

最近は、どんどん商用化が進んで、いろんな人がインターネットに入ってくるという状況になり、たとえば意図しないメール(宣伝メールとか、あるいは嫌がらせメールとか)が次第に増えてきました。

・ 意図しないメールの受信

メール爆撃 (Mail Bombing)

宣伝メール (Spam)

Unsolicited Commercial Email (UCE)

・ 意図しないメールの中継

組織外から組織外へ(踏み台)

Third-Party Mail Relay

->CPU、ネットワーク、ディスクの圧迫

->時間の浪費（転送時間、メールの整理）

意図しないメールを送る手段のひとつとして、リレー（中継）の為のサーバとして使われることも次第に増えてきました。こういうことが起こりますと、ＣＰＵとかネットワークとかディスクを圧迫したり、不要なメールを捨てるために、わざわざ削除の操作をしたり、さらに、種々のメールの中から、本当にいらないメールなのか重要なメールなのかを区別する必要がでてきたりします。 そういう、無駄な時間を使う必要が出てきたので、それを何とかしたいという要望が叫ばれるようになってきたわけです。

・ ポリシーによる通信先制限

組織内のみ

学生実験、アルバイト、出向社員

・ メールボックスのサイズ問題

SPAM で溢れ、重要なメールが受け取れない

一種の Dos アタック

ディスク管理の問題

POP で「サーバに残す」、IMAP

メールを読まない人たち

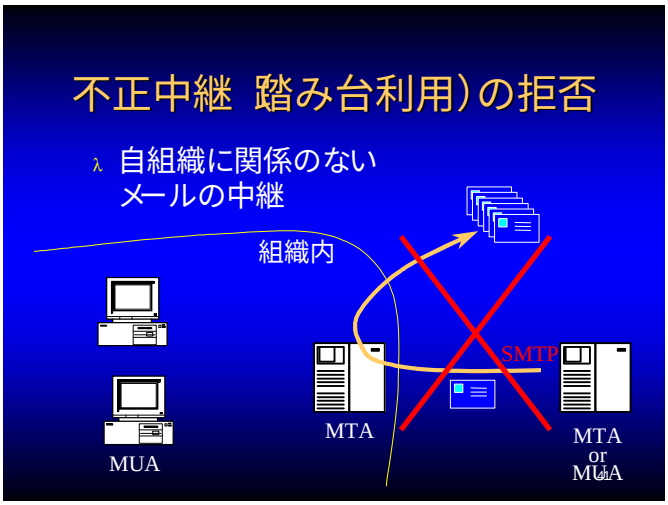
expire 処理

さらに、SPAM だけに関する問題ではなくて、もうちょっとポリシーがらみの話で、大学の中で学生はあまり外と通信してはだめだとか、あるいは会社でアルバイトや出向社員に対して、あまりメールを外とやりとりしてもらっては困るといわれるようなことがあって、組織内のメールだけを許すような設定がしたいといった声もあります。

管理上の問題として、ディスクをどんどん使ってしまうので、重要なメールが受け取れなくなってしまったり、POP でサーバにメールを残す必要があったり、あるいはＩＭＡＰを使うことによって、ディスクがたくさん必要になったので、困ったり、あるいはメールを読まない人たちがいて、メールがたくさん貯まって困る、という課題がでてきます。

そんな時に、メールを少なくすることも必要だということで、いろんな要求が出てきます。

不正中継（踏み台利用）の拒否



まずは不正中継に関する説明から始めます。前半、SMTP の話をしたときに、受信者のアドレスをエンベロープにたくさん書くことができると説明しました。メールは電子的なものですから、簡単にコピーできます。右図で、メールが１通、外（右下）から飛んできて、ここ（真ん中 MTA）で踏み台にされて外（右上）に出ていくとき、宛先がたくさん（SMTP の仕様では 100 個までですが、）アドレスが書いてあると、この途中のメールサーバまでは１通（宛先名が例えば 100 人あるメールが１つ）送られます。それがこのサーバで実際に配送しようとしたときに、100 通に分かれるので、最初にメールを出す人は１通分送るだけで良いのですが、ここで 100 倍に増幅させられるので、踏み台に使われるということになります。

SPAM の踏み台の役割

・ たくさんの宛先をもったメール 1 通から宛先の数だけのメールを複製する

-> 踏み台ホストへの影響

発信者の回線費用、CPU 資源などをかたがわり

SPAM 発信の嫌疑、苦情対応の手間

組織に対するマイナスイメージ

ブラックリストに載り、メールが届かなくなる

->対策

明示的な転送（ソースルーティング）を拒否

通常不必要な 組織外 → 組織外 の転送を拒否

メール１通が、その宛先の数に分かれるので、100 倍になりますから、それだけの通信回線費がかかる、ＣＰＵ負荷もかかる。そういう、計算機資源の問題だけならあきらめればいいのかもしれませんが、受信者に届くと、受信者から苦情が出てきて、対処に時間がかかる。さらに最近は、踏み台にされる設定をするのは管理が甘いという風潮になってきていますので、企業とか組織のマイナスイメージにつながる。さらに、そういう踏み台アクセスを許しているところは、データベースに載せて、そういうところからはメールを受け取らなくするようなシステム、仕組みが出てきていますから、もしそうなると、自分のところからメールを送ろうとしてもメールが届かないという状況が発生することになります。リレー、踏み台に関しては、組織外から組織外のメールの流れを防ぐ対策をすれば良いわけで、これは技術的にうまく設定すれば可能ですから、まず、その説明を先にしたいと思います。

配信制限のための判断材料

配信制限、リレーの制限の為の判断材料として、何が使えるかということです。

・ SMTP 接続元/先のホストの－

IP アドレス

ドメイン名（DNS の逆引きによる）

・ エンベロープの－

発信者のメールアドレス

受信者のメールアドレス

・ ヘッダの内容

・ 本文の内容

ここでは、インターネットの SMTP でメールがやってくる状況を基本として考えています。SMTP による通信をする状況で、いろんな情報が得られます。最初に得られるのは、SMTP で接続が確立したときに、自分のメールサーバに対して接続をしてきている別のメールサーバの IP アドレス、ドメイン名、ホスト名がまず分かります。

