分散システム特論レポート

RFC1034 DOMAIN NAMES - CONCEPTS AND FACILITIES
RFC1035 DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

情報工学専攻 修士課程 1 年 岡村研究室
長尾孝行

平成 14 年 2 月 27 日

1 概要

DNS(Domain Name System) はホスト名と IP アドレスのマッピングを分散管理して実現するものである。以前はホスト名とアドレスのマッピングは Network Information Center(NIC) により単一ファイル HOSTS.TXT により管理されていた。このマッピング表は各ユーザが FTP を実行することにより入手していた。しかし、このような単一管理は以下に示す問題点を含んでいる。

- ネットワークトラフィック, プロセッサの負荷の増大
- HOSTS.TXT 内の名前の衝突
- ネットワーク上の全てのマシンが一貫して同一の HOSTS.TXT を持つことができない

DNS は分散データベースであり、データベースの各部分はローカルで管理されるが、クライアントサーバメカニズムによりそれぞれの部分はネットワーク全体で利用される。また、データの複製とキャッシュによりシステムの安定性と十分な性能を達成している。

2 DNS の構造

DNS は以下の 3 つの要素で構成される。

- ドメイン名空間と資源レコード (RR)
  木構造の名前空間の記述
  名前と結びつけられたデータに関する記述

- ネームサーバ
  ドメイン木構造に関する情報の保持・情報の設定
  データベースの一部分の情報 (ゾーン) の管理

- リゾルバ
  ネームサーバへの問い合わせを行う client
  ユーザプログラムから直接実行可能なシステムルーチン

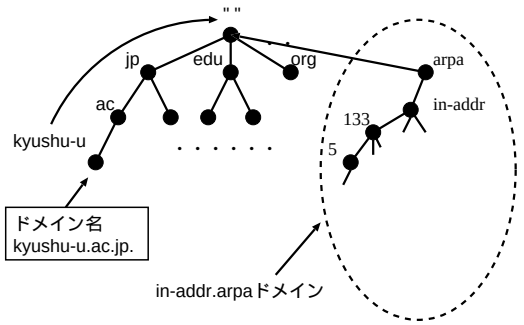

図 1: ドメイン名空間の例

図 1 は、ドメイン名空間を示す木構造の例である。ドメイン名とはあるノードからルートノードにたどり着くまでに経由するラベルを列挙したものである。このドメイン木の各ノードは最大 63 オクテット長のラベルがつけられ、根 (ルートドメイン) は 0 オクテット長である。また、ドメイン名は最大 255 オクテット長と定められている。

それぞれのノードは資源レコード (RR: Recource Record) と呼ばれる情報資源の集合を持つ。資源レコードの内容は以下の通りである。

- 型
  符号化された 16bit の値
  資源レコード内の型の指定
  - A : ホストの IP アドレス
  - CNAME : エイリアスに対する正式名
  - HINFO : CPU,OS の情報
  - MX : ゾーンのメールを管理するメールサーバ
  - NS : ドメインに対して権威を持つネームサーバ
  - PTR : 逆引きドメイン名に対するホスト名
  - SOA : ゾーン全般に関するパラメータの定義
- クラス
  符号化された 16bit の値
  プロトコルの一つやプロトコルを示す

  - IN : インターネットシステム
  - CH : カオスシステム

- TTL
  32bit 整数値 (単位:秒)
  RR をキャッシュできる時間

- RDATA
  型に依存するデータを保持

  - A : 32bit の IP アドレス (IN)
  - CNAME : ドメイン名
  - MX : 16bit の優先度とメイルサーバのホスト名
  - NS : ホスト名
  - PTR : ドメイン名
  - SOA : 幾つかのフィールド値

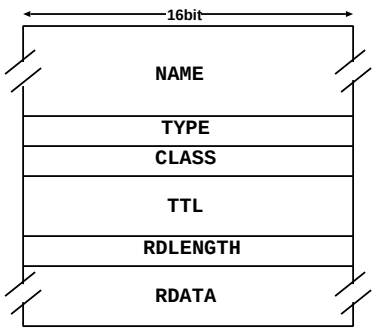

図 2: 資源レコード (RR) のフォーマット

図 2 に示すように資源レコード中の NAME,RDATA フィールドは可変長であり、RDLENGTH は RDATA の大きさを格納する。この資源レコードは後で述べる問い合わせ、応答パケットの一部として用いられる。

