DNS&Mail

安藤 一憲

(トランス・コスモス(株))

1999 年 12 月 15 日

Internet Week 99 **パシフィコ横浜**

（社）日本ネットワークインフォメーションセンター編

目次

1 概要

この講演では、セッションは 2 部構成になっています。前半は DNS、後半はメールに関しての話題です。前半の DNS については、まず DNS のしくみ、つまり階層構造を持った分散データベースの内容を説明し、named で使用されているファイルの設定、bind-8.2.2 での新機能ならびに設定の説明をします。後半はメールの配送モデルについての説明をし、MX 配送と static 配送の使い分け、さらにメールの抱える問題について、SPAM 対策、ウイルス対策、チェーンメール対策など、メールサーバを実際に運用する際に直面すると思われる問題点についてまとめています。

2 DNSのしくみ

DNS は、ドメインネームサービスを行っているシステムで、インターネットの基幹部分を構成しているソフトウェアです。そして、唯一インターネットで成功している分散型のデータベースと言うことができます。インターネットの各ドメイン毎にデータベースが分散しています。DNS は指定されたドメインのリソース情報へ到達する鍵となるものであり、順次ネームサーバ(NS)を手繰ってデータを引くしくみを持っています。IP アドレスの付け替え時やドメインの変更時は、データが大幅に変わりますので、データベースの取り扱いには充分な注意が必要です。DNS は基本的なソフトウェアですので、これが落ちるとほぼすべてのサービスに影響します。メールはもちろん、Web の URL の解析も影響を受けます。したがって、DNS サーバが落ちると、サーバリソースへのアクセスができなくなり、そこのドメインのインターネットサービスはできないということになります。したがって、ネームサーバは死守すべきであり、多重化して運用すべきです。

2.1 黒子として働くDNS

ネームサーバがどのように動くかを説明します。

DNS は FQDN（Full Qualified Domain Name:省略しない形のドメイン付きホスト名)と IP アドレスを対応付けます。FQDN から IP アドレスを対応付けることを「正引き」、IP アドレスから FQDN を対応付けることを「逆引き」と呼びます。

メールサービスの中で、メールを送信する場合には、FQDN から IP アドレ

スを「正引き」して、そこに接続します。着信メールを記録する場合には、相手マシンの IP アドレスから FQDN を「逆引き」して記録する、という動作をします。

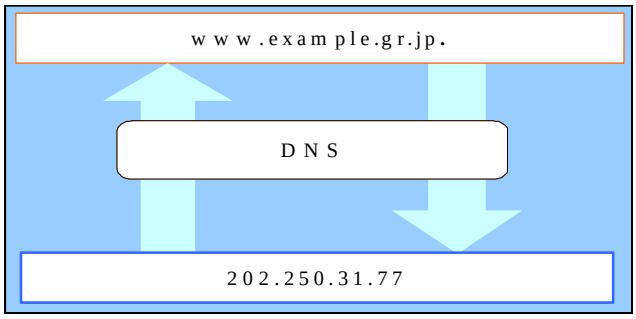

図 1:「正引き」と「逆引き」

● ドメイン名の階層構造

ドメイン名は、ピリオドを仲立ちにして分かれています。jp の後のピリオド(.)は、DNS では特別な意味を持っています。このピリオドは、ルートを表わしドメイン名の一番おおもとを意味しています。メールアドレスでは、ドメインの一番最後にピリオドが付くことはありませんが、DNS のデータを記述する時にはこのピリオドを忘れないで設定ファイルに記述してください。ピリオドを忘れると、named(ネームサーバのプログラム)がこの後にもドメインが続くものと勘違いします。

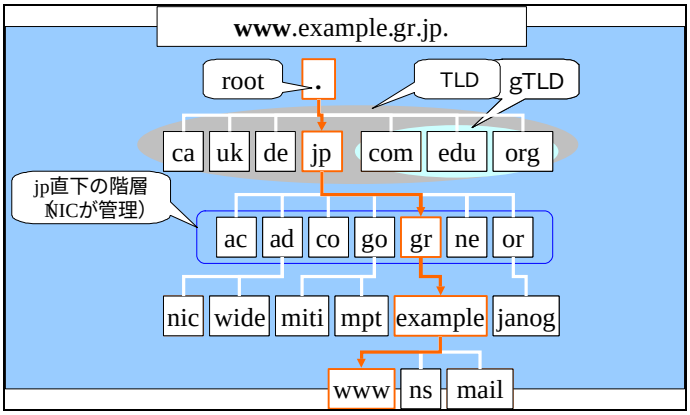

図 2:ドメイン名の階層構造

一番上は、ピリオドで、ルートです。ルートの直下にくるドメインを TLD(Top Level Domain:トップレベルドメイン)と呼びます。そこには、ca、uk など

の別ドメイン（現在 243）および、US が管理している com、edu などの gTLD(generic TLD)があります。さらに、TLD の下にセカンドレベルドメインと呼ばれる第 2 層のドメインがあります。ac、ad、co、go、gr、ne、or などで、日本の場合 JPNIC が管理しています。第 3 層としては、nic、wide などがあり、その後にマシン名がきます（さらにドメインが来る場合もあります）。DNS はこのような階層構造を持っています。

2.2 階層ごとの担当サーバ（NS）

図 3 の階層構造が示すように、それぞれデータを持つサーバは分散されています。たとえば、ルートを持つサーバは、ピリオド（.）から jp に行くルートを教える役割を持つサーバで、TLD は全部で 13(a から m)存在し、最後の m が日本で動いています。DNS では、ドメイン名(たとえば、www.example.gr.jp)の最後から最初の方に各階層のサーバ情報を手繰っていきます。したがって、www.example.gr.jp の場合には 4 つのサーバが関与しているデータベースとなります。