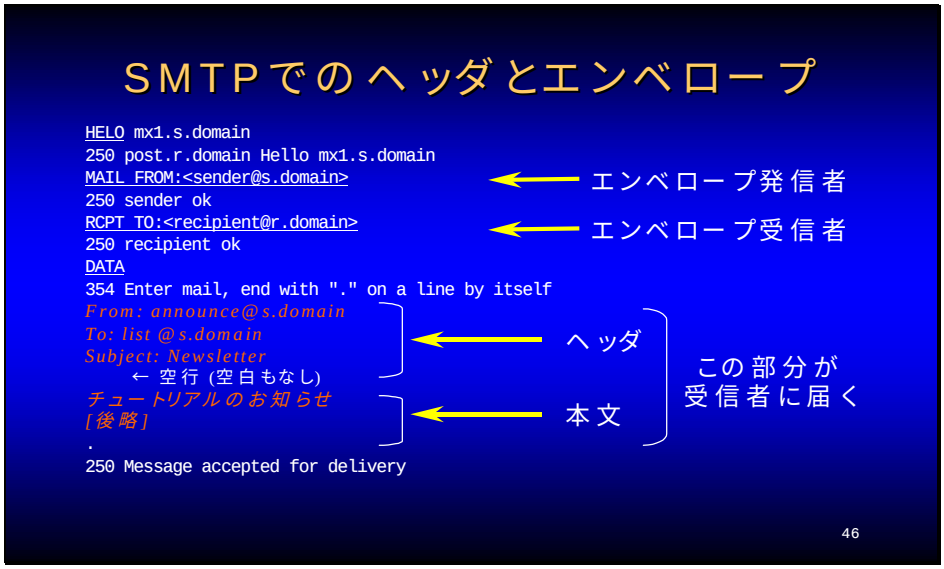
次に、実際に、やりとりが開始されるわけですが、やりとりされる情報として、エンベロープのメールアドレス、発信者のメールアドレス、受信者のメールアドレスがあります。さらに、ヘッダが送られて本文が送られてくるという形で一連の通信が行われるわけですので、これらの情報から、いらないメールの転送を見つけて拒否することができれば制限が可能になります。

ヘッダとエンベロープ

- 封書に似ている
- ヘッダ(header)のアドレス
  - 内封された書面の送り主/宛て先
  - 配信処理中は基本的に書き換えなし
- エンベロープ(envelope)のアドレス
  - 封書の表書きの送り主/宛先
  - 配信処理中に状況に応じて書き換え
  - 着信後は UNIX From や Return-Path: に残る
    - 単に残るだけであることに注意

ヘッダは書き換えられないし、その最初の発信者が書いたそのままです。それに対してエンベロープは、実際のメールのやりとりの際に使われるもので、随時、書き換えられていきます。実際にメールがやってくるときは、エンベロープの情報に従ってメールが送られてくるわけですから、まず、エンベロープの情報のほうが重要になります。

SMTP でのヘッダとエンベロープ



SMTP で送られてくるときに、どういう情報が渡されるかといいますと、まず、HELO でホスト名が渡されて、その次に MAIL FROM でエンベロープの発信者が渡される。その次に RCPT TO でエンベロープの受信者が渡される。たとえば SPAM とかで 100 人宛のメールが送られてきた場合は、この RCPT TO の部分が 100 個、この位置に続くという形になるわけです。さらにデータがあって、メールのヘッダと本文が続きます。情報を 1 つずつ受け取りながら、ある時点で「このメールはいらない」と判断するというのが、ここでの考え方になります。

SMTP のどの時点で制限するか

どういう時に判定すれば良いのかという殊になります。当然、情報は少しずつもらってくるわけですから、ある情報をもらった時点で判断できる材料がそろっていれば、その時点で拒否すれば良いのです。

- SMTP コネクションの接続時
- SMTP セッション中の応答
  - HELO/EHLO ホスト名
  - MAIL FROM: <エンベロープ発信者アドレス>
  - RCPT TO: <エンベロープ受信者アドレス>
  - DATA (ヘッダ、本文の内容)
- SMTP 受信完了後・配信前
  - ->負荷を抑えるためにも SMTP の時点で拒否したい
  - エラーメールの生成は送信側の仕事になる

拒否するポイントとなるところというのは、どういうところがあるかというと、まず、SMTP

コネクションを接続しに来たときに制限するという方法があります。当然、この時点で制限できるのは、接続しに来たホストが既に通信したくないホストであるということが分かっている状況でしか、ここで拒否することはできないわけです。
さらに進んで、セッション中の応答ということで、順番に、HELO で渡されてきたホスト名でチェックする。これはあまり信頼性がないので、あまり有効ではないと思います。次に MAIL FROM で、エンベロープの発信者アドレスでチェックする。ここで、たとえば常連の発信者のアドレスだったら拒否する。次に RCPT TO を見て、場合によっては、FROM と TO の組み合わせで拒否するということもあります。データが始まってから、そのメールの内容を見てチェックするということも可能です。さらに、メールを受け取ってしまって、次のサーバに送ろうとする瞬間に拒否する、エラーにすることもできます。
これだけの可能性があるのですが、まず負荷を抑えるために、SMTP の時点で拒否したいということがあります。どういうことかというと、SMTP の時点というのは、「接続時」や「応答」のところです。SMTP 受信完了後は、もう、SMTP はこの瞬間に終わっていますから、この、応答と受信完了の間で、それより前は SMTP がつながっている、それより後ろは SMTP が切れているという違いがあります。SMTP がつながっている時点では、送ろうとしてきているサーバに対して、「いやだよ」という返事を１行返すと、送ろうとしてきているサーバが、そのエラー処理をしないといけない。それに対して、SMTP が切れてから「だめ」と言った場合は、自分自身でエラー処理をしないといけないという違いがあります。負荷を考えると、自分はできるだけ仕事を減らしたいわけですから、SMTP の時点で拒否できる方が良いというのが、ここでのポイントになります。

制限すべきパターンは何か

拒否すべきパターンとしては何があるかということです。

- 組織外から組織外へ
  「From:組織内」は許す?
- 「From:組織内」が組織外からは?
- 「From:組織外」が組織内からは?

