※ http://www.atmarkit.co.jp/fnetwork/rensai/dns01/dns01.html からの引用（一部改変）

集中連載：DNS の仕組みと運用（1）

DNS の仕組みの基本を理解しよう

■DNS はなぜ必要か？

まず最初に，「DNS とは何か」を説明するために，「なぜ DNS が必要になるのか」を考えてみよう。それには，歴史的経緯から考えるのが分かりやすい。

DNS はご承知のとおり，IP アドレスとホスト名をマッピングして相互解決するための仕組みだ（これを「**名前解決**」と呼ぶ）。IP アドレスは単に数字でしかないので，さまざまなホストを管理したり，アプリケーションから特定のホストを指定するには，人間にとって理解しやすい「名前」の方が便利だ。

DNS と同じように，IP アドレスとホスト名のマッピングを行う最も原始的な仕組みが「**hosts**」**ファイル**だ。各ホストがローカルファイルとしてマッピング情報を管理している。Linux では「/etc/hosts」，Windows 2000 では「*Windows ディレクトリ*¥system32¥drivers¥etc」の中にある。

```
127.0.0.1       localhost
192.168.1.10    host1
192.168.1.1     gw
10.1.210.10     www
```

リスト 1　hosts ファイルの例

hosts ファイルは，インターネットの前身である ARPANET で生まれた一番最初の「名前解決」サービスだ。現在でも，小規模なネットワークでは有効な方法だ。非常に単純な対照表で運用も設定も楽だが，登録されるホスト名が増えれば逆に非効率的な仕組みであることも，直感的に理解してもらえるだろう。 hosts ファイルは，各ホストごとに設定しなくてはならないし，そもそも各ホストで同じ名前で設定できているという保証もない。

ARPANET においても，ネットワークに接続されたホストが数百を超えたあたりから運用の限界に達した。そこで考案されたのが DNS であり，1984 年より実際に運用が始まった。hosts ファイルのような対照表を管理する専用サーバ＝**DNS サーバ（ネーム・サーバ）**に情報を集約して，名前解決が必要なクライアントやアプリケーションから問い合わせをさせよう，という発想である。これにより，データ管理は DNS サーバだけで行えばよいことになる。だが，単に hosts ファイルをそのままサーバへ移行するだけでは問題も残る。

例えば，たった 1 つの DNS サーバだけで全世界に存在する数百万以上のホスト情報を処理するのでは負荷に耐えられないだろうから，複数サーバによる運用は 当然必要となる。だが，それらで完全に同じデータを持ち合うのでは，非常に非効率的だ。インターネットに接続する組織で，（当時は考えもしなかっただろう が）全世界に存在する数百万以上のホスト情報を保持しないといけないとしたら，これはかなりの負担だ。

また，「ホス ト名のユニーク性」という問題もある。もしホスト名を付ける場合に，数百万ものホストが登録されたデータに対して，手動でユニーク性を確認しなければならないとしたらどうだろう。最も効率的なのは，ある種のホスト命名規則を用いて，ユニーク性を保証することだ。

これらを解決するのが，DNS の根本ともいえる「**ドメイン・ツリー**」の概念である。

■「ドメイン・ツリー」と「分散環境」

普段使用される「**DNS 名（ドメイン名）**」などと呼ばれるホスト名は，「.（ピリオド）」でいくつかの階層に区切られる。それぞれの階層ごとに，それ以下に含まれる下位ドメイン名やホスト名を管理するという「分散型」サービスとなっている。

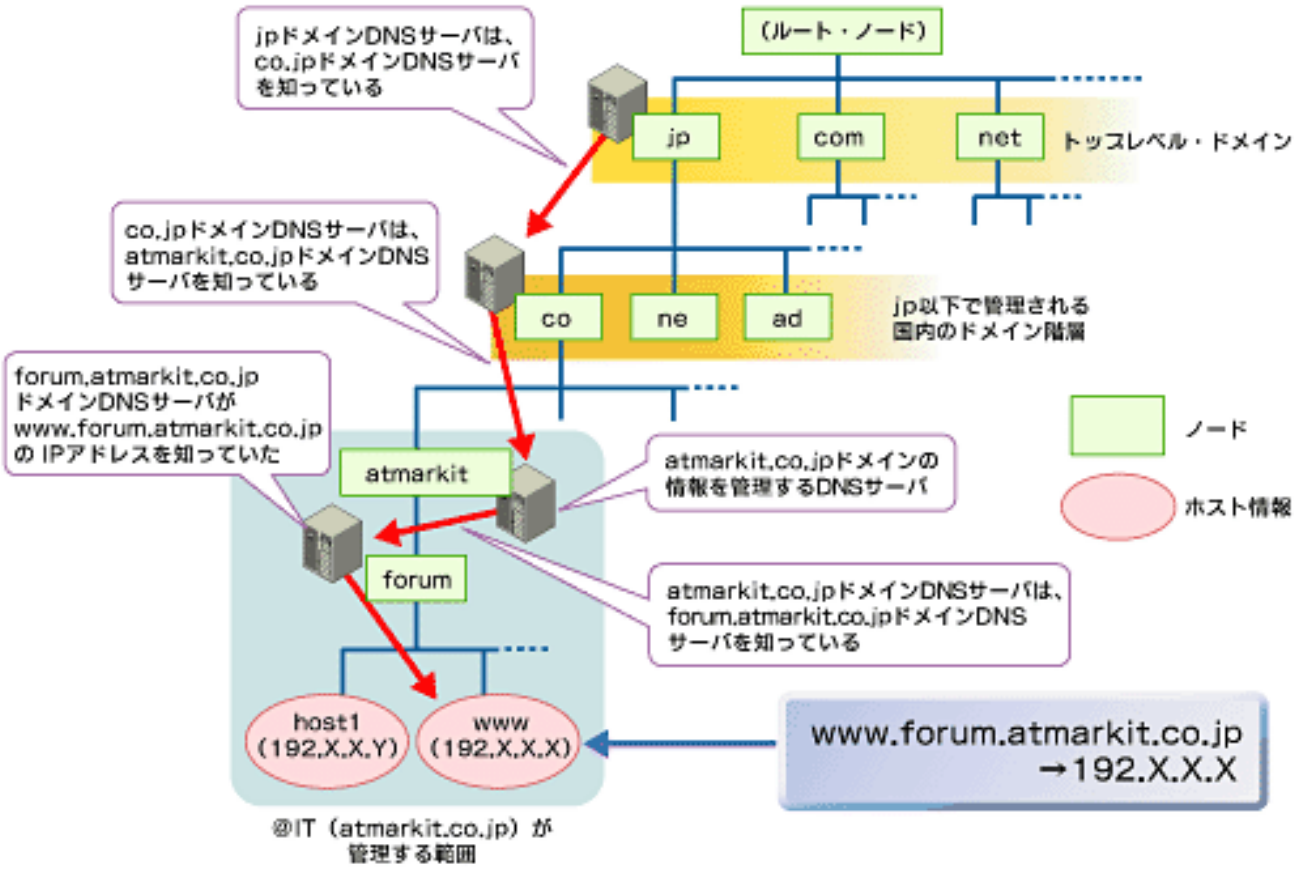

図 1 ドメイン・ツリーの概念

このツリーの「枝分かれ」部分に該当するのが，いわゆる「**ドメイン**」である。「atmarkit.co.jp」など，上位ドメイン名を付けたものと区別するために「**ノード**」などとも呼ばれる。また，このようなデータ構造を含む名前を「**名前空間（ネーム・スペース：Name Space）**」などとも呼ぶ。

各ノードでは，

- 自身の下位ドメイン（サブドメイン）として何があるのか。そのサブドメインの情報を持っている DNS サーバは何か
- 自身に所属するホストの情報（IP アドレスなど）

などの情報が管理される。これらを管理しているのが，それぞれのノードに位置して，そのドメインを管理する DNS サーバである。

ドメインのレベルによっては，主にサブドメイン情報のみを管理するサーバもあるかもしれないし，ホストしか管理していないサーバもあるだろう。だが，それぞれの DNS サーバの挙動や役割は，大きくは変わらない。

基本的に，各ノードに位置する DNS サーバでは，自身の管理するドメイン内情報とサブドメインの DNS サーバ名しか知らない。そこで，DNS への問い合わせ側（DNS クライアントのことだが「リゾルバ」などとも呼ばれている）は，この階層を上位から順にたどっていけばよい。ルートから jp ドメインの DNS サーバへ，jp ドメインの DNS サーバは co ドメインの DNS サーバを知っているので，次に co ドメインの DNS サーバへ atmarkit.co.jp ドメインの DNS サーバを尋ねる。そして最終的には，「www.forum.atmarkit.co.jp」の IP アドレスを知る DNS サーバ（ここでは forum.atmarkit.co.jp ドメインの DNS サーバ）までたどり着けるだろう。

一見，非常に煩雑に見えるかもしれない。だがこのようにデータが配置できれば，各ノードで分散して情報を保持できるし，あるノードにおいて「host1」というホストを登録しても，ノード内のユニーク性さえ保証すれば，ドメイン名をホスト名に付加することで[*1]，自動的にユニーク性が保証される。実は，非常に効率的かつ柔軟なシステムなのである。すなわち DNS とは，全世界に張り巡らされた「**分散協調型データベース・システム**」なのだ。

[*1] このようにホスト名にドメイン名を付加したのが **FQDN（Fully Qualified Domain Name）**だ。例えば，「www.atmarkit.co.jp」が FQDN で，「www」を単にホスト名と呼んで区別することがある。なお，厳密には FQDN は一番右側に「.（ピリオド）」を付加して示す。この例では「www.atmarkit.co.jp.」となる。この最後にあるピリオドは，ここが**ルート・ノード**であることを示している。すなわち「**絶対パス形式**」名というわけだ。ピリオドがない場合には「相対パス形式」と考えられる。ただし，Web ブラウザで FQDN を指定する場合などは，相対パスではあるものの，そのスタート・ポイントのデフォルト値はルート・ノードとされるため省略可能なのだ。ピリオド付きの FQDN を指定するのは，現実には DNS サーバ設定時ぐらいしかない。