また、資源レコード中の TYPE,CLASS フィールドのうち使用されていない値は型やクラスを拡張する際のために予約されている。

3 DNS の分散管理

ドメイン名空間に関する情報を処理するプログラムはネームサーバであり、ネームサーバはゾーンと呼ばれるドメイン名空間の一部について完全な情報を管理している。このとき、そのネームサーバはゾーンに関する権威を持つという。例として図 3 に示すドメイン名空間を考える。

図 3 において ac.jp ドメインはそのノードをルートとする部分木全体を指すことになる。従って ac.jp ドメインには ac.jp ドメインおよび、kyushu-u.ac.jp ドメインのデータなどを含むことになる。しかしながら、ac.jp ゾーンには ac.jp のデータしか入っていない。

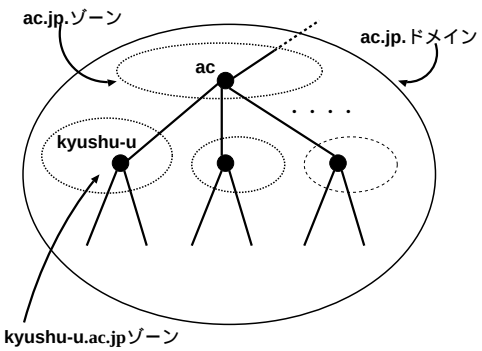

図 3: ゾーンとドメイン

上記の例では、ac.jp ドメインのサブドメインである kyushu-u.ac.jp ドメインは管理を委任されることによって、「管理の分散化」を実現する。管理の分散化の実現のため、各ネームサーバはドメインではなく、ゾーン情報を管理することになる。

実際に委任を行うために ac.jp のゾーン情報には以下のような NS レコードの記述がされなければならない。ac.jp ゾーンを管理するネームサーバが委任をたどるときにそのドメイン名には別名 (CNAME) ではなく、一つまたは複数のアドレス (A) レコードがつけられていなければならない。また、NS レコードだけの記述だけでは ac.jp のネームサーバは委任先の権威あるネームサーバの名前を返すだけで、その IP アドレスを取得することができない。そのため、サブドメインへの委任を行うためには NS レコードに加えて、kyushu-u.ac.jp の権威あるネームサーバの IP アドレスを記述したレコードが必要である。これをグルーレコード (glue record) と呼ぶ。下記の記述例では 2 つの A レコードの記述がグルーレコードに相当する。

```
kyushu-u                86400 IN NS test1.kyushu-u.ac.jp
                        86400 IN NS test2.kyushu-u.ac.jp
test1.kyushu-u.ac.jp 86400 IN A  133.5.x.x
test2.kyushu-u.ac.jp 86400 IN A  133.5.x.x
```

4 DNS への問い合わせ

DNS への問い合わせと応答は UDP を用いて行われ、RFC1035 でメッセージフォーマットは下記の通り規定されている。

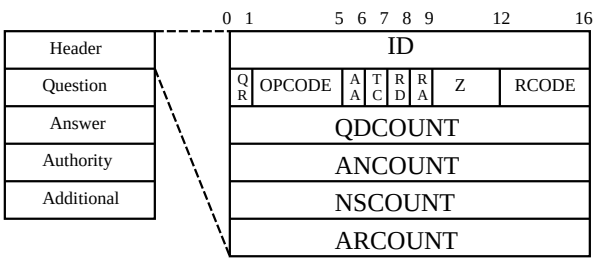

図 4: ドメインプロトコルのメッセージフォーマット

トップレベルのフォーマットは下記の通りに構成される

- ヘッダ部 (Header)
- 問い合わせ部 (Question)
  ネームサーバへの問い合わせ内容を記述したフィールドを持つ
  問い合わせ型 (QTYPE: 16bit)
  問い合わせクラス (QCLASS: 16bit)
  問い合わせ先のドメイン (QNAME: 可変長)
- 回答部 (Answer)
  問い合わせに対する答えとなる RR
  資源レコード (RR) を連結したもの
- 権威部 (Authority)
  権威を持つネームサーバを示す RR
  資源レコード (RR) を連結したもの
- 付加情報部 (Addtional)
  問い合わせに関連した RR
  資源レコード (RR) を連結したもの