今、不正な中継・踏み台アクセスをやめさせたいと思っているわけですから、ここでのポイントは、組織外から組織外、つまり外から来て外へ帰っていくメールは、関係ないだろうと思うわけです。外から自分のところへ来たメール、あるいは自分のところから外へ行くメール、さらに内側だけでやりとりされているメールは自分のところのメールサーバが関連するメールですが、外から来て外へ帰るメールは、本来自分のところのメールサーバを通る必要がないわけですから、そういうメールは拒否すれば良いというのが、まず分かるわけです。
組織外から組織外のメールを防げば良いわけですが、いくつか、例外があるかもしれません。組織内の人が、いつも自分の会社の中にいるのであれば良いのですが、出先から、出

先の別のプロバイダに接続して、そこからメールを出すという状況もあります。そのときに使うメールアドレスに、自分の会社側のメールアドレスを使いたいと思った場合に、そのメールアドレスで自分の会社宛にメールを送ると、自分の組織内のアドレスが FROM について外から来るという状況が起こり得ます。そのメールが外から内側に向けて来るのであれば良いのですが、メールサーバとして自分の組織のメールサーバを指定しておいて、さらに宛先が組織外だった場合に、組織外から組織外という通信が起こってしまいます。でも、実際に使っている人は自分の組織の人なので、FROM が組織内だから良いかと、ちょっと悩みます。

逆に、FROM に組織外のアドレスがついているメールが組織内からやってくるという状況もあるかもしれません。たとえば、オープンな組織外の人が参加できるメーリングリストを、会社組織内で運用しているというところがあたっとすると、組織外から組織外の通信に一見見えるような状況も起こり得ますし、その時に FROM を見ると組織外の人がメールを出している、という状況になるわけです。

そういうメーリングリストのことも考えると大変だと思ってしまいます。そのへんを、SPAM 対策というか踏み台対策をする場合は、ちゃんと整理して考えないといけません。

転送制限とメーリングリスト、転送設定

・ 組織外ＭＬに参加

組織外アカウントからの転送

FROM が組織内のものが、組織外からくる

宛先が組織内なら許可？

・ 組織内ＭＬに組織外の参加者

組織外への転送

FROM が組織外のものが、組織内からくる

組織内のホストからの接続は許可

発信者の書き換えを義務づけ・転送の禁止？

・ エンベロープ発信者を書き換える ML は OK

逆に、組織外のメーリングリストに自分の組織の人が参加している場合とか、あるいは組織外のアカウントがあってそこから自分の組織向けにメールが転送されてくる、という場合も FROM が組織内のものが外から来ることになります。プロバイダを使う以外にもこういう例があります。

中継形態の分類

８つに分けたたのが、右の絵です。中継形態の分類として、SMTP を張りに来たホストが組織内か組織外かということでまず区別しています。次に、それぞれエンベロープの FROM と TO が、組織内か組織外かを区別して分類しています。そうすると、SMTP を張ってくるアドレスと、エンベロープの発信者のアドレスと受信者のアドレスという、３つの選択肢に対して、それぞれ内、外という区別があるので、８つの分類ができるわけです。

それぞれに対して、どういう状況かを考えながら見ていくのですが、組織内からやってくる FROM が内側で TO が外というメールは、普通に発生する状況です。内側の人が外に向けて、普通にメールを出した状況です。

内側からメールが飛んできたけれど、FROM が外で TO が外だとすると、これは先程説明しました、内側でメーリングリストを動かしていて、あるいは内側に、アカウントの forward の設定をしている人がいて、それが使われているという状況が考えられるわけです。そういうのは、踏み台というよりは、その組織のポリシーの問題になってくると思いますので、「？」をつけてありますが、一応、OK にしようというわけです。

逆に、 FROM が内側で TO が内側は、内側同士のメールですが、なぜかゲートウェイを経由して送られてくる。これも、負荷を減らすためには直接送って欲しいのですが、まあ良いかなと思うわけです。
最後に、内側からやってきたメールを見ると、エンベロープの FROM が外側で TO が内側というのがある。これはきっと、内部のメーリングリストで、その配信先が内側なのですが、上の２つが OK なら、これも OK かなというわけです。
とにかく、組織内からやってきたものは何でも OK でいいではないかと OK にしておきます。
問題は、外から来るメールです。踏み台アクセスというのは、「外から来て外に帰る」というものなので、注意深く見ます。まず、エンベロープの FROM が内側で TO が外側というものがあります。TO が外側というのは、当然、外から来て外に帰っていくメールですが、FROM のところが内側になっていますので、そうすると、自分の組織の中の人が、たとえば ISP につないで外からメールを送ってきているのかな、という気がするわけです。これは運用方針というか、難しいところなんですが、これを許してしまうと、ＳＰAM を送る人が、その組織の人のメールアドレスをかたってメールを送ってきてしまっても、メールが通ってしまうことになりますので、このへんはあきらめてもらうしかないのか、あるいは、あとで出てきますように、POP で認証して、一定時間は許すとか、そういう工夫をしないといけないことになります。基本は禁止する方向で動くしかない、という状況にありそうです。
次の２つ目は、FROM も TO も外のものがやってきているのですから、これはもう、当然、許しがたいものになります。外からやってきているもので、FROM が内側なのだけれど、TO も内側というものは、外側にメーリングリストがあって、組織内の人がメールを出して、そのメールが返ってきたときに、もしかすると起こるかもしれない。ここで、メーリングリストと言っているのは、メーリングリストの設定によって、エンベロープをきちんと書き換えていれば、とくに何も問題ないわけで、そういう設定をしていないメーリングリストの場合に起こり得る可能性がありますから、逆に言うと、メーリングリストの設定をきちんとしなさいという言い方になるかもしれません。ですから、「ＮＧにしたいかも」と書いてありますが、踏み台という観点からすると、とくに拒否する理由もないという意味で OK にしてあります。
組織外からの、FROM が外で TO が内側というのは、通常の、外からやってくる普通のメールですから、これはまあ、OK にしないと問題があります。
というわけで、ここでのポイントというのは、組織外から来たメールで、TO が外側、エンベロープの受信者が組織外のメールがあった場合に、それを拒否すれば良いと思います。

中継判定の一般化

- **３つのパラメータを持つ判定関数**

  f (h, s, r) = OK, NG

  h - SMTP **接続元ホストのアドレス**

  s - SMTP **エンベロープ発信者**

  r - SMTP **エンベロープ受信者**

中継判定の一般化ということで、今、３つの要素（SMTP **の接続元ホストのアドレス、**SMTP **のエンベロープの発信者のアドレス、受信者のアドレス）、３つのパラメータの組み合わせによって、**OK **かＮＧかが決まるというので、上の関数を決めるという話に落とし込むことができると思います。**

現在の、sendmail **でやっている不正中継の防止に関して、今の段階での判断材料としては、３つのパラメータをチェックして、拒否するか** OK **とするかを決めるということになります。**

組織外からの「From 内」許可の是非

先ほども触れましたように、組織外からのメールで、FROM **が内側だったらどうするかという課題です。**

- ISP **にダイアルアップして、**From **が自組織のメールを出したい**
- **アドレスを騙られることで、踏み台になる**

  ISP **のメールサーバによっては、内部からは、付与したアドレスを** From **に持つメールしかだせない**
- **エンベロープ発信者を書き換えない** ML **の存在**