また，こうした運用を行う上で，上位ドメイン（ノード）はサブドメイン（ノード）の DNS サーバ情報のみを保持しており，そのサブドメイン以下の情報についてはまったくタッチしていない。つまり，サブドメインを完全に信頼して，サブドメイン以下の定義（さらにどのようなサブドメインが含まれるか，どのようなホストが含まれているかなど）は，そのサブドメインに任せてしまうことになる。実は，1 台の DNS サーバで自身のドメインと，そのドメインのサブドメ イン情報をも管理してしまうことは可能なのだが，この場合，サブドメインの情報の管理は，別の DNS サーバに任せてしまうわけだ。

なぜか？ 例えば，一度組織にドメインが割り当てられてしまえば，そのドメインでどのようなサブドメインを作ろうと，ホスト名を登録しようと，完全に自由だ。ドメイ ン以下のことはそれぞれの組織に任せてしまった方が，管理効率は向上するからだ。これを，上位ドメインからサブドメインに対する「**権限委譲**（delegation）」と呼ぶ。権限委譲とは，自身から別の DNS サーバへその部分の管理権限を委譲する，という意味である。すなわち，ドメイン・ツリーは世界中の無数の DNS サーバをつなぐ「信頼の連鎖」ツリーでもあるのだ。

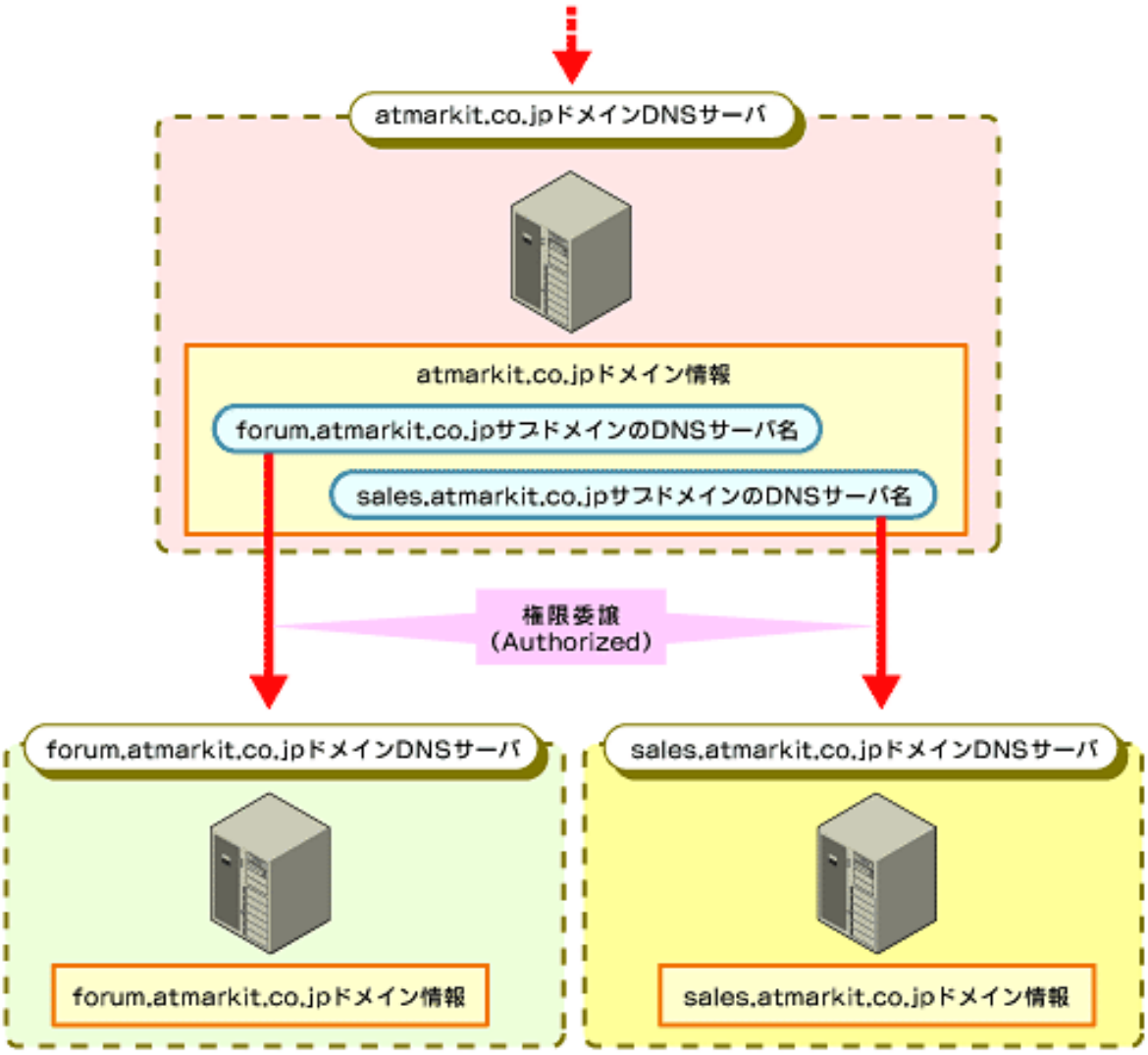

図 2 権限委譲とドメインの関係

皆さんの組織のドメインも，必ずどこかの上位ドメインから権限委譲されている。例えば，atmarkit.co.jp ドメインは，co.jp ドメイン（を 管理する DNS サーバ）から管理権限を委譲されている。詳しくは本連載の第 2 回以降で説明するが，各組織の DNS サーバをインターネット上に立ててドメイ ン情報を登録し，上位ドメインでそのドメインへ権限委譲していることを登録して初めて，インターネットのコミュニティから皆さんの組織のドメインとホスト が，DNS 上の一員であることが確認できるのである。

■「ゾーン」と「レコード」

では，各ノードの DNS サーバではどのような情報が必要になるのだろうか。各ノードで管理される DNS 情報の単位を「**ゾーン**」と呼ぶ。ゾーンは，DNS 管理から見た場合のデータ範囲だ。1 台の DNS サーバで複数のドメイン（親ドメインとサブドメインなど）を含む単一のゾーン情報 を管理している場合もあれば，1 台の DNS サーバで複数のゾーンを管理している場合もある。つまり，1 つのゾーンは必ずしもドメインや物理的な DNS サー バの単位と一致するわけではないのだが，少しややこしいので，ここでは「“1 ゾーン＝1 ドメイン”を 1 台の DNS サーバで管理している」と考えてもらっていい。

ゾーンに含まれるデータの単位が「**レコード**」だ。まさしく表データの中の 1 つ 1 つの行だと思っていい。ただし，データの種類は多岐にわたり，行というイメージからは程遠いかもしれない。実は，ゾーンに登録されるレコードは，ホスト名と IP アドレスの情報だけではなく，表 1 に示すような情報が登録されている。に以下に，主なレコードの種類を紹介する。

まず「**SOA（Start Of a zone Of Authority）**」は，ドメイン定義を宣言するレコードだ。ドメインの DNS サーバ名のほか，主にゾーン情報としてほかの DNS サーバへ転送する際の管理情報などから構成さ れている。ゾーン転送については連載の第 3 回にて解説する予定なので，いまはドメインのヘッダ情報のようなものだと考えていてほしい。

NS レコードは，ドメインとそのドメインの DNS サーバを指定するレコードだ。このレコードによって，サブドメインの存在とその DNS サーバへの権限委譲が確認できる。

次に，**A レコード**がホスト名から IP アドレスを得るためのレコードだ。例えば，Web ブラウザがユーザーに URL を入力されて，そのホスト名から IP アドレスを取得する場合は，DNS サーバに対してこの A レコード情報を返答するよう命令しているのである。

A レコードなど，ホスト名をキーにして IP アドレスといった付随情報を取得する DNS 解決方法を「**正引き**」と呼ぶ。逆に，IP アドレスをキーにしてホスト名を得る方法を「**逆引き**」と呼んでいる。逆引きのためのレコードが「**PTR**」レコードだ。それぞれ背中合わせの同じような処理に見えるかもしれないが，DNS においてはまったく別のレコードに依存している，別々の処理である。

レコード種別	意味
SOA	ゾーン（ドメイン）情報を記載する。以下のようなデータを保持する **・ドメインの DNS サーバ名** **・ドメイン管理者のメール・アドレス** **・シリアル番号**―ゾーン転送時に情報が更新されているかどうか判断に用いられる（本連載の第 3 回参照）。数値が大きくなっていれば更新済みという意味だ。番号は任意だが，管理しやすいように通常は「年月日＋連番」などの書式が多く用いられている **・更新間隔（refresh）**―このゾーン情報のゾーン転送間隔時間を秒で指定する **・転送再試行時間（retry）**―ゾーン転送に失敗した場合の再試行までの猶予時間を秒で指定する **・レコード有効時間（expire）**―ゾーン情報を最新と確認できない場合の有効時間を秒で指定する **・キャッシュ有効時間（TTL）**―このゾーン情報をキャッシュする場合の有効時間を秒で指定する
NS	ドメインの DNS サーバ名を指定する
A	ホストの IP アドレス
PTR	IP アドレスに対するホスト名
CNAME	ホスト名のエイリアス（別名）
MX	ドメインのメール・サーバ名
HINFO	ホストの追加情報。ホストのハードウェア・ソフトウェア（OS）情報を記述する
WKS	ホストで実行されているサービス情報（Well Known Services）
TXT	ホストへのテキスト情報