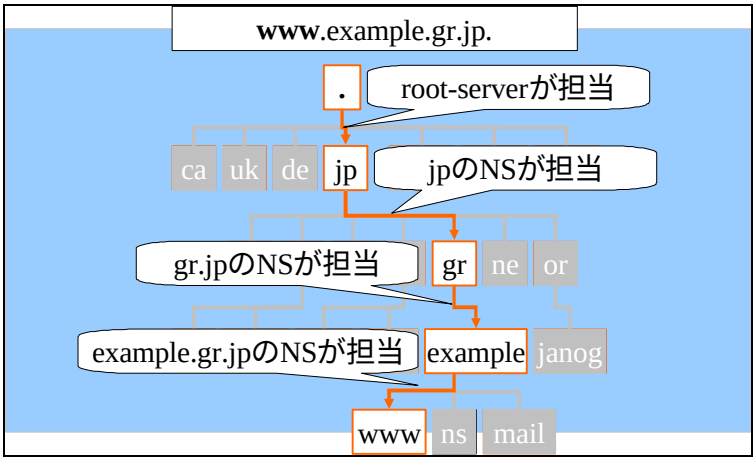

図 3:階層毎の担当サーバ

● DNS の検索のしくみ

www.example.gr.jpを例に検索のしくみを説明します。まず、① ルートサーバに jp を聞き、jp のサーバを得ます。② jp サーバに jp ドメインのネームサーバを聞き、③ jp ネームサーバに gr.jp ドメインのネームサーバを聞き、④ gr.jp のネームサーバに example.gr.jp を聞くという具合に手繰ります。そして、⑤ example.gr.jp のサーバにwww.example.gr.jpのアドレスを聞くと、IP アドレスが返ってきます。サーバの IP アドレスを得て実際にアクセスすることができます。

ここまで４つのサーバを経由していますから、途中のサーバの１つでもトラ

ブルを抱えていると、アクセスできません。実際の運用では各階層に複数のサーバが関与していて、１つでも生き残っていればアクセス可能なように構成されています。

3　namedの設定

3.1　ドメインを支えるネームサーバ群

実際にドメインを支えるネームサーバ群の関係をもっと詳しく説明します。

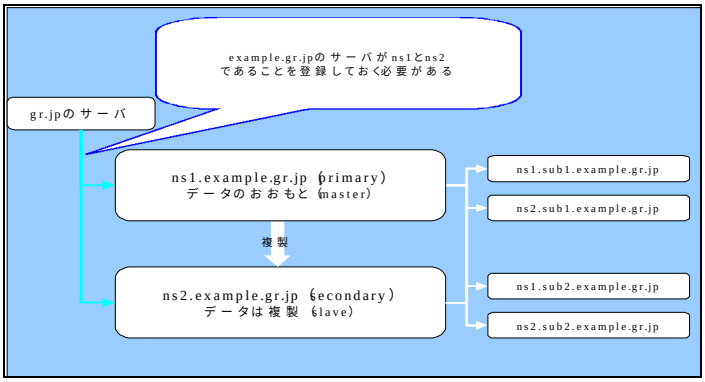

図 4:ネームサーバ

たとえば、example.gr.jp の情報は上の gr.jp サーバが持っています。上のサーバは常に下のサーバを知っているというのが DNS の「みそ」です。この例では、プライマリ n1 とセカンダリ n2 で多重化し、合計 2 台のサーバを登録しています。プライマリにデータのおおもとのマスターを持ち、セカンダリにマスタの複製のスレーブを持っています。多重化によってシステムを冗長化しています。

3.2　localhostの設定（正引き）

localhost の正引きの具体的な設定を説明します。
ns1.example.gr.jp(primary)のマスタの設定です。

```
zone “localhost”{
        type master;
        file “localhost.zone”;
};
```

ゾーン(zone)は DNS の管理区分を指します。localhost の zone という指定なので自分自身に対する zone の情報を意味します。

3.3 localhostの設定（逆引き）

設定はもう１つあります。 逆引きの設定です。ns1.example.gr.jp(primary)マスタの設定です。IP アドレスから localhost という名前を得るためのものです。

```
zone”0,0,127.in-addr.arpa”{
        type master;
        file”localhost.rev”;
};
```

zone”0.0.127.in-addr.arpa”という指定で、IP アドレス(127.0.0.1)を逆に並べたものが in-addr.arpa の前に来ています。これは、逆引きの約束事です。逆引きなので、ファイル名拡張子が.rev となっています。localhost.rev ファイルに「1 IN PTR localhost」と先頭に 1 が定義してあり、結局、論理的には”1.0.0.127.in-addr.arpa”が目的ローカルホストの逆引きアドレスになっています。

3.4 root-serverの設定

次に、サーバが他ドメインのデータを取りに行くための設定を見てみます。ネームサーバは他のネームサーバと連動してデータをやり取りしていますから、自分のところのドメイン設定に加えて、そのための様々な設定をしなければなりません。インターネット上にあるサーバはすべてルートサーバを知らなければなりません。この設定が、type hint　file”root.cache” に書いてあります。

ns1.example.gr.jp(primary)サーバの設定です。

```
zone “. ”{
        type hint;
        file”root.cache”;
};
```

すべてのネームサーバは、最新のものを設定しておく必要があります。“root.cache”用のファイルは ftp(file transfer protocol)で配布されていますので、bind という named のソフトウェアを作っているところの ftp サーバからファイル(ファイル名:named.root)を得ることができます。最新のファイル

をインストールすることが必要です。

3.5 masterサーバの設定

example.gr.jp のデータ設定方法の説明に入っていきます。マスタサーバは次のように設定します。ゾーンファイルは、type が master(マスタサーバ)で、ファイル名が"example.zone"となります。マスタ（ゾーン）ファイルの内容詳細については後で説明します。

ns1.example.gr.jp(primary)の設定です。

```
zone"example.gr.jp"{
        type master;
        file"example.zone"
};
```

3.6 slaveサーバの設定

マスタサーバから複製を受ける方の設定は次のとおりで、type slave と指定します。マスタの場所は、masters に IP アドレスを設定します。

ns 2.example.gr.jp(secondary)の設定です。

```
zone"example.gr.jp"{
        type slave;
        file"zone/example.zone";
        masters {202.250.31.148;};
};
```

したがって、masters に指定された IP アドレスのサーバから、zone ディレクトリの example.zone というファイルの情報を複製して、自分のところの zone ディレクトリに持つということを示しています。

4 マスタ(ゾーン)ファイルの設定

図５がマスタ(ゾーン)ファイル example.zone です。見た目は localhost と似ていますが、当然中身は異なります。

```
; $ORIGIN example.gr.jp.
$TTL 86400
@       IN      SOA  ns1.example.gr.jp. hostmaster.ns1.example.gr.jp. (
                     2000010101      ; Serial
                     3600            ; Refresh
                     900             ; Retry
                     360000          ; Expire
                     86400 )         ; Minimum
        IN      NS      ns1.example.gr.jp.
        IN      NS      ns2.example.gr.jp.
        IN      MX   10    mail-g1.example.gr.jp.
        IN      MX   20    mail-g2.example.gr.jp.
; hosts data
ns1             IN    A   202.250.31.148
ns2             IN    A   202.250.31.149
mail-g1         IN    A   202.250.31.150
```