  戻ってきたメールが拒否される

これを考えないといけないわけですが、ISP **を使いながら、**FROM **が自組織のメールを出したい時にどうするかで、最近は、**POP **でがんばるぞ、というものが多くなってきていると思います。**

組織外からの「To 外」について

組織外からの TO＝外について、１つ、注意する点があります。

・ 他組織のバックアップ MX を引き受けている場合に注意！

- プライマリ MX が落ちたときに問題が発覚する
- http://www.wide.ad.jp/~motonori/mtachecker.html などで確認を

インターネット上に、プライマリの MX ホストが止まっていた時に代わりにメールを受け取って、プライマリが回復した時にメールをそちらに送ってあげるという、バックアップ MX の役割を持つホストがいくつかあります。そういうバックアップ MX を引き受けている場合に、組織外から来たメールを受け取って、さらにそれを組織外に投げるという状況が起こりますので、そういうことを引き受けているホストは注意しなければなりません。

普通、バックアップ MX は、プライマリサーバが元気なときは使われず、プライマリ MX が落ちたときに、バックアップ MX にメールが届くようになるのですが、そのときに拒否されるような設定になっていると、エラーになってメールがちゃんと届かなくなります。

そういう、まれに起こるような事態になって、初めて気がつくという状況があるので、さらに注意が必要になります。

わざわざ、バックアップのメールサーバに対してメールを送る処理をして、実際にメールの配信が拒否されるかどうかを調べるためのウェブページをつくってありますので、こういうものを使って確認して戴けると良いと思います。

実際の設定

実際の設定ということで、qmail の場合と sendmail の場合と、２つ書いてあります。

sendmail

qmail

sendmail では

sendmail も、昔は、拒否設定はできなかったのですが、最近の 8.8 や 8.9 では、拒否設定ができるようになってきています。

・ sendmail 8.8/8.9 から柔軟な拒否設定が可能

- sendmail.cf から拒否機能を呼び出す必要あり

オリジナル sendmail 付属 m4 版生成ツール

CF-3.6W 以降

- sendmail 8.9 用の sendmail.cf にはデフォルトで

拒否設定が組み込まれる (m4, CF-3.7W)

・ 設定は従来の sendmail.cf に単純に追加可能

CF-3.7W では追加部分のみの生成も可能

sendmail のバイナリ自体には、機能はあるのですが、実際に sendmail.cf から呼び出さな

いと、機能の効果は発揮されないということで、8.8 から使えるのですが、拒否設定用の sendmail.cf が生成できるようになってきたのは最近のことです。オリジナルの sendmail 付属 m4 版ツールであれば、8.9 についてくるものから、設定ができるようになっていますし、CF の場合は 8.6 以降、SMTP チェックを取り込むことによって、チェックすることができるようになっています。
昔からの sendmail.cf を使い続けている人（わりと多いかもしれません）は、sendmail のバイナリを 8.9 にして、CF で生成する、SMTP の不正中継のチェックのための部分だけを追加して、不正リレー、不正中継を防ぐという方法もあります。こういう方法を使って、オープンリレーの不正中継を防ぐことができます。

sendmail 8.8 でできること

sendmail.cf の中で、次のようなルールのところを使うことによって、SMTP のそれぞれの段階でチェックをさせることができるようになっています。

- 接続元ホストの選別

  check_relay
- エンベロープ発信者アドレスのチェック

  check_mail
- エンベロープ受信者アドレスのチェック

  check_rcpt
- 受信後のアドレスチェック

  check_compat

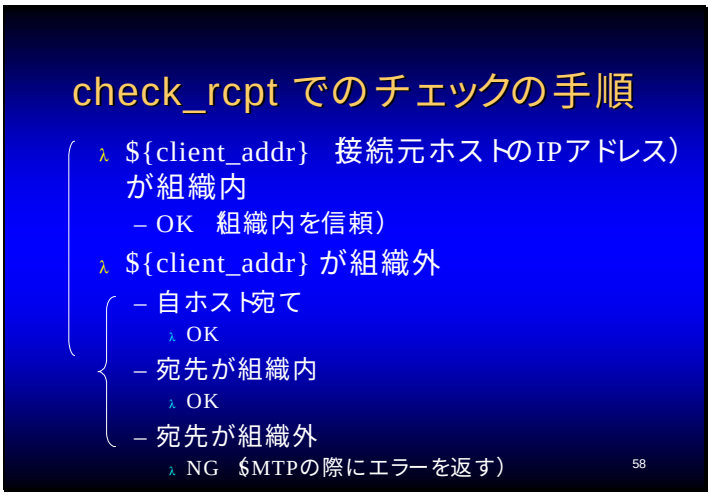
sendmail 8.9 で可能になったこと

さらに、sendmail 8.8 のところで弱かった部分に関して、sendmail の 8.9 で機能が強化されています。

- ヘッダの内容をチェックして拒否
  - 複数ヘッダの組み合わせによる判定はできない…
- 正規表現によるパターンマッチ機能
  - 存在し得ないアドレスをまとめて表現
- DNS で「存在しない」と「引けなかった」の区別
  - すぐエラーにするか、しばらくキューに保存するか
- バックアップ MX 指定の自動認識
  - 勝手に MX に指定されてしまう可能性...

正規表現によるパターンマッチングで、ヘッダがあるパターンになっているときは、そのメールを拒否するとか、そういう、不正中継ではなくて、SPAM 自体がやってくるときの拒否の機能というのを拡張、充実しています。

check_rcpt でのチェック手順



先ほどの８つの分類を、実際に、処理の流れの中でどう反映すれば良いかという説明です。

たとえば組織内からやってきたメールは、先ほどの８つの分類ですと、すべて OK に分類されていましたので、チェックの手順としては、表現的にちょっと分かりにくいかもしれませんが、client_adder に、SMTP 接続をしてきたホスト、接続元ホストの IP アドレスが入ってきますので、それを見て、組織内かどうかをまず判断をする。それで組織内だったら OK にします。組織外の場合は、それが自分のホスト宛とか、宛先が自分の組織内、つまり外から中の場合は OK にしましょう。外からやってきているのだけれども、宛先が外の場合はエラーにしましょう。この時点でエラーを出すと、SMTP の処理の途中でエラーを返すことになりますから、エラー処理を実際にするのは接続元のホストがやる仕事という形になります。