表 1　主な DNS レコードの種類

逆引きは何のために使われるのかと思われるかもしれないが，確かにそれほど必須というべきレコードではない。使用例を挙げると，Web サーバのログに記載 されている IP アドレスをホスト名へ変換する場合などだ。しかし，そうした利用を想定しないのであれば，PTR レコードをいちいち設定するのは手間なの で，クライアント・マシンなどでは登録が省かれることも多い。

そのほか，**MX レコード**はそのドメインのメール・サーバを示す。例えば，

president_fujimura@atmarkit.co.jp

という右辺がドメインになっているメール・アドレスに対して，転送元メール・サーバ（sendmail などの MTA）は通常，atmarkit.co.jp ドメインの DNS サーバから MX レコードを検索する。

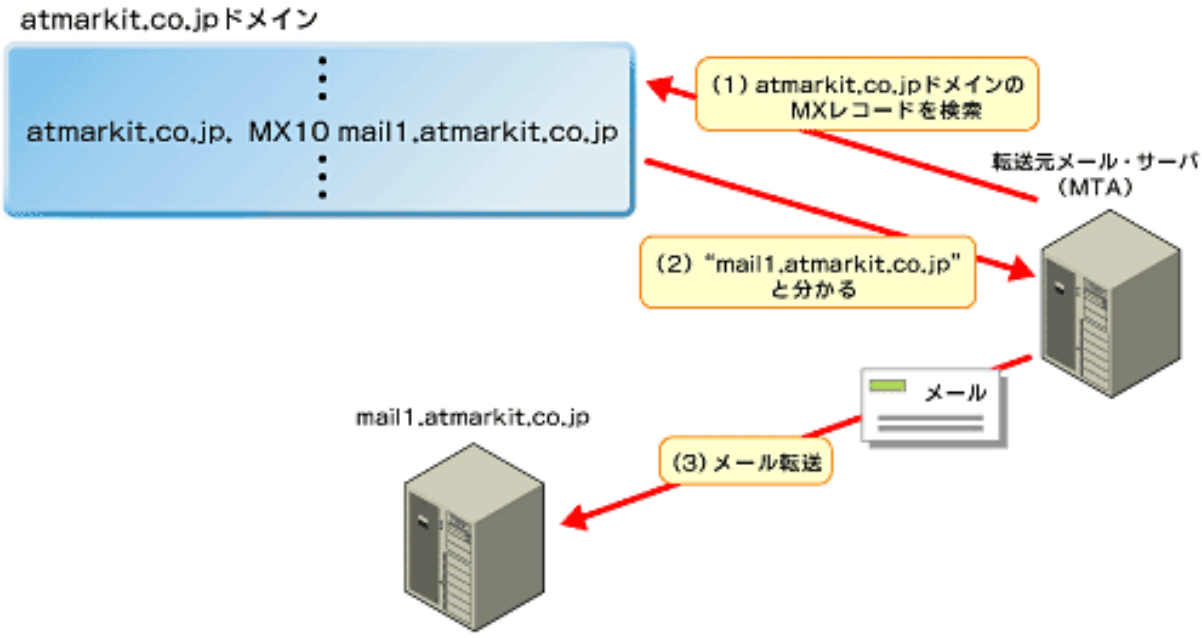

図 3　MX レコードを用いたメール転送

そして MX レコードから atmarkit.co.jp ドメインの受信用メール・サーバ名を取得し，そのメール・サーバに対してメールを転送する。本来，メール・アドレスの右辺はメール・サーバ名そのものだったのだが，このように DNS との連携によってドメイン名を用いることもできる。

■DNS 検索の仕組み

では実際に，どのように DNS 解決が行われているのかを見てみよう。すでに述べたように，DNS 解決はドメイン・ツリーに沿って行われる。例えば，「www.atmarkit.co.jp」というホ

スト名から IP アドレスの解決を行うには，最上位の jp ドメインの DNS サーバに接続して，次に co.jp ドメインの DNS サーバ名を知る必要がある。

だが，ちょっと待ってほしい。すでに問題が発生している。そもそも jp ドメインの DNS サーバ名（またはその IP アドレス）をどうやって知ればいいのだろうか。実をいえば，上の文章には多少ウソがある。jp ドメインはドメインとしては最上位だが，DNS サーバのツリー構造においては，最上位では“ない”のだ。

もう一度，**図 1** を見てほしい。ここの一番上に「**ルート・ノード**」というのがあることが分かるだろうか。つまり，**トップ・レベル・ドメイン**（TLD：Top Level Domain）の上位にルートと呼ばれる特別なノードが存在しているのだ。このノードの正体は「ルート・ネーム・サーバ」と呼ばれる DNS サーバだ。　ルート・ネーム・サーバは，全世界の DNS 環境の最も頂点に位置する DNS サーバである。「.com」「.net」「.jp」など，最上位のトップ・レベ　ル・ドメインの DNS サーバ情報を保持する DNS サーバだ。つまり，DNS 問い合わせの「スタート・ポイント」であり，ここからすべての検索は開始される。このルート・ネーム・サーバを知らない限り，ドメイン・ツリーからの検索は不可能だ。逆にいえば，すべての**リゾルバ**は，このルート・ネーム・サーバの情報さえ最初に与えられれば，確実に必要とするレコードまでたどり着けることだろう。

図 4 に示すように，DNS 検索は，原則的にはこのルート・ネーム・サーバからまずスタートして，順に下位 DNS サーバへの問い合わせを繰り返して，最終的に目的レコードまでたどり着くことになる。

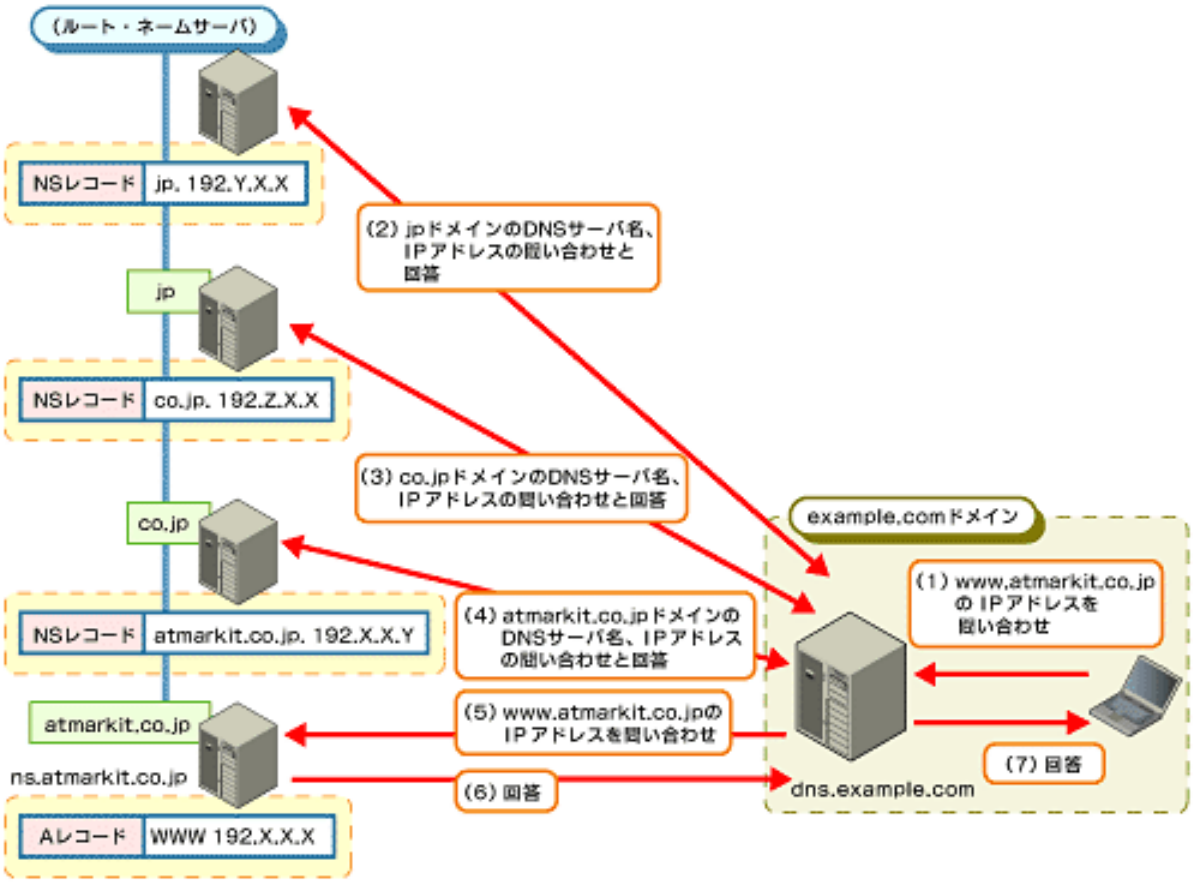

図 4　DNS 検索の仕組み

■「再帰検索」と「リゾルバ」

図 4 で少し注意が必要なのが，図の右側に見える DNS サーバの存在だ。これまで例として挙げてきた「atmarkit.co.jp」ドメインではないうえに，これまで説明してきた「自身のドメイン情報を提供する」DNS サーバとは位置付けが違うのを分かってもらえるだろうか。実は，この DNS サーバはリゾルバ（DNS クライアント）として動作している

例えば，「ns.atmarkit.co.jp」や上位の DNS サーバ は，外部からの問い合わせに対応したり，ドメイン情報を管理したりする DNS サーバだ。だが，普段解決する情報は自身が管理しているゾーン情報だけで，ほかのド メインに関する問い合わせに対しては「知らない」とただ返答するだけだ。

これに対して「dns.example.com」は，ユーザーからの外部ドメインなどの情報問い合わせ要求を受け付けて，先ほど説明したようにルート・ネーム・サーバやそのほかの DNS サーバへ順に問い合わせを行い，DNS 解決を最後まで行おうとする。