図 5:マスタ(ゾーン)ファイル（一部）

4.1 ゾーンファイルの内容

① **$TTL**

TTL(Time To Live)**は、有効期間を表します。単位は秒です。この記述がないと** SOA(the Start of a zone Of Authority)**の** Minimum **の値が使われます。ただし、**$TTL **の記述がないとワーニング（警告メッセージ）が多数出て面倒なので記述するようにしてください。**

② SOA RR

```
@   IN  SOA  ns1.example.gr.jp. hostmaster.ns1.example.gr.jp.(
         2000010101      ;Serial
         3600            ; Refresh
         900             ; Retry
         360000          ; Expire
         86400  )        ; Minimum
```

SOA RR(**リソースレコード**)**です。**SOA **は、**the Start of a zone Of Authority **の略で、ゾーンファイルの先頭にあり、マスタファイルの原本のありかと、管理者、設定パラメータを記入します。**

@: 最初の@はカレントオリジン(current origin)**を意味します。**named.conf **で指定したドメインで、例では** example.gr.jp **を指し、@1つで指定しています。**IN SOA **は、インターネットの** SOA **レコードという意味で、**ns1.example.gr.jp.**は原本**(master)**のあるサーバで、**hostmaster.ns1.example.gr.jp.**はメールサーバの管理者メールアドレス** hostmaster@ns1.example.gre.jp**を指します。メールアドレスではありますが** jp **の後のピリオド**(.)**を忘れないように注意してください。ゾーンファイルの管理者メールアドレスに関しては@もピリオドで代用することになって**

います。管理者のメールアドレスは hostmaster にします。

Serial：更新された時、更新を他のネームサーバに通知する番号です。Serial 自体は符号なしの 32 ビット整数という制限があります。したがって、4294967295 が上限で、これを超えるとゼロに戻り、「Serial の 2000 年問題」と言われています(4295 年問題？)。たとえば、991215001 から 000101001 への直接変更は数字が減少するので許されていません。これを避けるために、RFC1982 では、上限値の半分以内（2147483647 以内）の増加を 2 回繰り返すことで 00 に戻すことが可能になります。足し算を 2 回繰り返すというわけです。こうすれば、991215001 から 2000121501 に変更できます。

Refresh：セカンダリサーバからプライマリサーバのデータの Serial 値更新があったか否かをチェックする間隔です。単位は秒で、3600 秒は 1 時間毎ということになります。大きくドメインが変更された場合には、時間的なパラメータを小さくして早目に更新させるという使い方をします。

Retry：セカンダリサーバがプライマリサーバにアクセスできなかった時にリトライする間隔（秒）です。900 は 15 分を表します。

Expire：セカンダリサーバがプライマリサーバにアクセスできなかった時にそのデータをセカンダリサーバが削除するまでの時間(秒)です。360000 は、100 時間を表します。

Minimum：当初は、TTL の値を指していましたが、$TTL が出てから意味が変わってきました。現在は negative response における最小 TTL の値を指します。negative response は、bind8 で出てきた考えで、あるドメインのサーバを引いた時、それが存在しないことを一時記憶しておくことを言います。したがって、Minimum の値はネームサーバが「相手が存在しない」ということを覚えておく時間です(RFC2308 section4 に記載されています)。

③ NS(Name Server) RR

これからが、本当の意味でドメインのデータ(RR)になります。ネームサーバを指定します。

```
      IN    NS    ns1.example.gr.jp
      IN    NS    ns2.example.gr.jp
```

example.gr.jp のプライマリ(ns1)とセカンダリ(ns2)のネームサーバを記述しておきます。

④ MX(Mail eXchanger）RR

そのドメインのメールサーバを記述します。

```
     IN     MX   10     mail-g1.example.gr.jp
     IN     MX   20     mail-gr2.example.gr.jp
```

@example.gr.jp のメールアドレスに対するメールの送り先サーバを指定します。10,20 という数字はプリファレンス（優先度）で、0～32767 の値が指定でき、数字の小さい方が最初にアクセスされます。

⑤ A(Address) RR(リソースレコード)

```
ns1                     IN    A    202.250.31.148
ns2             IN     A    202.250.31.149
mail-g1         IN     A    202.250.31.150
mail-g2         IN     A    202.250.31.151
www             IN     A    202.250.31.77
proxy           IN     A    202.250.31.78
```

A レコードと呼ばれるサーバ名を記述します。ホスト名称と IP アドレスの対応を記述します。後述するように、NS RR,MX RR に記述したホストのアドレスを必ず A レコードに記述すると、アクセス効率が向上します。このレコードの右辺には CNAME と呼ばれるエイリアス(別名)のサーバ名を書いてはいけません。

⑥ CNAME(Canonical Name）RR(リソースレコード)

```
cache           IN     CNAME    proxy.example.gr.jp.
```

別名を設定します。CNAME にはホストの別名(エイリアス)を記述します。ここに記述した cache.example.gr.jp は、proxy.example.gr.jp.の別名だと宣言しているものです。

⑦ Delegation

```
; subdomain sub1 (delegation and glue)
sub1            IN      NS      ns1.sub1.example.gr.jp.
                IN      NS      ns2.sub1.example.gr.jp.
                IN      NS      ns2.example.gr.jp.
 ns1.sub1       IN      A       202.250.31.33
 ns2.sub1       IN      A       202.250.31.34
; subdomain sub2  (delegation and glue)
 sub2           IN      NS      ns1.sub2.example.gr.jp.
```

sub1 というサブドメインに問い合わせが来たら、そのサブドメインの担当ネームサーバが、ns1.sub1.example.gr.jp.　　で、もう一台は ns2.sub1.example.gr.jp.という指定の例です。ネームサーバは何台あってもかまいません。2 台の場合はこのように記述します。

デリゲーションとは、権限委譲のことです。サブドメインがある場合、管理機能のデリゲーション（権限委譲）をしなければなりません。枠内がその記述です。sub1 の担当ネームサーバが ns1.sub1.example.gr.jp.と ns2.sub1.example.gr.jp.に権限委譲されていると宣言しています。IN NS ns2.example.gr.jp.の指定で、ns2.example.gr.jp.を sub1 のセカンダリとしても引けるように設定します。先程の説明と同様に、A レコードを直下に記述すると、1 回 named を引くだけで A レコードの情報がおまけで返されるので少し高速になります。