CF を利用する場合のパラメータ

CF を利用して設定する場合は、いくつか、設定のパラメータがあります。

- LOCAL_HOST_＊

f (src_host,＊,＊) = OK/NG

・ CLIENT_＊

f (src_host, from_domain,＊) = OK/NG

・ ROAM_＊

f (src_host, from_user,＊) = OK/NG

関数表現で書いてありますように、まず、１番目に関しては、接続元のホストが内側か外側かというのを識別するために使う設定、定義になります。LOCAL_HOST_＊は、そのホストが自分の組織に属するものであることを宣言するもので、そうしますと、これに属するホストからのメールの送信要求はすべて受け付ける。そうでないものに関しては、そのあとにある、他の設定に関してチェックして、OK にするかＮＧにするかを決めるという手順になります。

もう少し、細かく判断をするものとして、CLIENT_＊と ROAM_＊が用意してあります。CLIENT_＊は、自分の組織の IP アドレスと、さらに、そいつが使っていい発信者のエンベロープアドレスの２つを指定することによって、この２つのペア条件がマッチしていた場合に OK を返します。組織外のアドレスを使ってメールを送って欲しくないような場合に、CLIENT_＊を使います。逆に言うと、その組織、IP アドレスが正しくて、FROM のドメインが正しければメールを投げることができますので、ある特定の組織の外にある IP アドレスに対して、そのエンベロープの発信者アドレスが条件を満たしていれば、メールの中継を許してもいいという場合も、CLIENT*の設定を使うことができます。当然、この２つの情報、つまり、IP アドレスが扱える計算機にアカウントがある人で、さらに、この FROM ドメインを設定すれば、この中継ホストを使えるということが分かってしまうと踏み台にされてしまうので、注意する必要があります。

ドメインだけが制限されるのは、少し弱いので、ユーザレベルまで制限をしたいという場合には、この ROAM_＊が使えます。こうなりますと、IP アドレスと、さらに user@domain が等しい場合に、そのメールサーバが使えるということになります。

ALLOW_RELAY_FROM と ALLOW_RELAY_TO があります。

・ ALLOW_RELAY_FROM

f (*, from_domain,＊) = OK/NG

from_domain で偽装されると

どこからでもメールをリレーしてしまうので危険

・ ALLOW_RELAY_TO

f (＊,＊, to_domain) = OK/NG

lower MX を引き受けている場合は

定義を忘れずに

重要なのは、ALLOW_RELAY_TO です。これは、発信元の IP アドレスとか、発信者エンベロープにかかわらず、宛先となるエンベロープのアドレスが、to_domain であれば OK を出すという設定です。バックアップ MX、lower MX を引き受けている場合は、この設定をしておかないと、プライマリの MX が死んでいるときにメールを受け付けられなくなってしまいます。これが重要です。

上の ALLOW_RELAY_FROM は、FROM のところが from_domain であれば何でも OK という設定になるのですが、そうすると、IP アドレスに関係なくどこからやってくるメールであっても、エンベロープの FROM が指定ドメインであればメールが通ってしまうことになりますので、いろんな ISP を渡り歩きながら、メールサーバは自分の組織のものを使いながらやりたいという場合に便利かもしれません。しかし、SPAM の踏み台にされやすいという意味では危険ですので、これはできるだけ使わないようにしたいところです。

ネットワークアドレスの一部へのクラスマッチング(3.1W 機能)

・ Sendmail では文字列によるパターンマッチングで判定を行う

・ IP アドレスマッチングは、オクテット単位の判定

・ CIDR 時代には、もっと細かな判定が必要

・ 8.9.1+3.1W パッチではネットマスク表記が利用可能

C{Network} 200.3.4.64/27

C{Network} _MASKED_ADDRESS_MATCH_

この場合、200.3.4.64 - 200.3.4.91 がマッチする

・ マップの場合は maskedaddr map を利用

組織内かどうかというのを区別するのに、IP アドレスを使って判断するということがわりと多いわけですが、そのときある程度（クラスCとかクラスB）まるごともらっている組織であれば、設定は、sendmail ではわりと簡単になります。ただ、CIDER によってクラス C の一部しかアドレスをもらっていないという場合には、sendmail で行っているような、「.」の区切りで分けてパターンマッチングするという方法が使えないという問題があります。

sendmail へのパッチで、3.1W が出ていますが、それを使いますと、sendmail のパターンマッチングで、ネットマスク表記を使ったマッチングができるようになっています。IP アドレスの定義のところで、たとえば 200.3.4.64/27 と書くと、これで 200.3.4.64 -200.3.4.91 が、この IP アドレスの表記にマッチすることができるようになりますので、組織内の IP アドレスの範囲をさらに厳密に定義することができます。ただし、すべてこういうパターンマッチング処理をすると、処理が非常に重くなりますので、マスク表記を使いたい場合は、同じところに MASKED_ADDRESS_MATCH という、普通の IP アドレス表記では絶対に発生しないような文字列を一緒に入れておくことで、サブネットマスクをつけたパターンマッチングができるようになっています。

qmail では

- デフォルトで中継を拒否
- 中継を許可させるには

  /var/qmail/control/rcphosts に転送を許すアドレスを追加

  Lower MX を提供しているアドレスも忘れずに

qmail は、最近出てきた実装ですので、基本的に、sendmail が昔からのインターネットを支えてきたような、基本的にやらないというものがデフォルトになっています。中継を許可させるためには、コントロールのディレクトリの中に、rcphosts というファイルに転送を許すアドレスを追加することになっています。ですから、ここでバックアップ MX、lower MX になっている場合は、その MX の向いているアドレスをここに書くということで転送ができるようになります。

さらに、組織外か組織内かの区別も同様に必要になりますが、sendmail の場合は sendmail.cf に IP アドレスを定義することで、それを区別していましたが、qmail の場合は、qmail でメールを受信するときに起動する qmail-smtpd というプログラムがあるのですが、それを起動するときに RELAYCLIENT という環境変数が設定されているかどうかということによって動作が変わるようになっています。

・ ローカルクライアントからのメールの転送

qmail-smtpd の起動時に RELAYCLIENT を設定する

tcp_wrapper や tcpserver (ucspi-tcp) から起動

tcpserver を利用すれば、クライアントのアドレスを見てリレーの可否を制御することも可能

IDENT 情報も利用可能

tcprules コマンドでルールを即時反映

qmail のこのサーバは、一般に、tcp_wrapper とか、tcpserver とか、そういうものから起動されるようにして使うものですので、そうすると、これらが起動するときに設定ファイルを見て、組織内からの要求である場合は、こういう環境変数を設定しながら起動するというふうにすることで、その組織内からのメールは中継を許可するという設定ができるようになっています。