この違いは，実は問い合わせ方法にある。「dns.example.com」は，この例では「**リカーシブ**（Recursive：「再帰検索」と呼ばれる）」というタイプの問い合わせをユーザーから受け付けている。この場合，DNS サーバは DNS 解決が完結するまでドメイン・ツリーをたどり，（他のサーバに対して）検索を行い，最終結果を返さなければならない。

一方，「ns.atmarkit.co.jp」が「dns.example.com」から受けたのは，「**イテレイティブ**（Iterative：「反復検索」と呼ばれる）」問い合わせだ。この場合，自身が管理しているゾーン情報にのみ返答し，ほかのサーバへ問い合わせを行うことはない。

多少ややこしいのだが，DNS サーバはこのように問い合わせ種類によって，まったく異なる動作を行う。ここでは，別サーバとして表現したが，組織によっては同一サーバで兼ねていることもある。

「dns.example.com」のように，再帰検索によって完全に DNS 解決を行えるリゾルバを「**フルサービス・リゾルバ（Full-Service Resolver）**」，「dns.example.com」へ要求を送るだけの DNS クライアント（多くの場合は，皆さんが使用しているクライアント PC やメール・サーバなど）を「**スタブ・リゾルバ（Stub Resolver）**」と呼んで区別している。また，スタブ・リゾルバ機能だけを実装している DNS サーバ（俗に「**スレーブ・サーバ（Slave Server）**」と呼ぶ）もある。DNS 問い合わせを受け付けるが，自身では再帰検索を行わないで，別のフルサービス・リゾルバ DNS サーバ（「**フォワーダ**」と呼ばれる）へそのまま検索を転送して，依頼をするサーバだ。社内の DNS サーバを階層的・多段的に配置したい場合などに使用されることがある。

このように，普段クライアント PC へ設定している「DNS サーバ」の項目は，多くの場合，実はマスタ・サーバではなく，フルサービス・リゾルバ用 DNS サーバなのである。要するに，DNS サーバにはさまざまな機能が詰め込まれており，多様な使用方法があるということなのだが，技術書によっては，これらの違いは当然のこととして表されている場合も多い。最初は分かりにくいと思われるので，しっかりと把握して混乱しないようにしよう。

■「逆引き」と「逆引き用ドメイン・ツリー」

これまで説明したドメイン・ツリーによるドメイン検索は，正引き用の検索だ。つまり，SOA レコードや A レコード，MX レコードなどはこの検索経路をたどる。注意が必要なのは，逆引きのときの PTR レコード検索だ。つまり，PTR レコード検索の場合は，最初に与えられるのは IP アドレスなので，当然このようなドメイン・ツリーは検索できない。そこで逆引きのためには特別な「逆引き用ドメイン・ツリー」が用意されている。

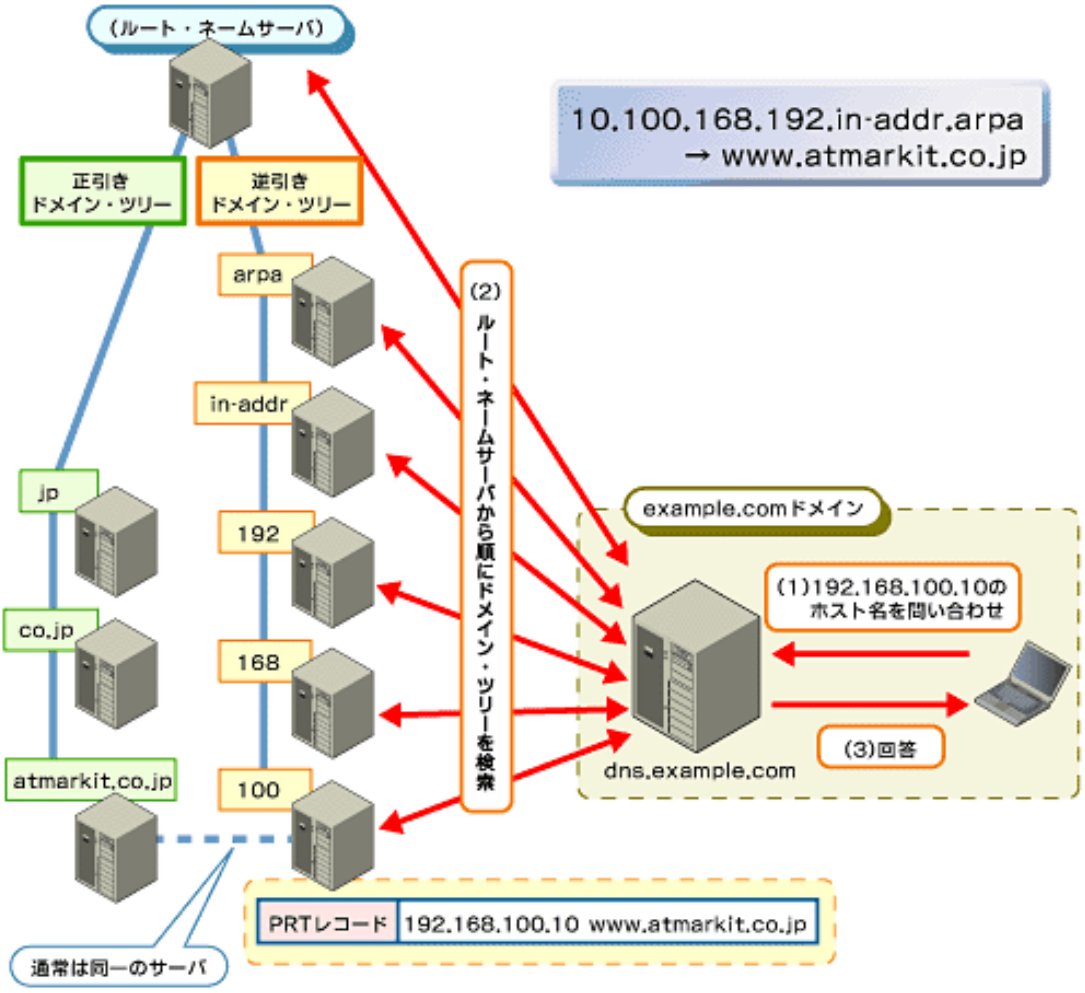

図 5　逆引きレコードの DNS 検索

考え方は完全に正引きと同じだ。ただし，トップ・レベル・ドメインとセカンド・レベル・ドメインは，「.in-addr.arpa」という特別な名前が付けられている。もちろん，それぞれのノードに該当するDNS サーバもちゃんと用意されている。ルート・ネーム・サーバは正引きと共通だ（一部提供しないルート・ネーム・サーバもあるようだ）。

ポイントは，それ以下のドメイン（ノード）部分で，検索の基になる IP アドレスを逆順に並べたものだ。つまり，IP アドレスの各オクテットを仮想的にノー ドと見なして，「第 1 オクテットを担当する DNS サーバ」→「第 2 オクテットを担当する DNS サーバ……」と順に検索を行う。このあたりの動作原理は，正 引きのときと完全に同じだ。違うのは，DNS サーバが保持して管理しているデータの部分だけだ。

問題となるのは，このようなツリーを構築して DNS サーバで管理するためには，同じオクテットに属する IP アドレスをどの組織に割り当てるか，明確にして管理しなければならないという点だ。だが現在では，**CIDR**（Classless Inter-Domain Routing：クラス・レスな経路集積型ルーティング）のために，IANA など，各ネットワーク管理機関が地域別にまとめて割り当てる管理が行われているため，普通は混乱することはない。

もっとも，組織内でプライベート・アドレスを基に DNS 環境（独自のドメイン・ツリー）を構築する場合にも，CIDR を意識した IP アドレスの振り分けを していないと，ルーティングだけでなく，DNS 環境においても影響が出る可能性がある。複数のサブドメインを運用するならば，個々のサブドメイン内で使用 する IP アドレスは，同一サブネットに属するアドレスをまとめて使用した方がいい。

インターネットを支える DNS

ここでは，DNS という巨大な分散システムが，インターネットの中で実際にどのように運用されているのかを見てみよう。

■ドメインの種類

前回説明したように，**ドメイン名**（FQDN）とは「ドメイン・ツリー」と呼ばれる DNS の根幹そのものの書式を示している。

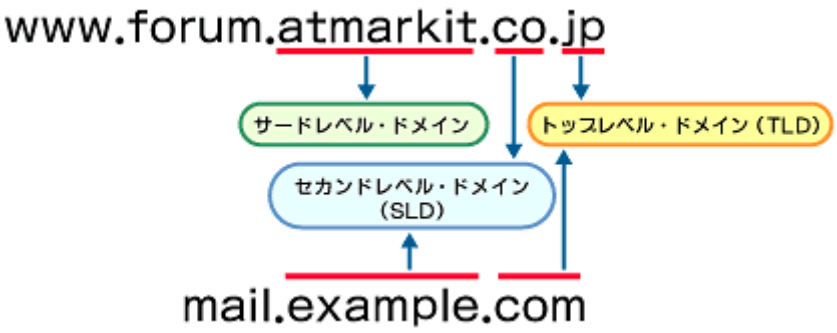

図 1　ドメイン・ツリーを用いた書式

ところで，この**トップ・レベル・ドメイン**（TLD：Top Level Domain）や**セカンド・レベル・ドメイン**（SLD：Second Level Domain）は，だれがどのように決めて管理しているのだろうか？

TLD などを決定・運営しているのが ICANN（Internet Corporation for Assigned Names and Numbers）で，ドメイン名のほか，IP アドレス／ポート番号管理などを指揮する国際的な非営利法人団体である。すなわち，ルート・ネーム・サーバのゾーン情報など，すべての権限は ICANN が管理しているわけだ。