⑧ sub1 の secondary にする設定

ns2 のサーバで次のように記述すると、ns2 のネームサーバはデータをコピーして自分のファイルに持つことができます。つまり上下関係に関係なく複製を持っておくことができます。

ns2.example.gr.jp(secondary)の設定

```
zone ”sub1.example.gr.jp” {
        type ｓｌａｖｅ;
        file ”zone/sub1.exapmle.zone”;
        masters {
             202.250.31.33;
             202.250.31.34;
    };
};
```

⑨ 逆引きアドレス

```
202.250.31.0/24
      ↓
31.250.202.in-addr.arpa.
```

アドレスの順番が逆になることが鍵になります。202.250.31.0/24 はネットワークアドレスになります。/24 は上の 3 つ(24 = 8bit*3)の部分が共通で、ゼロになっている部分、IP アドレス 0-254 までの一群のアドレスが入るということを表わしています。これがすべてドメインの中に収まります。逆引き用のドメインが zone”31.250.202.in-addr.arpe”を指定します。逆引き用ファイルですから.rev 拡張子を付けています。

ns1.example.gr.jp(primary)の設定

```
zone ″31.250.202.in-addr.arpa″ {
        type master;
        file ″202.250.31.rev″;
};
```

⑩ネットワーク名のマッピング

```
0       IN      PTR    example-net.example.gr.jp.
        IN      A      255.255.255.0
example-net.example.gr.jp.    IN   PTR   0.31.250.202.in-
addr.arpa.
```

PTR RR ですが、ネットワーク名のマッピングを表わし、ネットワークにアドレスを振ることができます。A でネットマスクを指定することになっています。逆引きの逆引きまで可能なようなデータ構造を持ったものをネットワークに対して指定することができます。利点として、netstat コマンドを打った時に、IP アドレスだけでなく、名前まで表示することができるようになります。

⑪ Classless IN-ADDR.ARPA delegation

```
; 202.250.31.32/27 delegation
32/27    IN      NS      ns1.sub1.example.gr.jp.
         IN      NS      ns2.sub1.example.gr.jp.
$GENERATE 33-62 $ CNAME $.32/27.31.250.202.in-addr.arpa.
```

逆引きアドレスを権限委譲する場合を説明します。IP アドレスの一部をサブドメインのサーバに割り振り、サブドメインのサーバ側で逆引きデータを管理して欲しい時に使用します。202.250.31.32/27 を ns1.sub1.example.gr.jp に権限委譲する設定です。31.32 のアドレスから 32 個分のアドレスということになります。その分のアドレスを ns1.sub1 のサーバに権限委譲する指定になります。逆引きですから、先頭の数字は 1 から 254 までの数字かと思われるかもしれませんが、権限委譲に限って 32/27 のような書き方が許されています。32/27 ですから、ネットマスクが 27 ビットで、32 番から 64 番までのアドレスを ns1.sub1.example.gr.jp.と ns2.sub1.example.gr.jp.にデリゲートするという指定です。

$GENERATE は、$ CNAME pc$.example.gr.jp.ステートメント中の$に、$GENERATE 直後の数字範囲(33,34,35,…61,62)の数字を組み込んだリソースレコードを機械的に生成するものです。

権限委譲を受ける側の指定です。

```
zone ”32/27.31.250.202.in-addr.arpa” {
            type master;
            file “zone/202.250.31.32.rev”;
};
```

おおもとが次のようなデータを持っています。$ORIGIN で指定して、数字に応じた PTR が並んでいます。

```
; $ORIGIN 32/27.31.250.202.in-addr.arpa.
$TTL 86400
@       IN   SOA   ns1.sub1.example.gr.jp. hostmaster.example.gr.jp. (
                2000010101      ; Serial
                3600            ; Refresh
                900             ; Retry
                360000          ; Expire
                86400 )         ; Minimum
        IN      NS      ns1.sub1.example.gr.jp.
        IN      NS      ns2.sub1.example.gr.jp.
32      IN      PTR     example-sub1.example.gr.jp.
        IN      A      255.255.255.32
33      IN      PTR     ns1.sub1.example.gr.jp.
34      IN      PTR     ns2.sub1.example.gr.jp.
35      IN      PTR     mail.sub1.example.gr.jp.
36      IN      PTR     www.sub1.example.gr.jp.
```

202.250.31.33 の逆引きをまとめると次のようになります。

```
33.31.250.202.in-addr.arpa
        ↓       CNAME on ns1.example.gr.jp
33.32/27.31.250.202.in-addr.arpa
        ↓       PTR on ns1.sub1.example.gr.jp
ns1.sub1.example.gr.jp
```

202.250.31.33 を named に問い合わせます。すると、権限委譲している方に CNAME を聞き、さらに PTR を問い合わせると求まります。

1 回 CNAME を使ってアドレスを変更し、変更した後の形のデータを実際のサブドメインのサーバが持っているのが Classless IN-ADDR.ARPA delegation です。

4.2 マスタ(ゾーン)ファイルについての注意点

マスタファイル(master file)更新時の管理上の注意点は次のとおりです。

・データを更新したら、必ずその master file の Serial を増加させる。

secondary は更新を Serial でチェックしているので必ず必要です。

・変更が終わったら、named を restart する。

ndc restart で起動すると、新しいデータが反映されるようになります。

bind8.(現在最新)では更新要求がマスタからスレーブ（セカンダリ）に伝えられ、セカンダリのデータも更新されるしくみ（ダイナミックアップデート）が実装されています（RFC1996)。

4.3 bind8の新しい仕様

bind8 の新しい仕様を次に記述しておきます。

- RFC1995：IXFR
  zone ファイルの変更部分だけを転送して更新の効率を上げます。
- RFC1996：DNS NOTIFY
  master ファイルの更新を secondary に能動的に通知します。
- RFC2065：DNSSEC
  DNS データの信頼性を向上させるためのもので、３つの目的に使われます。

-key distribution	データへの電子署名附加
-data origin authentication	認証
-transaction and request authentication	データを引く動作への認証

5 DNSとメール

DNS とメールの接点は、前半の DNS の説明に出てきた MX レコードです。ゾーン情報の中にあり、ドメインのメール受け取りサーバを指定する MX リソースレコードです。

5.1 DNSとメール

メールに user@example.gr.jp というアドレスがあった場合、このアドレスから実際の配送先サーバを見つける方法がテーマになります。手がかりは@以下のドメイン名の部分です。DNS は、@以下の MX レコードを調べます。配送には、MX とサーバの A レコードが必要になります。「MX 直下に A を書いておく」という話をしました。こうすると MX を参照したときに MX と A の情報が同時に返ってくるので、効率が良くなります。そうしないと片方ずつ聞くことになり、2 回聞くことになりますので効率が落ちます。MX と A レコードを同じ named が知っているというのがベストです。これが、DNS