DNS への問い合わせを行う際には図 4 のヘッダ部のうち、OPCODE フィールドが参照される。OPCODE フィールドにより DNS に対してどのような問い合わせを行うかを指定する。

- OPCODE 0 : 標準問い合わせ
- OPCODE 1 : 逆問い合わせ
  IP アドレス以外の任意のデータを索引としてドメイン名を検索
  一つのネームサーバ内でのみ検索可能
- OPCODE 2 : 状態問い合わせ
  未定義
- OPCODE 3-15 : 将来のために予約

実際に用いられる DNS への問い合わせの大部分は標準問い合わせ (OPCODE=0) である。標準問い合わせではヘッダ部の QNAME,QTYPE,QCLASS を指定して合致する資源レコードを要求する。問い合わせの際にはポート番号 53 を指定して UDP により行う。また、UDP によるメッセージは 512 バイトに制限されているため、512 バイトを越えるメッセージは切り取られ図 4 のヘッダ部における TC ビットが設定される。この後 TCP を用いてクエリの再送信を行う。

標準問い合わせには以下の 2 通りの方法がある。

- 反復的問い合わせ (iterative query)
- 再帰的問い合わせ (recursive query)

反復的問い合わせではネームサーバが持つ情報の中で最適なものを要求するため、ネームサーバへの負荷が小さくなる。一方、再帰的問い合わせでは要求を受けたネームサーバに対して完全な名前解決を要求する。そのため名前を解決するための負荷のほとんどが一つのネームサーバにかかる。

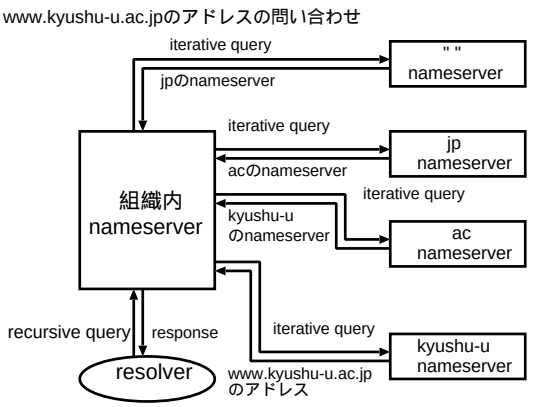

図 5: DNS への問い合わせ

一般に、リゾルバはローカルネームサーバに対して再帰的問い合わせを行い、完全な名前解決を要求する。このようなリゾルバをスタブリゾルバと呼ぶ。リゾルバからの要求を受けたローカルネームサーバはルートネームサーバに対して反復的な問い合わせを行い、名前解決する。図 5 に例を示す。

5 DNS の管理

DNS は信頼性を確実にするため、その仕様においてプライマリマスタとセカンダリマスタという 2 種類のネームサーバが存在する。

- プライマリマスタ
  権威をもつゾーンのデータを自分が動作している自ホストから取得
- セカンダリマスタ
  ゾーンの権威を持つ他のホストからゾーンデータを取得

セカンダリマスタはプライマリマスタから TCP(ポート番号 53) を用いたゾーン転送の問い合わせを発行する。ゾーン転送の際には下記 SOA レコードのフィールドを参照する。

- SERLIAL
  シリアル番号
- REFRESH
  プライマリへの SOA クエリの送信間隔
- RETRY
  REFRESH 間隔で失敗した時の再送信間隔
- EXPIRE
  EXPIRE 間隔プライマリにアクセス不可の時データ破棄

セカンダリマスタはプライマリマスタの SOA レコードの SERIAL を REFRESH の間隔で定期的に確認し、SERIAL が増加した場合にはプライマリのゾーンが更新されたと判断しゾーン転送を行う。ゾーン転送の詳細については RFC2136 で記述されているが、下記の 2 種類がある。

- AXFR : ゾーンの完全転送
- IXFR : ゾーンの増分転送 (RFC1995)