トップ・レベル・ドメインは，大きく分けて 2 種類ある。1 つは gTLD（generic TLD）と呼ばれる汎用目的のためのドメインで，従来からは 7 つ定義されている。もう 1 つが ccTLD（country code TLD）と呼ばれる，国ごとに定められた TLD である。ccTLD のドメイン名は ISO-3166（JIS X 0304 としても定義されている）で定められた国コードから割り当てられている。原則的に，すべての国／地域が所有し自由に使用できるドメイン名である。

TLD名	登録資格要件	管理主体（レジストリ）
gTLD		
.com	商用または企業。ただし現在では制限なし	ICANN（VGRS）
.net	ネットワーク関連団体。ただし現在では制限なし	ICANN（VGRS）
.org	非営利各種団体。ただし現在では制限なし	ICANN（VGRS）
.edu	米国内 4 年制大学（今後は 2 年制大学も含まれる予定）	Educause
.gov	米国政府機関	U.S. General Services Administration
.mil	米軍関係機関	DoD Network Information Center
.int	国際機関	IANA
ccTLD		
.jp .kr など	一般的には各国内に居住／住所地を持つ個人や団体／企業。ただし各国のポリシーによる	一般的には各国ごとのネットワーク管理機関。日本では JPNIC（JPRS）

表 1　TLD の種類

gTLD は，さらに登録資格要件や登録管理主体によって分類できる。「.com」「.net」「.org」は，それぞれ利用目的が定められているものの，実質的に 国籍や目的，団体／個人を問わず自由に取得できる国際ドメインだ。管理責任は最終的に ICANN が負う。そのほかの「.edu」「.mil」「.gov」 などは，目的からして国際的には開放されていない。これらは，当初 gTLD 自体が米国国内向けを想定していたことの名残といえる。「.gov」などは，当 然米国政府そのものが登録を担当することになる。

また，2000 年に ICANN が新たな 7 つの gTLD を承認し，今後運用が開始される予定だ。現在までに，5 つの gTLD がすでに登録受け付けや運用を開始している。

TLD 名	登録資格要件	管理主体（レジストリ）
.biz	商用（ビジネス）向けに特化したドメイン。多様化して本来の意味をなくしている「.com」を補完する目的がある	NeuLevel 社
.info	原則的に利用目的や適格性が無制限なドメイン	Afilias Limited 社
.name	個人の氏名を表すドメイン。例えば taro.tanaka.name など。SLD は考えられる姓を予約しておく予定だそうだ	Global Name Registry 社
.museum	博物館や美術館向けドメイン。chicago.art.museum などのように，SLD は博物館や美術館などの種別となる	Museum Domain Management Association
.coop	消費者連合などの協同組合向けドメイン	NCBA（National Cooperative Business Association）
.aero	航空業界向けドメイン	SITA（Societe Internationale de Telecommunications Aeronautiques SC：国際航空通信共同組織）
.pro	弁護士・医師など専門職向けドメイン。例えば tanaka.med.pro など。専門職であることを証明する必要がある	RegistryPro 社

表 2　新規 gTLD。黄色の背景のものが，運用（テスト含）開始または見込みのドメイン。そのほか（灰色の背景）のドメインも，ICANN とレジストリとの合意が達せられた段階で登録受付開始予定

とはいえ，特にぴんとこないドメインばかりに感じられるかもしれない。実際，より多種の gTLD を求める声は多く，ICANN を中心に現在も gTLD についての議論が続いている。これは，TLD が一種のマーケティング・ツールともとらえられているからだ。

「サイバー・スクワッティング（cyber squatting）」という言葉を聞いたことがあるだろうか？これは，ドメインの取得は原則として先願主義（早い者勝ち）であることを逆手に取って，転

売目的で商標を表すドメイン名の「横取り」をする行為のことだ。 近年，こうした行為が多発しているのは，本来 IP アドレスを分かりやすく表すための「記号」でしかなかったはずのドメイン名を，サイバー・スペース上での 「会社名」や「登録商標」ととらえる傾向が，非常に強くなってきたからだ。企業にとっては，重要なマーケティング要素と見なされるようになったということだ。こうした影響は，gTLD においても例外ではない。もともと DNS というシステムでは想定されていなかった，このような多様化や意味の冗長化現象には 懸念も根強いものの，社会要請という点からは，今後もより多くの gTLD が追加される可能性が高い。

一方，ccTLD では各国の**ネットワーク管理組織**（NIC：Network Information Center）が管理主体となる。日本では JPNIC（(社)日本ネットワークインフォメーションセンター）が担当している。実際に，どのようなポリシーでどのように管理されるかは，国によって異なる。中でも SLD をどう取り扱うかは，完全にそれぞれの国ごとのポリシー次第だ。

例えば，日本では jp ドメイン以下の SLD として「co」「ne」「go」など，組織業態ごとに取得可能な SLD が定められている。「co」は企業組織を 示しているが，ほかの国でも同じとは限らない。au ドメイン（オーストラリア）などでは，「com」で企業組織を示す。

SLD 名	登録資格要件
組織種別 SLD	
ac	4 年制大学など学術組織
co	法人格を持つ企業組織
go	政府機関
ad	JPNIC 会員
ne	ネットワーク関連組織（ISP など）
or	法人格を持つ任意団体や組織
gr	法人化されていない任意団体や組織
ed	高校～幼稚園までの学校組織
地域別 SLD	
.suginami.tokyo.jp など	都道府県など地方公共団体に付帯されたドメイン。住所に即したドメイン階層が表せる。それぞれの都道府県に居住していたり関係する個人や団体向け

表 3　jp ドメインの SLD 種類

DNS 導入に向けての予備知識

多くの読者は，すでに DNS 環境を構築したか，あるいはこれから DNS と付き合わなければならないのかもしれない。それは，勤務先の DNS サーバの場合 もあれば，個人でドメインを取得しようという方もいるだろう。最終回の今回は，これまでの内容を基に，実際に DNS 環境を構築するに当たって必要となるいくつかのヒントについて述べてみよう。

■DNS サーバの配置と種類

DNS 環境を構築する場合に考慮しなくてはいけないのが，実際に DNS サーバをどのように配置するかだ。実は一言で DNS サーバといっても，その役目に応じていくつかの種類に分けられる。自社や個人利用の目的に応じて，配置や種類を決定しなくてはいけない。

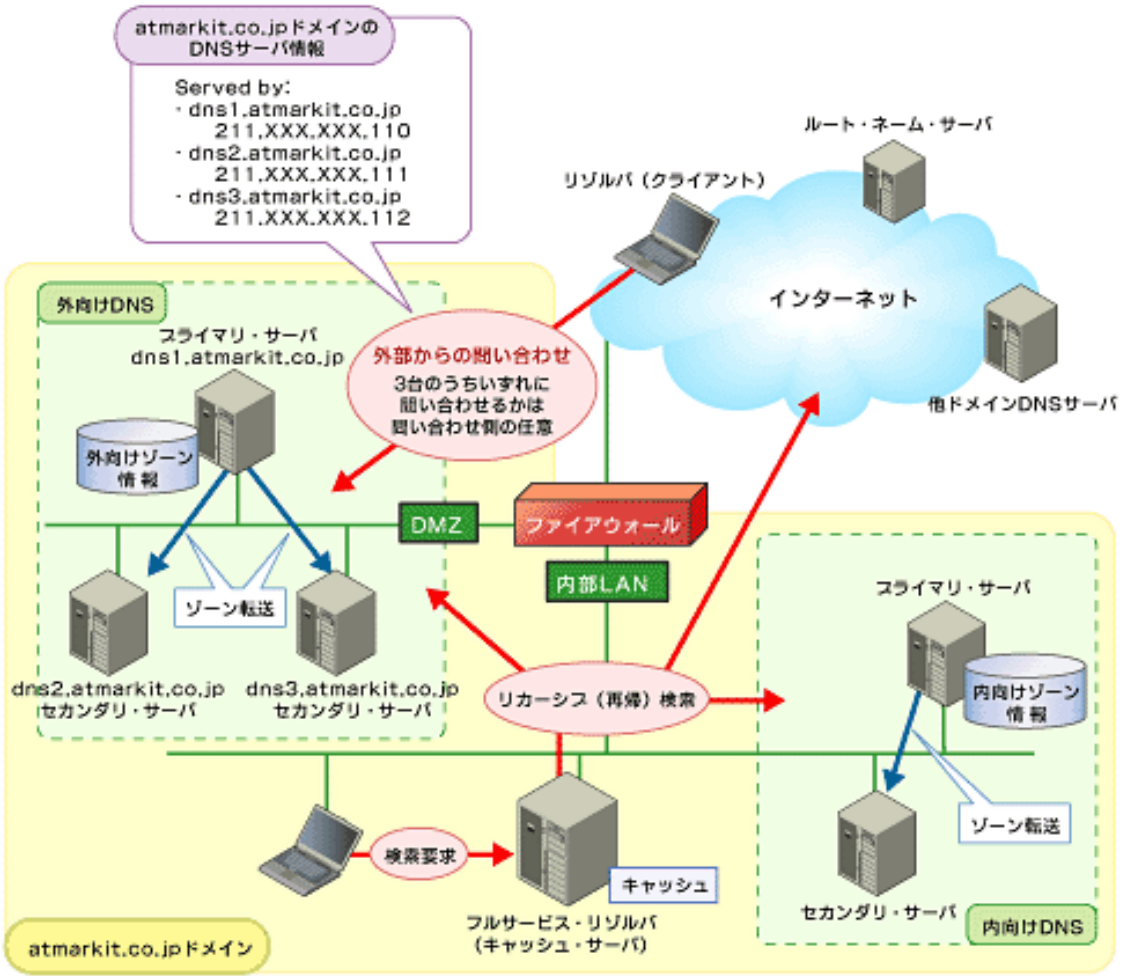

図 1　DNS サーバの種類

●プライマリ・サーバとセカンダリ・サーバ

まず DNS サーバには「**プライマリ・サーバ**」と「**セカンダリ・サーバ**」の 2 種類がある。プライマリ・サーバは自ドメインの全情報を記述した「**ゾーン・ファイル**」を起動時にローカルから読み込み，その情報を基に DNS 問い合わせに応答する。いわゆる「DNS 環境の立ち上げ」とは，まさしくこのプライマリ・サーバを 立ち上げることにほかならない。一方，セカンダリ・サーバとは定期的にプライマリ・サーバへ接続し，プライマリサーバに設定されているゾーン情報を取得し て，同じく DNS 問い合わせに応答するためのサーバだ。プライマリ・サーバからセカンダリ・サーバへのゾーン情報のコピーを「**ゾーン転送**」と呼んでいる。