を使う時の効率向上の秘訣となります。

・ MX は CNAME を使用しない

CNAME は仮の名前なので、正しい名前を手繰るのが正しい処置です。しかし、IETF(Internet Engineering Task Force：インターネット技術委員会)は CNAME を手繰らない方向に動いており、Sendmail はオプションにする模様です(O DontExpandCnames=False)。

・ ワイルドカード MX は使用しない

ワイルドカード MX の仕様は、「*.ドメイン名」 という MX レコードリソースを記述し、あるドメイン以下にあるすべてのサーバ名（あるいはサブドメイン名）に対応するメールアドレスのメールを、すべて１個の MX メールアドレスを送るための指定方法です。ワイルドカード MX を使うと、そのドメイン以下のメールをすべて 1 個所に集めることができ、簡単に定義できて便利です。

しかし、この指定が有効に機能するのは限定された環境下に限定されます。Sendmail の README には、有効に機能する場合の環境としては、ファイアウォールが１つだけ外に出ていて、残りのサーバはすべてファイアウォールの内にある場合だと記載されています。その時ワイルドカード MX で MX をすべてファイアウォールに向けておけば良いのです。それ以外に、A が優先されるかワイルドカード MX が優先されるかという問題もあり、メールがどこに飛んでいくか判らない場合もあるのです。このような理由から、ワイルドカード MX は使わないようにしましょう。

6　メール本体とエンベロープ

メールにはメール本文とエンベロープとが必ず付いています。この両者の区別を説明します。

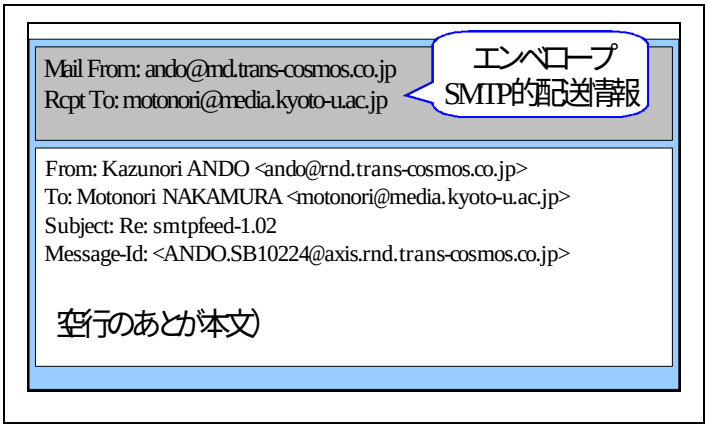

図 6:エンベロープとメール本体

6.1 エンベロープ

実際にメールを配送する時には、メール本文とは別に、そのメールをどこに届けるか、誰が出したかなどの配送用情報が裏で密かに働いており、これがエンベロープと呼ばれています。

エンベロープには、SMTP(Simple Mail Transfer Protocol：メール配送に使われている手順)的な配送情報が含まれています。メール本文に書かれている From と To は、配送とは全く関係がありません。ですから、メール本文の To:に書いてある名前とは全然別の人に届くということが有り得ます。その典型的な例としてメーリングリストがあります。

本文のヘッダに、配送経路(Received:)やリターンパス(Return-Path:)などの配送用情報が記録されている場合もあります。

読者の中には、本文の To に別の名前を書くとそこに送られた経験がある方も居るかもしれません。しかし、これはメール配送プログラムの仕業ではなく、メールを読み書きする MUA(Mail User Agent:クライアント側ソフト)の仕業で、基本的には本文の To と配送先は関係が無いのです。MUA 側で、ヘッダに書いた To アドレスを SMTP 的な配送先情報として MTA(Mail Transfer Agent:サーバ側ソフト)に渡しているからです。メーリングリストで自分にメールが届くのは、MUA 側のメーリングリストドライバがメーリングリストをエンベロープに展開して MTA に渡しているからです。

6.2 メール本体

[ヘッダの話]

ヘッダの情報についてまとめておきます。コロン(：)の前の部分を Field-name、後の部分を Field-body と呼びます。

標準として決められているものは次の通りです。

From:	差出人アドレス
Sender：	差出し人が複数ある場合の、実際の送信者
To:	宛先アドレス
Cc:	カーボンコピー(同じメールの送付先)
Reply-To:	返信先アドレス
Message-Id:	５年間固有の ID
Subject:	タイトル
Date:	差出時間
Return-Path:	エラー返信先アドレス
Received:	配送経路

In-Reply-To:　　どのメールに返信したかを示す
References：　　どのメールに返信したかを示す

7　メール配送モデルとメールサーバ

7.1　メール配送モデル

メールの配送モデルを説明します。ファイアウォール(バリアセグメント)があり、MTA が外と内に設置され、さらにローカルの MTA に繋がっています。MUA はローカルのメールサーバが持っているスプール内メールを、POP や IMAP で持ってきて、自分の端末で読むというモデルが主流です。

モデルのそれぞれの位置に対応するサーバが存在しますが、注意点を説明します。

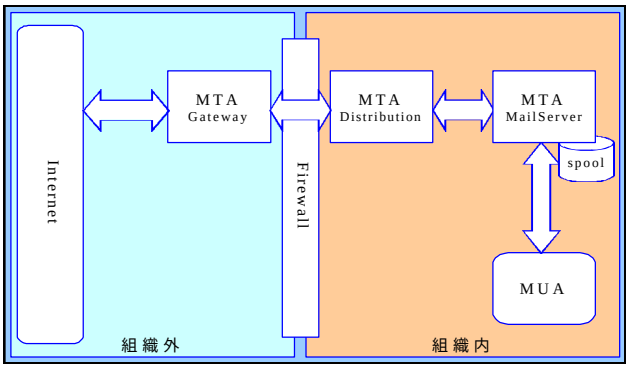

図 7:メール配送モデル

(a) ファイアウォール外側のサーバの注意事項（組織外の MTA）

・ SPAM 対策をきちんと行う
・ 高いセキュリティが必要
・ メールウィルス
このサーバを通過して内に入られるとどんなトラブルが発生するかわかりません。

(b) ファイアウォール内側のサーバの注意事項(組織内の MTA)