さらには、tcpserver を利用すると、もうちょっと高度な方法ができます。たとえば IDENT 情報を見て、SMP を張ってきた、その要求元のホストに対して、このコネクションを張ってきている人のユーザ名は誰ですか、というのを聞く。それに対してユーザ名が返されると、そのユーザ名に従って OK を出すといったことができます。あまり IDENT を、そういう目的で使うべきではないという話もありますので、こういうものが有効かなあと思っている人は便利かもしれません。そういうところを、色々といじらないといけない場合は、tcprules コマンドを使うことでルールが即時に反映されるのでわりと簡単です。

設定後の動作確認

不正中継、踏み台アクセスは防ぐことができるはずなのですが、組織の中で管理している人にとっては、その組織の外から来たメールが、ちゃんと不正中継の対策で拒否されるかどうかを調べることはできないわけです。組織の中からは調べることができないので、組織の外の別アカウントから、実際にメールを送ってみる必要が出てきます。

・ 組織外のアカウントを利用

・ テスト用 WWW ページを利用

http://maps.vix.com/tsi/ar-test.html

http://www.wide.ad.jp/~motonori/mtachecker.html

・ 正規のメールを拒否させないように注意！

設定ミスをしているホストへの連絡は困難

そういうアカウントを持っていれば良いのですが、ない人の為のテスト用のページが用意されています。ひとつは、SPAM の不正中継サイトのデータベースを管理する maps がありますが、その URL の付近にチェック用のページがあります。さらに、私のつくったチェック用のページもありますので、実際にチェックをしていただいたら良いと思います。設定を間違ってしてしまった場合には、当然、メールが受け取れなくなりますので、電子メ

ールで設定が間違っているというメールが届かなくなってしまいます。そういうこともありますので、慎重に行う必要があります。

メールの発信者の認証

メールの発信者の認証ですが、出歩いて、外の ISP から会社のサーバを使いたい、あるいはISP そのものに関しても、どこからでもメールを送れるというのはまずいということで、認証を使わないといけないようにしようという動きが出ています。

・ 組織外からのメールの発信の認証

POP 認証を併用

1) POP サーバにアクセス
2) アクセス元のアドレスをデータベースに登録
3) SMTP の際にデータベースを検索
4) データベースに載っている場合はアクセスを許可
5) 一定時間後にデータベースから削除

sendmail では makemap で DB を作ると sendmail の再起動が不要

ポピュラーな方法としては、POP を使って認証する方法です。接続しに来た IP アドレスとメールを送ろうとしている IP アドレスは同じはずなので、その関係を利用して、POP でアクセスしに来たときにユーザ名とパスワードが交換されますから、それで認証して OK だった IP アドレスからのメールをしばらくの時間の間だけ OK にしようというのが、これの仕組みです。

処理の順番としては、まず最初にユーザが POP サーバにアクセスをしに来る。アクセスしに来たユーザの認証が成功した場合は、そのアドレスをデータベースに登録する。そのデータベースは、IP アドレスが記述されたデータベースになりますので、SMTP が次に起こったときにデータベースを検索して、その IP アドレスが載っている場合はアクセスを許可する。そうでないときは拒否する。それも、しばらく時間がたったあとは、そのアクセスを拒否するようにデータベースからその IP アドレスを削除する。こういう仕組みを実装することによって、POP で 1 回認証してからメールを送るということができるようになります。

sendmail では、makemap でデータベースをつくるようにすると、sendmail の再起動がいりませんので、わりと簡単に CF ベースで、こういう仕組みをつくることができます。逆に言うと、POP サーバのほうで、認証に成功した IP アドレスをデータベースに登録するという部分をつくれば簡単にできます。次に出す CF は、こういう機能の入ったものを出す予定です。

SPAM とは

・ Hormel Foods Corporation の食品の名前

・ 意図しないメールの受信
　メール爆撃 (Mail Bombing)
　宣伝メール (Spam)
　　Unsolicited Commercial Email (UCE)
　　Unsolicited Bulk Email (UBE)
　　受信者の興味等に関係なく無差別に送られてくる
　　不法なものも
　宣伝コストが非常に安い!

不正中継に関しては、技術的に対応できるのですが、実際に SPAM としてやってくる宣伝メール等の、自分宛のメールは、技術的にどこをチェックしたら良いという事が簡単には言えないのでどうすれば良いかという課題について紹介します。

SPAM の予防

まず、予防の話です。

・ なぜメールアドレスが知られるのか？
　NetNews への投稿
　メーリングリストの参加者リスト
　Web に記載されたアドレス（情報収集ロボット）
・ SPAM 予防策
　user@domain.nospam といったアドレスの細工
　　機械的自動返信機能が利用できない
　　返信の際に手間がかかる
　　初心者にわかりづらい
　SPAM リストからの削除要求の返送をしない？

なぜ、自分宛にメールが飛んでくるのかという原因を考えると、たとえばネットニュースに投稿したら、そういうリストに載ってしまうかもしれませんし、何らかのメーリングリストに参加している時に、そのリストが他の人に取られて、それが SPAM の発信用に利用されたり、ウェブにいろいろ書いていて、その中にメールアドレスがあったら、そういう情報が収集されて SPAM の宛先に利用されるとかでメールアドレスが知られるケースがあります。

ですから、人目に触れるところにメールアドレスを出さないとか、あるいはニュースを出すときに、アドレスに細工をするという人もいるようですが、そういう場合に、機械的に自動返信機能が利用できないとか、返信するときに、返信しようとしている人の手間がかかるとか、いろいろと、その見返りとしてコストがかかるようにするしくみを使っている人もいます。

あるいは、メールを送ったときに、「こういうメールを今後送って欲しくない人は返事を

ください」と書いてあるわけですが、返事を出したら本当に、今後メールを送って来なくなるのかは疑わしい。逆にたくさん送られてくるようになるという不安もあるわけです。

SPAM はどこからくるか

・ 不正中継を許すホストから
・ 中継されず直接に
　　不正中継ではないので、完璧な対策が困難

不正中継に関しては、どんどん撲滅する方向で世の中が動いていますので、次第に減ってくると思うのですが、直接来るものに関しては、防ぎようがないと思います。

SPAM のフィルタリング

SPAM を排除するために、どういうところに注目すれば良いかの判断材料です。

・ 判断材料
　　発信者のメールアドレス（常連アドレス）
　　発信者のドメイン名
　　発信ホスト・ネットワークのアドレス
　　　　DNS 逆引き設定の有無？
　　ヘッダの内容
　　　　SPAM 特有の特徴をとらえる
　　本文の内容？