両者の違いは，ゾーン情報をどこから入手するかだけだ。管理者はプライマリ・サーバ用のゾーン情報ファイルで DNS 情報を設定しておけば，後は自動的に複数のセカンダリ・サーバにも反映されることになり，管理の手間も省ける。一般には，1 台のプライマリ・サーバと，1 台または複数台のセカンダリ・サーバを配置することになるだろう。「**インターネットを支える DNS**」で， ルート・ネームサーバや jp ドメイン DNS サーバが複数設定されていたのを覚えているだろうか。実はこれらがプライマリ・サーバとセカンダリ・サーバ群 だ。複数台の DNS サーバをそのドメインの DNS サーバとすることで，障害時の対応や問い合わせ負荷の分散を行っているのだ。

しかし問い合わせ側からすると，実際にどれがプライマリ・サーバでセカンダリ・サーバなのかは分からない。実はプライマリ・サーバとセカンダリ・サーバの 違いは，ゾーン情報をローカルのゾーン情報ファイルから入手するか，ほかの DNS サーバ（プライマリ・サーバ）から入手しているかの違いでしかない。問い 合わせ側は特に区別はしない（できない）。リゾルバは，通常は複数の DNS サーバからランダムに選択して利用することになる。逆をいえば，実は複数台の DNS サーバがすべてプライマリ・サーバであるという設定も可能だ。見かけ上違いは分からない。ただし，ゾーン情報はそれぞれのサーバで設定することになり，手間が増えるだけなので，あまり意味はないだろう。

■「外向け DNS」と「内向け DNS」とは？

これらプライマリ・サーバ／セカンダリ・サーバの説明は一般に「**外向け DNS**」と呼ばれる利用方法を想定している。パブリック DNS とも呼ばれるが，自ドメイン以外からの問い合わせ応答用 DNS だ（実は内部からの問い合わせも含まれる）。一方，「**内向け DNS**」などと呼ばれる自ドメイン内ユーザーのための DNS も，別途必要になることもあるだろう。これらは，ファイアウォールの内と外にそれぞれ配置されるイメージからきている。

内向け DNS を別に分ける主な目的は，内部用の独自のゾーン情報を設定したい場合だ。イントラネット向けにローカル・ホストは登録するものの，外部には公開したくないなどのニーズを満たせる。つまり，外向け DNS とは異なる独自のゾーン情報（ときには仮想的にルート・ネーム・サーバを構築してドメイン・ツリーも含めて）を保持することになる。また，プライマリ・サーバやセカンダリ・サーバなどを複数台配置してもいい。

■DNS の重要な概念「ゾーン」と「オーソリティ」

ドメインにおけるプライマリ・サーバとセカンダリ・サーバには，DNS 情報の実体ともいえる重要な概念がある。それが「**オーソリティ（Authority）**」だ。

1 つまたは複数のドメインの情報を記載するのがゾーン情報であることは，すでに述べた。ドメイン運用に必要なすべての情報を網羅する。各ドメインは，自身に 関するゾーン情報を責任を持ってインターネット全体に告知する必要がある，ということになる。これを自ドメインに対する「**オーソリティ（権限）**」を持つと表現する。**SOA レコード**はドメインにおけるオーソリティを記述するレコードである。

ただし，より正確にはオーソリティを持っていると見なされるのは，上位ドメインのゾーン情報で，そのドメインの DNS サーバとして指定されたサーバ（NS レコードに記載されたサーバ）に限られる。ドメイン・ツリーに参加していないサーバが，いかにその情報を知っていると主張しても，外部からはそれが正しい情報かどうか確認できないからだ。

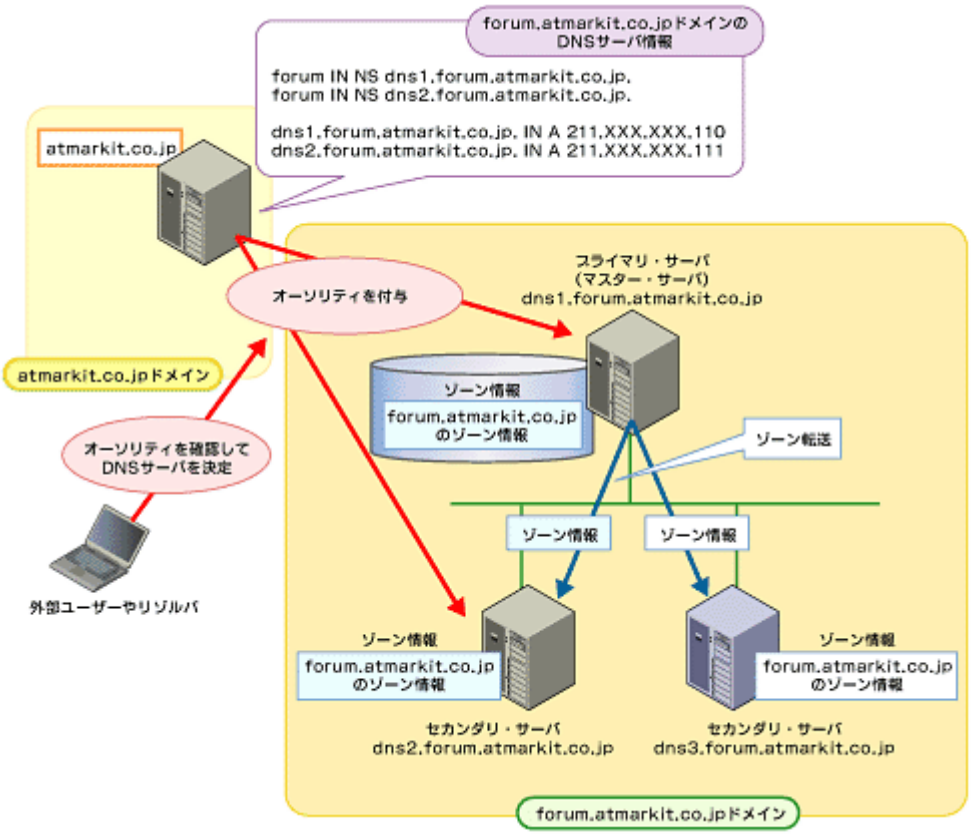

図 2 ゾーン情報を設定してサブドメインを分割する。サブドメインの DNS サーバがそのサブドメインにオーソリティを持つサーバとなる

図 2 では，「dns1.forum.atmarkit.co.jp」と「dns2.forum.atmarkit.co.jp」の管理するゾーン情報はオーソリティを持っているが，「dns3.forum.atmarkit.co.jp」は持っているとは認められない。

いい方を変えよう。forum.atmarkit.co.jp ドメインのゾーン情報は，最初はすべて「dns1.forum.atmarkit.co.jp」によって生成される。また，atmarkit.co.jp ドメインのゾーン情報によれば，「dns1.forum.atmarkit.co.jp」は DNS サーバとして設定されているので，「dns1.forum.atmarkit.co.jp」は forum.atmarkit.co.jp ドメインでオーソリティを持つサーバである。また「dns2.forum.atmarkit.co.jp」は，「dns1.forum.atmarkit.co.jp」からゾーン情報を取得し， atmarkit.co.jpドメインでDNS サーバとして設定されているので，やはりオーソリティを持っている。

しかし「dns3.forum.atmarkit.co.jp」は，同じく「dns1.forum.atmarkit.co.jp」からまったく同じゾーン情報を取得しているものの，atmarkit.co.jp ドメインで設定されていないため，オーソリティは持たないのである。一方，あるドメインのセカンダリ・サーバが別のドメインのセカンダリ・サーバでもある，という設定も行える。

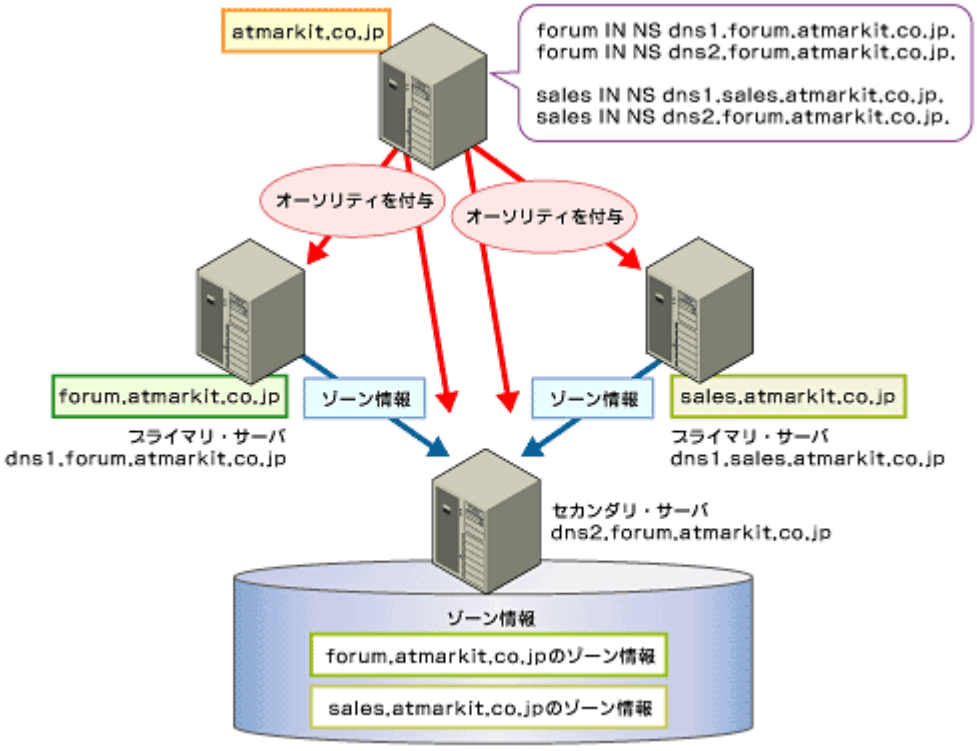

図 3　1 台のサーバで複数ドメインのセカンダリ・サーバを兼ねている例

図 3 の例では，このサーバが複数のドメインのオーソリティを持っている。つまり，このサーバは複数のドメインのゾーン情報を管理していることになる。また，1 台　で複数ドメインの