このサーバは、多くの場合中継配送を受け持っています。組織が小さければ配送ルールを static に書き込んでも良いのですが、組織が大きくなった場合、外側の DNS とは別に社内専用の DNS を作り、社内で MX をそれぞれ指定

してサブドメインを分け、各サブドメインのメールサーバに対してメールを割り振る、という役割をファイアウォールを越えた最初のメールサーバに受持たせるのが良い方法です。

(c) ローカルサーバの注意事項（組織内ローカル MTA）

皆さんになじみの深いメールサーバは、ローカルのスプールにメールを溜め込んでくれる、メールを受取ってくれるメールサーバです。このメールサーバは、届いているメールを溜め込んで、ユーザからメールのリクエストがあった時に、POP3/IMAP４などのプロトコルで MUA と通信をして、MUA に対してメールを渡す役割をするメールサーバです。MUA はクライアント側のソフトで、Eudra などがあります。

役割的には、この 3 つのサーバになります。

7.2 配送設定の基本要素

MX 配送か static(静的)配送かを戦略的に選択することができます。このドメインはこのサーバという具合に、特定のサーバが決まっている場合は、static 配送にした方が良い効率が得られます。自分のドメインから他の企業へメールを出すような対外配送の場合は、MX 配送しかありません。ただし、組織内部の配送は組織の規模によってどちらか選択することが可能です。組織内部で独自の DNS の定義をしている場合は MX 配送ですし、集中サーバなら static でも可能です。注意することは、組織内部の配送を DNS でやる場合、resolv.conf での参照する DNS を内部の DNS サーバに指定することです。この設定を間違えると内側の配送がうまく動かないという事態が発生します。

● メール配送モデルとネームサーバ

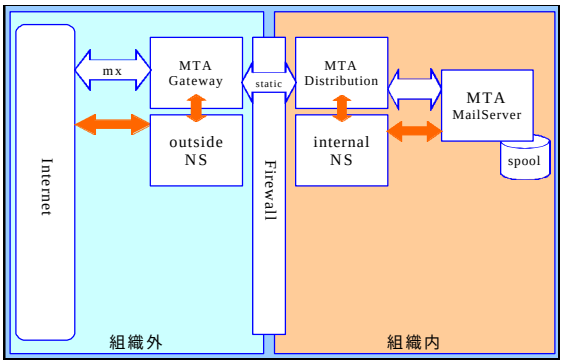

図 8:メール配送モデルとネームサーバ(NS)

図 8 に DNS のネームサーバと配送モデルを併記しました。３種類のサーバ

の位置(役割)によって設定は決まります。外側に関しては外向きの NS とやり取りをしてメール交換します。ファイアウォールを経由して内側の NS とやり取りをする場合は static で経路を指定するしかありません。内側の世界では別のインターナルな NS があり、内部ではこれを見て内部の配送先を指定してもらう設定にするわけです。

① ローカルサーバ(内部 MS)の設定

ローカルなサーバでは、次のように指定します。

```
DIRECT_DELIVER_DOMAINS＝’example.gr.jp’
DEFAULT_RELAY='smtp:gw-in.example.gr.jp’
```

内側は example.gr.jp の MX を見て配送し、外側へは smtp:gw-in.example.gr.jp に渡して static 配送すると指定しています。組織内は MX 配送、外向けは static 配送です。

② 組織内側サーバ(distribution サーバ)の設定

社内配送中継用サーバでは、次のように指定します。

```
DEFAULT_RELAY='smtp:gw-out.example.gr.jp’
```

内側に関しては MX を見て配送を行い、他のドメインに関しては外のサーバに対して送り出す指定になります。組織内は MX 配送、外向けは static で gw-out に渡すわけです。

③ 組織外側サーバの設定

外のサーバは次のように指定します。

```
STATIC_ROUTE_FILE=‘example-static.def’

GW  [202.250.31.150]
DOM  example.gr.jp
```

組織内へは static 配送をし、外向けは MX 配送で行うという設定です。

8　対外受信ホストの多重化

8.1　MXを複数にする理由

MX を複数にするのは、メールが集中して負荷が高い場合に一時的に溜め込んで、空いたら本当のサーバに送り込むという方法が取れるという利点もありますが、バックアップを用意しておけるというのが一番大きな理由です。可能なら、2nd MX は 1st MX とは独立して配信させるようにしてください。メンテナンス用としても、2 つ用意するのは意味があり、片方が停止しても受け取りに支障を出さないメリットがあります。

① 2nd MX のあるメール配送モデル

外側に 2 台の MX サーバを置いている場合の配送モデルです。プライマリの負荷の少ない時に、セカンダリからプライマリにメールを送り負荷分散します。

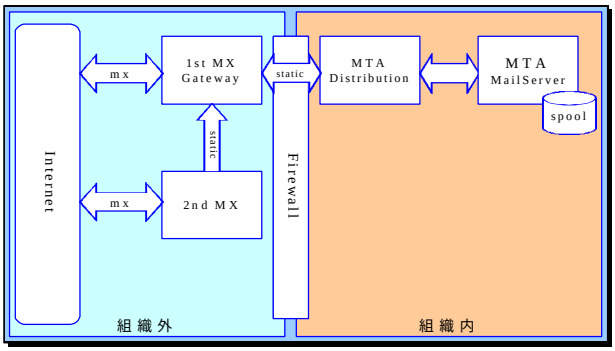

図 9:2ndMX の配送モデル

② FallbackMX

再送専用ホストです。DNS が引けなかった場合や全 MX に対してメールが送れなかった場合に、再送キューを特定のホストに集める指定(しくみ)です。これを指定すると、一時的に配送できなかったメールがそのホストにたまるのでネットワーク的なトラブルがすぐわかります。注意することは、再送を試みる期間の調整(デフォルトでは 5 日間 - sendmail の場合)です。また、大量配信を行う場合に、1stMX が外側に送ろうとして送れなかったメールを、FallbackMX に送って負荷を分散することもできます。