よく来るメールアドレス、これはエンベロープに関わらず、ヘッダの特徴がいつも来るような形のメールアドレスだったら、これを拒否しよう、このドメインはもう通信する相手がいないから拒否しよう、という方法で、虱潰しに潰していくしか方法がないのが面倒な点です。簡単にできるのは、DNS の逆引きはちゃんと設定されているかなどです。

フィルタリングの問題点

虱潰しでやっていく時に、みんなが同じ事をやっていくのは大変なので、みんなが共有して利用できるブラックリストがあると便利ということになります。

・ 定義したホストからのメールが届かなくなる
　　完全に拒否せず、マークをつけて受信者側で分類する方法もある
　　リレーでない(その組織からの)メールを制限しない工夫
・ 未知の踏み台が利用されると効果が下がる
・ ブラックリストのメンテナンスが面倒
　　->MAPS RBL などの参照

メーリングリストと SPAM 対策

・ 最近のメーリングリストサーバの標準的(?)な機能
　登録メンバ以外からのメールを拒否
　　投稿、アーカイブ検索と入手
　　メンバリストの入手
　ヘッダの宛先に ML のアドレスがなければ拒否
・ MTA とは独立したサーバプログラムで対応
　プログラムの機能整備が進んでいる？

SPAM ホストのデータベース

ブラックリストに関しては、SPAM のホストのデータベースということでいろんなものがあります。

・ MAPS RBL
・ ORBS
・ DUL

不正中継をするホストのデータベースから始まったものがありますが、だんだんと、SPAM そのものを防ぐためのものも出てきているようです。このへんの仕組みを、順番にご説明します。

MAPS RBL

一番最初に出てきたデータベースとして、MAPS RBL があります。

・ MAPS ＲＢＬ（MailAbuseProtectionSystem RealtimeBlackholeList）

http://maps.vix.com/rbl/

・ DNS で 4.3.2.1.rbl.maps.vix.com に対する A レコードが存在すれば拒否する
　IP アドレス 1.2.3.4 からの接続要求の場合
　DNS 参照のテスト用アドレス： 127.0.0.2
　　2.0.0.127.rbl.maps.vix.com
・ BGP もやっている

ドメイン名を DNS で聞きに行く時に、アクセスして来たホストの IP アドレスを逆順にします。逆順というのは、DNS を使って IP アドレスを逆引きするときの基本的なやり方ですので、それを踏襲しているのですが、そういう形で逆順に並べて、rbl.maps.vix.com というのをつけて DNS を検索すると、もしそこにＡレコードが定義されていれば、アクセスしてきたホストは不正中継をする悪いやつだということになります。このように、メールを受け取ろうとするたびに、チェックを毎回行って、Ａレコードが存在すれば拒否するという仕組みを sendmail や MTA に作ることによって、Open Relay からメールをもらうことを防ぐことができるようになっています。

ORBS

最近、ORBS というものが出てきています。変更ページということで、これは資料とは少し変更されているという意味ですが、Open Relay Blocking System というのがありました。

- Open Relay Blocking System
- http://www.dorkslayers.com/orbs/
- リストの参照方法は、MAPS RBL と同様

  4.3.2.1.orbs.dorkslayers.com に対する A レコードの存在を確認

ORBS は、プローブベースでいろんなところをスキャンして、Open Relay を見つけたらデータベースに登録していくということをやっていたのですが、ちゃんと Open Relay 対策をしたのに、そのチェックが、何度も何度も繰り返しやってくる等の運用方法に批判が多くて、現在は、サービスを停止しているようです。

sendmail で ORBS を見てチェックする機能を既に入れてしまっているところは、DNS でこのデータが引けなくなってしまっていますので、メールが受け取れなくなっているかもしれません。そういうところは、この ORBS のチェックをはずすことが必要になります。ですから、こういうのを安易に採用して廃止されたらそれに従って、こちらも作業しないといけない事は面倒と言えば面倒なので考えものですが、こういう方法も提供されています。

DUL

DUL は、今度はダイアルアップユーザのリスト、ダイアルアップ空間を定義したデータベースになっています。

- ORCA Dial-up User List

  - ダイアルアップユーザからの直接のメールを拒否するためのもの
- http://www.orac.bc.ca/dul/
- リストの参照方法は、MAPS RBL と同様

  - 4.3.2.1.dul.orac.bc.ca に対する

    A レコードの存在を確認

  - ゾーン転送も可能

参照方法は MAPS RBL 等と一緒で、こういうふうに引きますと、ダイアルアップサービスをされている空間に関してデータが定義されているので、そころから直接 SPAM が飛んでくるのを防ぐことができるサービスになっているようです。

SHUB

最近、こんなものもありますが、まだ詳しく調べていないので、登録方針がどういうふうになっているのかよく分からないのですが、リストの参照方法は MAPS や ORBS と同じようです。さらに、新しいものとしては何かというと、IP アドレスだけではなくて、エンベ

ロープのドメインとか、メールアドレスとか、を使って検索しに行くことによって、SPAMのチェックがもっといろんな局面から検索することができるようなサービスもあるようです。設定、データの収集をどういうふうにしているのかよく分からなかったのですが、サービスとして、このようなデータを皆んなで共有するのは、ある意味で有効だと思います。

常連 spammer をローカルに登録

共有のデータベースを使う以外にも、自力でがんばる方法もあります。

- sendmail (CF)

  SPAM_LIST*

- qmail

  control/badmailfrom

もちろん手動でメンテナンス

sendmail の（CF）の場合は SPAM_LIST*を使う、qmail の場合は badmailfrom に定義することによって、自分のところでローカルに SPAM を防ぐということもできます。

発信者アドレスに関するチェック　～ ごみメールをへらすために ～

ごみメールを減らすためのさらなる工夫です。

- **返信無用の SPAM は、発信者アドレスの不正が多い**
  - @domain **が省略されているもの**
  - user@host **形式の(FQDN でない）もの**
  - DNS **に登録されていない（返送不能な）もの**

    たまたま引けないものは受信エラーにしない（一時的拒否）
  - **偽造がみえみえなもの**

    ユーザ部が数字だけ、長すぎるもの

    正規表現でチェック

でも、アドレスが騙られると見分けられない

先ほど説明しました、DNS をチェックして、返信できないものは当然、受け取る必要もないと判断して拒否するというのもありますし、@domain が省略されているとか、FQDN になっていないメールアドレスとか、そういうもの。あるいは偽造が見え見えなものというものは、正規表現の機能を使ってチェックすることができますので、そういう形で、ある程度対策をすることができるわけですが、アドレスが偽られているという事例も最近出始めているようですので、さらにチェックする方法を探すというのも、ひとつの課題になりつつあります。