プライマリ・サーバとなることも可能なほか，（実用上の利点はまた別として）ある 2 台のサーバがそれぞれ一方のドメインのプライマリ・ サーバであり，もう一方のドメインのセカンダリ・サーバとなるような複雑な設定も可能だ。

結局のところ，オーソリ ティとはあくまでゾーン情報の権威の問題であって，物理的なサーバ配置に依存するものではない。ゾーンのオーソリティという観点からは，権限委譲とは，実 はゾーン情報の管理権限を別のサーバ（サブドメインの DNS サーバ）に引き渡し，その「証」に自ドメインのゾーン情報へ記載することであり，ドメインの 「信頼の連鎖」とは，ゾーンというデータの信頼連鎖である，と理解いただけるだろう。

BIND を設定してみよう

1 DNS サーバの役割と BIND の導入

インターネットの基幹となる技術，それが DNS だ。 DNS がなければインターネットはきわめて使いにくいものになっていただろう。ここでは，BIND で DNS サーバを構築しながら DNS そのものについても解説していく。

BIND と DNS

BIND は Berkeley Internet Name Domain の頭文字をとったもので，いわゆる DNS サーバとリゾルバライブラリ，各種ツールの集合体です。DNS は Domain Name System で，今日のインターネットを支える基幹技術の 1 つといって差し支えないでしょう。DNS の目的は，ホスト名と IP アドレスを関連づけることで す。そしてその仕掛けは，名前空間の階層化と権限委譲による分散型データベースに集約されます。

もともとTCP/IP ベースのネットワークとしてのインターネットでは，すべてのホストは IP アドレスで区別されます。0～255 の数値を 4 つ，ピリオド で区切って並べて表記しますが，人間にとっては覚えやすいものではありません。当然，分かりやすい名前でアクセスできるように対応表が用意されました。し かしインターネットに接続されるホストが増えると，これを人手でメンテナンスするのは不可能です。そこで名前空間を階層化するとともに分散型データベース としての DNS サーバが導入されたのです。

名前空間の階層化は，要するにピリオドで区切るということです。習慣でより広 範囲を表すラベルを右側に書くので，右から左に行くにつれて徐々に範囲が狭くなっていくわけです。こうすることで，各ドメインごとに一意の名前を使いさえ すれば，インターネット全体を通して名前が重複せずに済みます。

この階層に対応したサーバを用意することで，権限の委譲も可能になります。各サーバはインターネット上のホストすべてを知る必要はなく，下位のドメインに関する権限を持ったサーバを知っているだけで済みます。 知らない情報に対してはルートネームサーバを始点としたたらい回しで処理されるのですが，それこそ電子的な速度で処理されるので人間からすれば大した遅れには見えません。

2 名前解決の仕組みとゾーンファイルの設定

今回は，BIND の設定を行う。ゾーンファイルの編集を行って正引き・逆引きが行えるようにするほか，MX，CNAME など各種レコードの使い方を紹介する。また，名前解決の仕組みについてもここで理解しておいてほしい。

BIND の基本的な動作

BIND を設定する前に，BIND の動作を簡単に理解しておく必要があります。そうせずに，単に資料の引き写しの設定ファイルを使う方法もありますが，予期せぬ動作をしたときに対処できなくなってしまいます。

これまでに，「DNS は分散型データベースである」と述べました。つまり，どこかにすべてのデータを持ったサーバがあるわけではなく，あちらこちらにサーバが分散しているわけです。問題は，どうやって目的のデータを持ったサーバを見つけだすかです。

さすがに手掛かりゼロではどうしようもないので，最初の手掛かりとしてルートネームサーバが用意されています。いまのところ，ルートネームサーバは世界に 13 個存在します。BIND は問い合わせを受けると，まずこのルートネームサーバに照会します。すると，「そのドメイン名ならこのサーバに権限を委譲して いる」というデータが返ってきます。そこで，指示されたサーバに再び照会し直すと，さらに「そのドメイン名ならこのサーバに権限を委譲している」という回 答が返ってきます。こうして権限を委譲されたサーバをたどっていくことで，最終的に「そのドメイン名の IP アドレスは 210.xxx.xxx.xxx で す」と答えてくれるサーバに行き着くわけです。

もう少し具体的に，www.atmarkit.co.jp の IP アドレスを問い合わせるとしましょう。まず，BIND はルートネームサーバに照会します。 するとルートネームサーバは「jp ドメインはサーバ A に権限を委譲している」という答えを返します。ついで BIND はサーバ A に対して www.atmarkit.co.jp の IP アドレスを問い合わせます。するとサーバ A は「co.jp ドメインはサーバ B に権限を委譲している」と答えま す。サーバ B に同じような問い合わせを行うと，「atmarkit.co.jp ドメインはサーバ C に権限を委譲している」と回答します。ここでサーバ C に問い合わせると「www.atmarkit.co.jp の IP アドレスは 211.2.252.66 である」となって，無事解決できるわけです。

なお，ここでは単純にドメインごとにサーバを分けていますが，自前の DNS サーバを持たないドメインもあります。この場合は，より上位の DNS サーバが該当するドメインに対する問い合わせに答えます。

もちろん，問い合わせがあるたびにいちいちルートネームサーバから照会すると無駄なトラフィックが発生しますから，ある程度は BIND がデータをキャッシュしています。

BIND の設定

基本的な動作が分かったところで，順に設定ファイルを見ていきましょう。前提として OCN エコノミーなどで，/28（ネットワークアドレスが 28bits でホストアドレスが 4bits）の IP アドレスを割り当てられたことにします。この場合，ISP から DNS サーバのホスト名と IP アドレスを指定されるの で，そのコンピュータで BIND を動かします。DNS サーバの IP アドレスは，だいたい下位 4bits だけを見て 2 というのが一般的なようです（0 はネッ トワークそのもの，1 はルータに対して割り当てられることが多い）。

まずは，BIND そのものの基本設定ファイルです。これは通常/etc/named.conf ですが，ディストリビューションによっても違ってきます。例えば LASER5 Linux 6.4 はこのとおりですが，Debian GNU/Linux 2.2 では/etc/bind/named.conf です。必要最小限の設定を挙げると，以下のようになります。

```
options {
        directory "/etc/namedb";
};
zone "." {
        type hint;
        file "named.root";
};
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};
zone "atmarkit.co.jp" {
        type master;
        file "named.hosts";
};
zone "80.1.168.192.in-addr.arpa" {
        type master;
        file "named.rev";
};
```

リスト 1 named.conf

上から順に説明しましょう。まず options ですが，ここでは directory を使って BIND の設定ファイルがあるディレクトリを指定しています。 /etc/bind などでもいいでしょう。named.conf 以外は，明示的に指定したこのディレクトリにまとめておいた方が分かりやすくなります。

次の zone "."は，ルートネームサーバを記述したファイルを指定します。ここではファイル名しか書いてありませんから，前述の options で指定したディレクト リ内の named.root，すなわち/etc/namedb/named.root を意味しています。

zone "0.0.127.IN-ADDR.ARPA"は，localhost を処理するためのおまじないです。ここはどんなサイトでも同じになります。

zone "atmarkit.co.jp"では，atmarkit.co.jp ドメインのプライマリネームサーバになること，正引き（ドメイン名から IP アドレスを引くこと）のデータは named.hosts に記述することを指定します。

zone "80.1.168.192.in-addr.arpa"は逆引きの設定で，やはりプライマリネームサーバであることと，逆引き（IP アドレスからドメイ ン名を引くこと）のデータは named.rev に収められていることを意味しています。なお，ここでは IP アドレスとして 192.168.1.80/28 を想定しています。これはプライベート IP アドレスなので，実際には外部に向かってアナウンスすべきものではありません。実際に設定を行うときは，ISP から割り当てられた IP アドレスに適宜読み替えてください。また，zone に続く記述では IP アドレスを逆順に表記する点にも注意が必要です。

ルートネームサーバの設定

これは簡単です。named.conf で指定したファイルに，ルートネームサーバを記述しておけばいいのです。最新のデータは ftp://ftp.rs.internic.net/domain/named.root です。これをそのままコピーすれば OK です。

正引きの設定

続けて，正引きのデータを用意します。サンプルとしては以下のようになります。

```
@       IN      SOA     dns.atmarkit.co.jp.  root.atmarkit.co.jp.  (
        2000122401
        10800
        3600
        604800
        86400 )
;
        IN      NS              dns.atmarkit.co.jp.
        IN      NS              ns-tk023.ocn.ad.jp.
;
router  IN      A       192.168.1.81
;
sun     IN      A       192.168.1.82
mercury IN      A       192.168.1.83
vinus   IN      A       192.168.1.84
earth   IN      A       192.168.1.85
mars    IN      A       192.168.1.86
jupiter IN      A       192.168.1.87
saturn  IN      A       192.168.1.88
uranus  IN      A       192.168.1.89
neptune IN      A       192.168.1.90
```