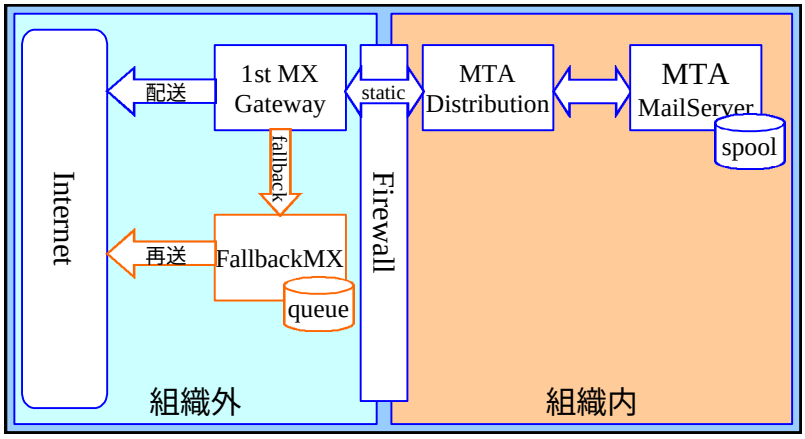

図 10:FallBackMX の配送モデル

9 MTAの種類

9.1 MTAの種類

① Sendmail （商用版：Sendmail Pro）

さまざまな意味で最も手堅い選択であると言えます。世界の MTA の 78%を占めているのでバグの修復が早く行われます。特徴としては、処理限界がファイルシステムのスピードに依存していることです。ファイルベースで処理をしているので、途中で電源が落ちても、メールを失わないような対策をきちんとしています。大規模メーリングリストの配送は苦手ですが、smtpfeed を併用することもできます。sendmail.cf の設定が難解ですが、GUI で設定ができる商用版 Sendmail Pro もあります。

② qmail

単機能の小さいプログラムの集合体として作られているので、必要なものだけが呼び出され、軽くて比較的パフォーマンスが良いのが特徴です。ただし、個々の機能を見ると必ずしもベストではありません。たとえば、同一 MX 宛てのメールをまとめ送りをする機能がなく、配送戦略は「力任せ」と言うことができます。弱点は、ログ情報の不足、エラーメッセージの不備があり、トラブルが発生しても原因が容易にわからないことです。

③ その他の商用製品

・Netscape Messaging Server

・InterMail
・SIMS（Sun Internet Mail Server）
これらすべてがオールインワンサーバ（MTA＋仲介サーバ）です。ユーザ数ベースの課金形態をとっています。これは商用データベースを併用しており、データベースがユーザ数による課金をしているからだと思われます。ユーザ数の上限を拡大しているのが特徴です。

9.2 MTAの選択

・MTA 選択の際のポイント

アカウント管理のやり方を、OS に依存しているのは Sendmail で、LDAP を利用しているのは SIMS などの商用製品です。実は Sendmail でも頑張れば LDAP を利用することはできます。

・仲介サーバは POP か IMAP か？

POP はメールをクライアントが保持するので、リスク分散と見ることができます。IMAP は便利ですがサーバがメールを保持しているので、パフォーマンスもリスクもサーバに依存しています。

・サーバの実際の収容数は？

サーバの実際の収容数は定期メールチェックのアクセス頻度に依存しており、ネットワーク接続の同時接続数の上限は OS に依存しています。メールチェックはネットワークとファイル I/O のリソースを消費します。たとえば、100Mbps のリソースも 10,000 人で使えば１人あたり 10kbps 分しか使えません。リソースの上限近くでは急激にパフォーマンスが悪化するので、上限近くまで消費することのないようにしてください。

少数ユーザでテストして大人数の場合を推測したり、ストライピング (RAID0)利用のストレージを使用したシステムでのベンチマーク(これは、製品の性能ではなくストレージの性能への依存が高いから) を示すようなごまかしに注意しましょう。

・セキュリティホール

そもそもセキュリティホールは「あると仮定しておくべきもの」と考えるべきます。セキュリティホールについては、次のことが言えます。

- 一般にシェアの大きい方が発見は早い
- OS 自体のセキュリティホールの方が圧倒的に多い
- 商用製品なら安心かというと必ずしもそうではない
- MTA 自体が完璧なら安心というものでもない

セキュリティホールが心配ならセキュリティホールに関する情報を得る努力をしてください。実際、セキュリティホールは呆れるほどたくさんあります(例:*http://www.securityfocus.com/*)。

10　知っておくべきメールアドレス

MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS（RFC2142)で挙げられているメールアドレスは知っておくべきです。

abuse@example.gr.jp	いざという場合の問い合わせ先
postmaster@example.gr.jp	メール配送についての問い合わせ先
hostmaster@example.gr.jp	DNS についての問い合わせ先

メーリングリスト周辺アドレスの例も知っておきましょう。

owner-hoe@example.gr.jp	メーリングリストの発信者アドレス
hoe-admin@example.gr.jp	管理者の別名として使われる
hoe-request@example.gr.jp	RFC2142 的管理者アドレス
hoe-errorsto@example.gr.jp	エラーメールの専用受信アドレスを用意している場合

11　メールの抱える問題

11.1　SPAM

SPAM は、一昨年から話題になっていましたが、受け取りたくないのに勝手に送り付けられてくるメールです。

① SPAM 中継の被害の構図

悪意を持った SPAMMER(SPAM 発信者)がいて、踏み台にされたホスト（オープンリレー）を経由して SPAM メールを送ってきます。踏み台にされたホストとは、管理上の手落ちによって発信者を問わずにメールをどこにでも送るような設定になっているサーバのことです。

SPAM は、中継ホストや偽装された発信者にも被害を与えます。

・ 中継ホスト：SPAM 対策をやっているにもかかわらず疑われるという

被害が発生します。

・ 偽装された発信者：SPAMMER は、発信者アドレスを偽装している場合が多く、そのために覚えのないエラーメールが大量に戻ってきます。

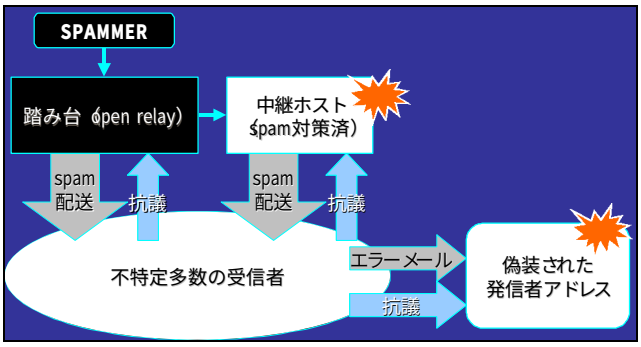

図 11:SPAM 被害

②SPAM 対策

SPAM 対策はいろいろ考えられますが、代表的なものを説明します。

(a) RBL（Realtime Blackhole List）を利用する方法

・ RBL とは、SPAM の発信源を登録する闇魔帳です。DNS と同じ枠組み(named)で作られており、MTA がメール送信元の IP アドレスを RBL のサーバに照会し、RBL と一致したら MTA はメールの配送を拒否するというしくみのものです。
・ また、類似のものが何種類もありますので、それを利用することもできます（ APS RBL、MAPS RSS、MAPS DUL、ORBS、IMRSS など）。
・ 自分のサーバが RBL に登録された場合には、自分が出したメールが相手に届かないというトラブルが発生するので、注意しましょう。
・ RBL のサーバが機能を停止すると「すべての IP アドレスはクロ」という判定になってしまい、問い合わせたサーバが受け取りを拒否する可能性があり、全然メールが来なくなるということがあるので注意が必要です。