実際のゾーン転送の順序は図 6 のようになる。

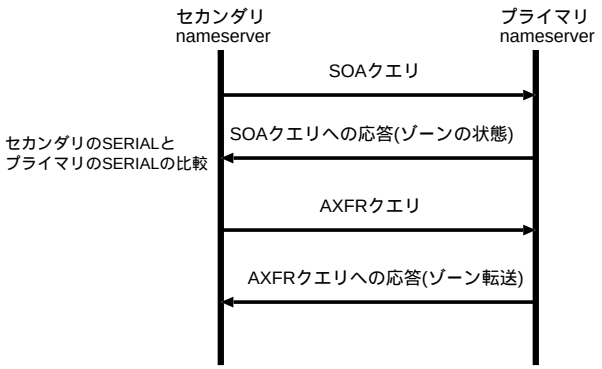

図 6: ゾーン転送

6 DNS の実装

ここでは DNS プロトコルを実装した BIND(Berkeley Internet Name Domain) における実装について簡単に説明する。資料作成時の最新版である BIND Version 9.2.0 では以下のものを含んでいる。

- Domain Name System サーバ (named)
- Domain Name System リゾルバライブラリ

6.1 ネームサーバの実装

- 制御
  - クライアントからの複数の処理を並列処理
  - TCP データのリフレシュやクエリに対するレスポンスを待つ間、UDP リクエストをブロックしない
  - 大文字小文字の区別をせずドメインツリーを探索
- データベースの管理
  - 目録データ (catalog data)
    標準問合せに対して一番近いネームサーバを特定するポインタ (glue)
  - ゾーン
    権威のあるデータ
  - キャッシュ
    権威を持たないデータ

6.2 リゾルバの実装

- res_research
  - 最上位リゾルバライブラリ
  - 関数 gethostbyname により呼び出される
- res_query
  - ドメイン名検索の作業
- res_mkquery
  - 問い合わせパケットの作成
- res_send
  - 再送アルゴリズムの実装
  - 再送パケットを UDP で送信
- res_init
  - resolv.conf を読み込んで_res 構造体を初期化
  - 再送間隔, 再送回数, ネームサーバなどの設定

_res 構造体

```
struct __res_state {
  int     retrans;   /* 再送間隔 */
  int     retry;     /* 再送回数 */
  u_long  options;   /* オプションフラグ */
  int     nscount;   /* ネームサーバの数 */
  struct  sockaddr_in  /* ネームサーバアドレス */
            nsaddr_list[MAXNS];
    .....
}
```

6.3 nslookup の実装例

nslookup のソースコードの一部

```
if (argc == 2) {
......
    if (inet_aton(*++argv, &addr)) {
        res.nscount = 1;
        res.nsaddr.sin_addr = addr;
    } else {
        hp = gethostbyname(*argv);
        if (hp == NULL) {
           ....
        } else {
            for (i = 0; i < MAXNS && hp->h_addr_list[i] != NULL; i++) {
            memcpy(&res.nsaddr_list[i].sin_addr, hp->h_addr_list[i],
                   hp->h_length);
            }
            res.nscount = i;
        }
        .....
```

nslookup では引数として明示的に問い合わせを行うネームサーバを指定して名前解決を行うことができる。上記のソースコードより、第 2 引数で指定した問い合わせ先ネームサーバに対して問い合わせを行う。対応するネームサーバが存在しない場合などは、resolv.conf を読み込んで_res 構造体に保持してあるネームサーバから名前解決を行うようになっている。

実行結果

```
[lets ~]% nslookup horse.csce.kyushu-u.ac.jp hoge
*** Can't find server address 'hoge': Unknown host

Server:  arashi.cc.kyushu-u.ac.jp
Address:  133.5.8.10

Name:    horse.csce.kyushu-u.ac.jp
Address:  133.5.7.47

[lets ~]% nslookup horse.csce.kyushu-u.ac.jp crest.csce.kyushu-u.ac.jp
Server:  crest.csce.kyushu-u.ac.jp
Address:  133.5.22.3

Non-authoritative answer:
Name:    horse.csce.kyushu-u.ac.jp
Address:  133.5.7.47
```