リスト 2　named.hosts（正引き用ローカルゾーンファイル）

最初に，SOA レコードを記述します。冒頭の@マークは，自ドメインに置き換わります。正確を期すなら，atmarkit.co.jp.などと明示的に記述します。ここで，ドメイン名の最後にピリオドが付いていることに注意してください。

続く IN は InterNet を意味しています。ほかにも指定できるパラメータはあるのですが，今日では IN しか使われていないといって差し支えないでしょう。SOA は Start Of Authority の略です。続けて，このデータが格納されているホスト名と，このデータに責任を持つ人間の電子メールアドレスを記述します。いずれもピリオドが最後に付くこと，@を.に置き換える点に注意が必要です。

続く数字は，セカンダリネームサーバに対する指定です。最初にシリアル番号がきます。これはデータを変更したときにより大きな数値に変更することで，ほかの DNS サーバに変更があったことを知らせます。何でもいいのですが，2000122401 のように**年月日＋連番**にしておくと，いつ変更したのか分かりやすくなります。

あとは，データを確認する間隔，失敗後再試行の間隔，データが無効になるまでの時間，データをキャッシュしておける時間を，それぞれ**秒数**で指定します。ここを変更する必要はほとんどありません。

SOA の次には，NS レコードを記述します。NS は **N**ame **S**erver の略で，ネームサーバを指定します。自分自身と，普通は ISP が用意するセカンダリネームサーバを指定しておきます。ここでも，ドメイン名の最後にピリオドが付いていることに注意してください。

ようやく，個々のホスト名を記述するところまできました。ホスト名は A（**A**ddress）レコードで記述します。ここでは，ホスト名の後ろにピリオドを付ける必要はありません。IP アドレスも普通に記述します。

■コラム　ピリオドの有無

BIND の正引き，逆引き設定ファイルでは，ピリオドの有無に気を付ける必要があります。それは，ピリオドがないと named.conf で指定したドメイン名が補完されるからです。www とだけ書くと www.atmarkit.co.jp になるのですが，www.atmarkit.co.jp と書くと www.atmarkit.co.jp.atmarkit.co.jp と書いたことになってしまうのです。一般的には FQDN を記述するよりもホスト名を 記述する頻度が高いでしょうから，その手間を減らすためにこのような仕様になっているのでしょう。

ホスト名の付け方に関しては，お好み次第です。小規模なうちは惑星，星座あたりを付ける人が多いようです。大規模になると，どうしても設置場所などをもとにした記号になってしまうようですが。

なお，ドメイン名が何を意味するのかは微妙です。例えば www.atmarkit.co.jp と書いた場合，www がホスト名で atmarkit.co.jp がドメイン名，と考えるのが普通です。しかし，RFC 1034 の 2.4. Elements of the DNS あたりを読むと，どうも www.atmarkit.co.jp でドメイン名となるようです。このあたりの誤解を避けるため，ホスト名を含んだドメイン名の記述を FQDN（**F**ully **Q**ualified **D**omain **N**ame：完全修飾ドメイン名 ）と呼んでいます。

逆引きの設定

正引きデータができたら，今度は逆引きデータです。サンプルは以下のとおり。

```
@       IN      SOA     sun.atmarkit.co.jp.  root.atmarkit.co.jp.  (
        2000081501
        10800
        3600
        604800
        86400 )
;
        IN      NS              dns.atmarkit.co.jp.
        IN      NS              ns-tk023.ocn.ad.jp.
;
        IN      PTR     atmarkit.co.jp.
        IN      A       255.255.255.240
;
81      IN      PTR     router.atmarkit.co.jp.
82      IN      PTR     sun.atmarkit.co.jp.
83      IN      PTR     mercury.atmarkit.co.jp.
84      IN      PTR     vinus.atmarkit.co.jp.
85      IN      PTR     earth.atmarkit.co.jp.
86      IN      PTR     mars.atmarkit.co.jp.
87      IN      PTR     jupiter.atmarkit.co.jp.
88      IN      PTR     saturn.atmarkit.co.jp.
89      IN      PTR     uranus.atmarkit.co.jp.
90      IN      PTR     neptune.atmarkit.co.jp.
```

リスト 3　named.rev（逆引きファイル）

SOA レコードと NS レコードについては，正引きのときと同じです。

逆引きのデータについては，PTR（PoinTeR） レコードで記述します。最初だけは例外的に，どこのドメイン名を扱うのかを宣言します。ここでは atmarkit.co.jp を指定しますが，最後にピリ オドが付くことを忘れないでください。続く A レコードでは，これも例外的にネットマスクを記述します。とりあえず，/28 のネットワークでは 255.255.255.240 になります。

ネットマスクは BIND や DNS ではなく，どちらかといえば IP ネットワー クの知識なのですが，簡単に説明しておきましょう。IP アドレスは 2 進数で書くと 32 けたです。そして，ネットワークを区別するためのネットワークアドレ スと，ネットワーク内でホストを区別するためのホストアドレスで構成されています。このとき，上位何 bits をネットワークアドレスとして使うかを，/に続けた数値で表します。つまり 192.168.1.80/28 と書くと，192.168.1.80 から始まる，ネットワークアドレス長が 28bits の ネットワークということになります。さらにいえばホストアドレスは 4(＝32－28)bits ですから 16 通りです。従って，192.168.1.80- 192.168.1.95 までがこのネットワークに所属する IP アドレスということになります。

このことを別の方法で表記するのが，ネットマスクです。こちらの方法だと，IP アドレスのうちネットワークアドレス部を 1 に，ホストアドレス部を 0 にしたものを，通常の IP アド レスの表記に従って 0～255 の数値 4 つで記述します。これはちょうど 2 進数で 8 けたに相当する範囲です。なので，/28 のネットワークだと最初の 3 つは 255，最後の 1 つは 2 進数表記で 11110000，10 進数表記で 240 となるのです。

さて，以後は普通に IP アドレスとホスト名を PTR レコードとして記述していくだけです。ここでも，ドメイン名の最後にピリオドを付加します。また，IP アドレスは最も下位の 1 オクテットのみを記述するだけです。

最後に，もう 1 つループバック用の逆引きファイルを用意します。これで BIND を動かすために必要な最低限の設定ファイルがそろったことになります。

```
@       IN      SOA     dns.atmarkit.co.jp.  root.atmarkit.co.jp.  (
        2000081501
        10800
        3600
        604800
        86400 )
;
0.0.127.in-addr.arpa.       IN      NS      dns.atmarkit.co.jp.
;
1.0.0.127.in-addr.arpa.     IN      PTR     localhost.
```

リスト 4　localhost.rev(ループバック用逆引きファイル)

MX の設定

以上で基本的な設定は終了です。しかし，まだ重要な設定が残っています。それはメールサーバを指定する MX（**M**ail e**X**changer）レコードです。OCN エコノミーで DNS サーバを自組織で用意するということは，すなわちメールサーバも自組織で用意するということです。そのため，MX レコードの設定は欠かせません。

MX レコードは，正引きデータを記述したファイルに

```
        IN      MX      10      sun.atmarkit.co.jp.
        IN      MX      20      mercury.atmarkit.co.jp.
```

と いった形で指定します。ここで MX に続く数値はプリファレンス値といいます。この数値が小さい順に，順次メールサーバとしてアクセスするという指定になり ます。相対的な大きさだけが問題なので，符号なし 16bits 数値の範囲なら好きなように設定できます。つまり，予備のメールサーバを作っておけるわけで す。

CNAME の設定

もう 1 つ，便利な設定に CNAME（**C**anonical **NAME**）レコードがあります。これは，1 つの IP アドレスに幾つかのホスト名を割り当てるときに使います。これも MX レコード同様に，正引きデータを記述したファイル内に記述します。例としてはこんなところでしょうか。

```
dns     IN      CNAME   sun
www     IN      CNAME   sun
smtp    IN      CNAME   sun
pop     IN      CNAME   sun
imap    IN      CNAME   sun
proxy   IN      CNAME   mercury
rfc     IN      CNAME   mercury
```

サービス名をそのまま CNAME で設定しておくと，実際にサービスを提供しているコンピュータが変わったときに，BIND のデータを変更するだけで済みま す。個々のクライアントは，一切変更する必要がないわけです。ただ，この方法は外部の悪意を持ったユーザーから容易に推察できるという点で，セキュリティ 的にはやや問題のある方法です。

また，Apache と組み合わせてバーチャルホストを使うときにも，CNAME は欠かせません。この方法を使うと，1 つの IP アドレスを割り当てた 1 台のホストで，見かけ上は複数の Web サーバを運用できます。

最後に，いま紹介した MX および CNAME レコードを正引きファイルに反映してみましょう。以下のリストが完成版です。

```
@       IN      SOA     dns.atmarkit.co.jp.  root.atmarkit.co.jp.  (
        2000122401
        10800
        3600
        604800
        86400 )
;
        IN      NS              dns.atmarkit.co.jp.
        IN      NS              ns-tk023.ocn.ad.jp.
        IN      MX      10      sun.atmarkit.co.jp.
        IN      MX      20      mercury.atmarkit.co.jp.
;
router  IN      A       192.168.1.81
;
sun     IN      A       192.168.1.82
mercury IN      A       192.168.1.83
vinus   IN      A       192.168.1.84
earth   IN      A       192.168.1.85
mars    IN      A       192.168.1.86
jupiter IN      A       192.168.1.87
saturn  IN      A       192.168.1.88
uranus  IN      A       192.168.1.89
neptune IN      A       192.168.1.90
;
dns     IN      CNAME   sun
www     IN      CNAME   sun
smtp    IN      CNAME   sun
pop     IN      CNAME   sun
```

リスト 5　完成版 named.hosts（正引き用ローカルゾーンファイル）