(b) SPAMLIST を利用する方法

SPAM LIST では、到着したメールの発信元について、メールアドレス（envelope from)、ドメイン、IP アドレス、のいずれかを指定して SPAM を排除できます。3 種類指定できるのは便利ですが、メンテナンスが大変です。

(c) POP before SMTP を利用する方法

ISP で利用されている手法で、MUA から MTA を POP アクセスすると、ある一定期間内（例：１時間）はその IP アドレスからの SMTP 接続を許可する方法です。例として qpopper(POP アクセスを受付けるデーモン）にパッ

チを当てて実現する方法があります。POP before SMTP は安全性が低いため、最近でてきた SMTP Authentication に置き換えられる方向にあります。

(d) SMTP Authentication（RFC2554）

POP before SMTP の置き換えを意図した SPAM 対策の技術です。 MUA での対応が条件になります。

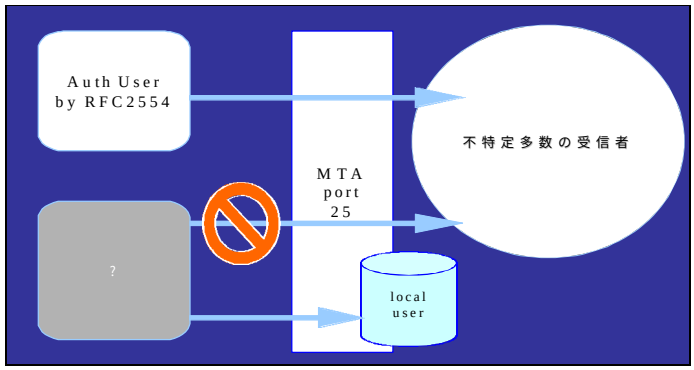

図 12:SMTP Authentication

SASL(RFC2222)を利用した Relay 認証で、sendmail-8.10 では cyrus SASL ライブラリを利用しており、デフォルトでは/etc/passwd を利用した認証を行います。認証を通るとそのサーバ経由の Relay 配送を許可します。これによって不正中継を防止できるので SPAM の問題がなくなります。

(e) Message Submission（RFC2476）

SMTP Authentication とは別にメッセージサブミッションという新たな枠組みも出てきました。

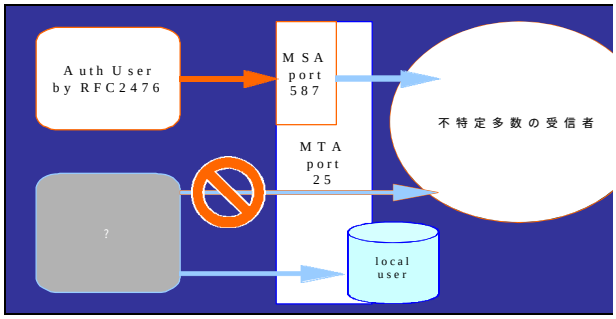

図 13:Message Submission

MSA（Message Submission Agent）を新たに定義することによって、メールを「出す」新たな枠組みを作り、SPAM を防止しようというものです。外のユーザからメールを出す時に SMTP で渡すのではなくて、他のポートに対

してメールを投げ、そのポート（port 587）で認証も行う方法です。
Sendmail-8.10 は MSA になる機能が組み込まれています。

メッセージサブミッションで認証を受けたメールは、MSAport587 で一旦受け取られます。それが不特定多数の受信者へ配送可能なメールです。このポートを利用しない普通の継配送に関しては受け取りを拒否します。SMTP はローカルなユーザだけ受け取ります。こうすることによって、SPAM の不正中継を防ぎます。

11.2 メール経由のウイルス

メール経由のウイルスは多くの場合添付ファイルが感染源です。

① マクロウイルス（Excel、Word、PowerPoint）

マクロウィルスでは、中に忍ばせてある Office オブジェクトが曲者で、代表的なものには、勝手にウイルスメールをばら撒く Melissa があります。大規模メーリングリストでは「添付ファイルは許可しない」という規制も考えられます。

② 実行形式ファイル

添付された実行ファイルを不用意に実行してはいけません。代表として、Happy99 があります。

11.3 その他の問題

● チェインメール

① 善意の協力依頼を装うチェインメール（あるいは本物）

「このメールを転載してください」が曲者です。本物の場合は期間や範囲を限定して一定数しか転載されない工夫をしているのが普通なので、無制限の転載を意図している場合には無視するのが正しい対処です。

② 不幸／幸福のメール

「このメールを５人に転送しないと...」というメールです。初心者の多い環境で流行りやすいので注意が必要です。

● メール爆撃（Bombimg）

メール爆弾には 2 種類あります。
・ 巨大なサイズのメールを送付

・ 膨大な数のメールを送付
どちらもサーバのスプールを膨らませる結果になり、ディスクがあふれてメールを受け取れなくなります。メーリングリストではさらに深刻な問題になります。サイズ制限、通数制限などの防御を講じることで防止が可能です（ サイズ制限例　O MaxMessageSize=500000 ）。

● アドレス詐称／隠蔽問題

SPAM、メール爆弾などでは、多くの場合、発信者アドレスが偽装されます。SPAM 発信者はエラーメールにより偽装された発信者を bombing したりします。メーリングリストに他人のアドレスを登録する場合もあります。メーリングリストでのメンバー登録が自動登録で確認なしの運用の場合には、いたずらされることもあるので必ず確認するようにします。

● まとめと展望

SPAM 対策が、「来たときの対策」から「出させない対策」へ移行してきています。対策として、SMTP Authentication（RFC2554)、Message Submission（RFC2476)、SMTP over TLS(Transfer Layer Security:SSL のようなもの（RFC2487）)が出てきています。

メーリングリストではアドレス一覧を出さない対策が必要で、たとえば、私が作った PPML は一般参加者の who コマンドに対して GECOS の一覧を出すだけで、メールアドレスを出さないようにしています。