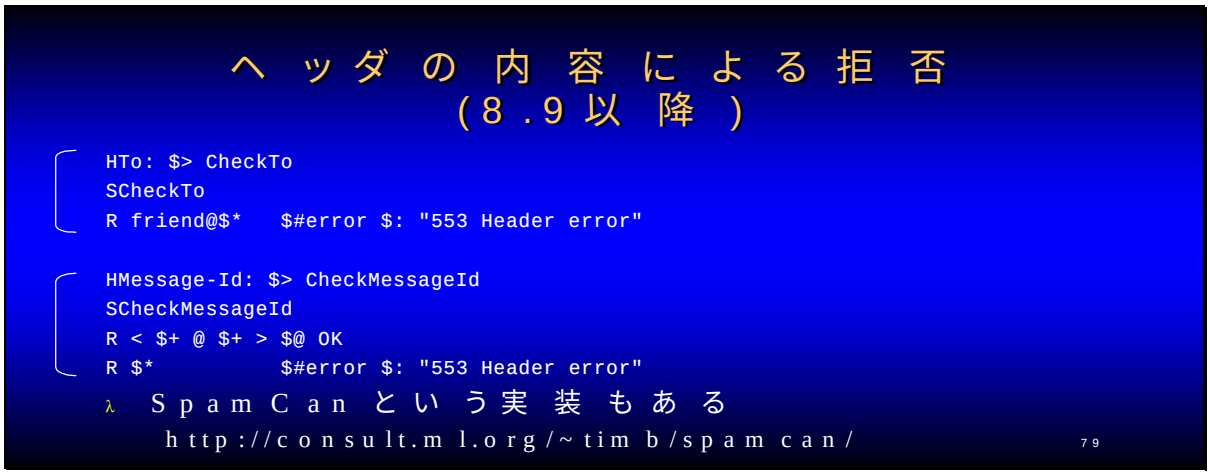
正規表現による拒否 (sendmail 8.9 以降)

正規表現の例をあげております。

```
Kcheckaddress regex -a@MATCH
        ^([0-9]+<@(aol | msn)¥.com | [0-9][^<]*
<@juno¥.com | .{10}[^<]+<@aol¥.com)¥.?>
R $+            $: $(checkaddress $1 $)
R  @MATCH       $#error $: "553 Header error"
```

たとえばＡＯＬとかＭＳＮで、数字だけユーザ ID が存在するようなアドレスは拒否するとかができます。

ヘッダの内容による拒否（８．９以降）



ヘッダ自体に、たとえば friend@＊＊＊**とか、そういうのが** to **に書いてある** SPAM **というのも割と飛んできますが、そういうものを拒否する。あるいは、メッセージ** ID **に@マークが含まれていないようなメールを拒否する。そういうこともsendmailの設定でできますし、さらに、これを組み合わせた高度な設定、拒否の設定を、**SpamCan **を使うことによってできます。**

苦情窓口の開設

運用上の問題ですが、不正中継とか SPAM の嫌疑などがあったときに、報告を受ける窓口というものがいくつかあります。

- abuse@domain の作成
  - (RFC2142 Mailbox Names for Common Services, Roles and Functions)
- domain@abuse.net
  - Network Abuse Clearinghouse (http://www.abuse.net)

RFC2142 を見ますと、abuse@domain をつくっておいて、苦情を受け付けるようにしておきましょうという話が載っていたり、サービスとして、abuse.net がありそこを経由してメールを送ると、ポストマスタにしろ、そのドメインに対する連絡先のところに自動的に転送してくれる仕組みがあります。

さらなる SPAM 対策

さらなる SPAM 対策ということで、今後の課題のようなものを挙げてみました。

- エンベロープ発信者が <> のものをどうするか
  - Mailer_daemon からのメール
- 発信者を実在のアドレスに偽装されたら...
- 最終的には個人レベルで対処するしかない？
  - 自分のアドレスがヘッダにあるものを通す
  - 参加しているメーリングリストだけ通す

    procmail などの利用

エンベロープの発信者が空っぽのものが、今現在、流れています。これは Mailer_daemon からのメール、つまりメールがエラーを起こしたときに、そのレポートメール、エラーメールは、このように発信者のところを空っぽにしたメールが送られるようになっています。これは、エラーメールに対するエラーメールが、何度もやりとりされることを防ぐために、エラーメールが必要ないということを指定するために使われています。そうしますと、このエンベロープアドレスは、結局、どこからやってきても、みんな受け取るということが要求されるわけですから、逆に SPAM を投げる人が返事をもらう必要がない場合に、発信者のエンベロープアドレスをこれにしてメールを投げてしまえば、どこでも通れてしまうことになってしまい、これも危険な要素のひとつになってしまいます。あるいは、これはシステムの弱点ではあるのですが、実在の発信者のアドレスを偽装された場合も困るので、それに対してどういう対策をしていくのかというのが、今後の課題になります。さらに、最終的には個人レベル、ホストレベルで対処するのもありますが、ある人に対しては SPAM なドメインであっても、別の人に対してはちゃんと受け取りたいドメインかもしれないので、そういうことを考慮していくと、最終的には個人レベルで設定していくしか方法がな

いのかなと思ったりします。

さらなる技術的対策

- メールの内容をチェックする MTA も
  ファイアウォル製品など
- 署名を普及させる
  署名の無い物は捨てる
- MTA 認証？

MTA の認証を、もっと充実させる、あるいは電子署名をもっと普及させるのも、手段のひとつとして、技術的な対策として有効かもしれません。

メールの内容をチェックする MTA もあるようですが、Firewall 製品で、必ずメールがそこを通るときにメールの内容を見て、特定パターンのメールは拒否するというふうに設定しておくと、それを捨ててくれるのもあります。そういう対策も必要かもしれません。これらは、自分に届くメールの中でいらないメールをどういうふうに減らしていくかという事で、これからの長い戦いになっていくような気がしますが、技術的に対処できるものに関しては対処していこうということで、いろいろな方法が用意されてきています。

５．Q&A

Q:POP 認証を利用した SMTP の件ですが、もう少し詳しいことを書いているようなページはありませんでしょうか。

A：ここには書いていませんが、ご質問の内容については、綾村さんが詳しく、情報も集めていらっしゃいますので、綾村さんのホームページ、「www.ayamura.org」が参考になると思います。POP に関する話題もあったと思います。私も参考にさせてもらっています。

＜　以上 ＞