2004 年度 修士論文

DNS におけるリゾルバの動作解析

提出日： 2005 年 2 月 2 日

指導：後藤滋樹教授

早稲田大学 大学院理工学研究科
情報・ネットワーク専攻
学籍番号： 3603U090-1

田中 政史

目次

図一覧

表一覧

第 1 章

序論

1.1 研究の背景

インターネットは、ドメイン名と IP アドレスの対応付けを行う DNS [1]によって、さまざまなサービスが機能している。アジア、ヨーロッパの TLD [2]を中心に多言語ドメイン名の登録サービスも開始され、利用されるドメイン登録数は増加の一途を辿っている。さらに ENUM、 RFID といった DNS の仕組みを利用した新しいシステムの登場もあり、 DNS そのものに対する関心は自然と高まりつつある。 IPv6 への対応や DNS メッセージのセキュリティを向上させる DNSsec、最も利用されている BIND 以外のソフトウェアの充実などの事例もあり、今後もその動向に目が離せない。

1.2 研究の目的

ほぼすべてのインターネットサービスの基盤を支え、大規模な分散データベースである DNS が適切に運用されることは、インターネットの安定性および信頼性を確保するうえで必要不可欠である。それはそれぞれのゾーンが正しく運用されることから導かれる。

しかし DNS はその重要性とは裏腹に、稼働されてる実際の状況はよく把握されていない。その原因は DNS が主に UDP を使うためであり、これは、同じトランスポート層のプロトコルである TCP と比較すると状態の遷移が把握出来ないことに起因する。つまり DNS では問い合わせメッセージを送信してその後は回答が返って来るまで待っているだけである。 DNS の動作を分析するため、本研究ではこの回答が戻って来ない場合、あるいは遅延したりする場合に着目した。それを分析して、調査結果を基に DNS 稼働の正常さ表示する指標を検討する。これを DNS の安定性および信頼性の向上に役立てることを検討する。

---

[1] Domain Name System

[2] Top Level Domain

1.3 本論文の構成

本論文は以下の章により構成される。

第 1章 序論

本研究の概要、構成について述べる。

第 2章 DNS について

DNS の仕組みを解説する。

第 3章 実験の環境

測定方法、環境を述べる。

第 4章 実験の概要

実験の概要を述べる。

第 5章 解析と考察

実験結果を解析、考察する。

第 6章 まとめ

本研究のまとめ、今後の課題を述べる。

第 2 章

DNS の仕組み

2.1 Domain Name 空間

DNS とは、IP アドレスとホスト名をマッピングして解決するシステムで、分散型データベースである。IP アドレスは単なる数字列でしかないので、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」の方が便利である。DNS のドメイン名空間の構造は、UNIX のファイルシステム (図 2.1参照) と似ており、逆木構造になっている。(図 2.2参照)

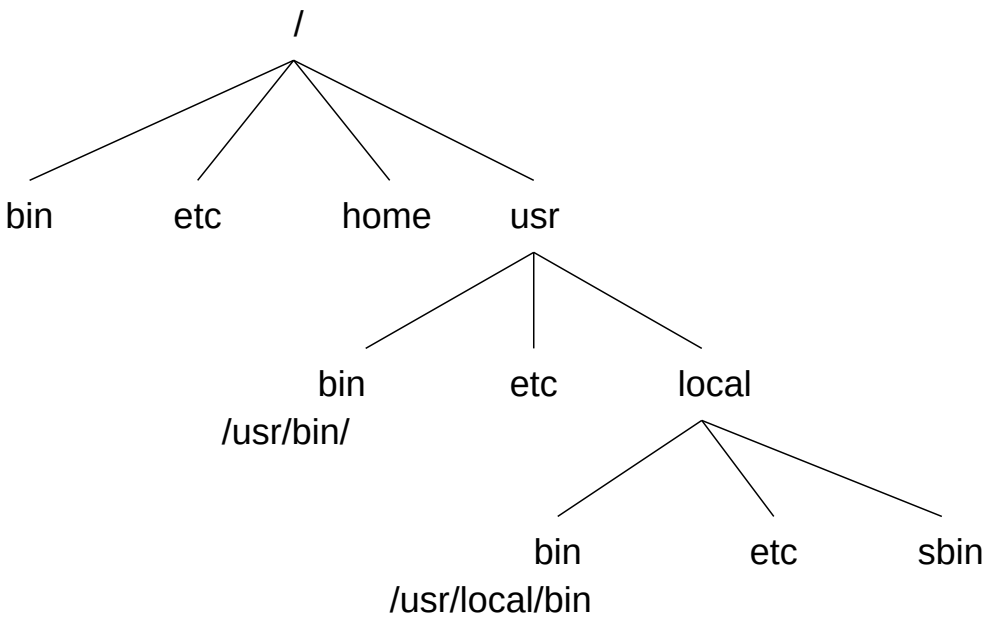

図 2.1: UNIX FileSystem

2.2 委任 (delegation)

階層型のゾーン構造は DNS が作られた目的でもあり、特徴でもある。

- ある階層のノードにおいて同じテキストラベルがなければ、
  ドメインのユニーク性が保証される
- ある階層以下 (ゾーン) の管理をその組織に委任すると、上位階層のサーバから独立する

この 2 つの取り決めによりネームサーバ間のトラフィックが削減され、上位階層のサーバにかかる負荷が軽減する。

ドメイン名における制限

各ノードにはテキストラベルが付けられ、最大 63 文字、英字・数字・ハイフン (-) のみの使用と制限されている。また、長さが 0、つまり空のラベルはルートのために予約されており、他のノードでは使用することができない。木構造の深さ (レベル) にも制限があり、最大 127 レベルまでとなっている。この詳細は第 2.9 節で述べる。

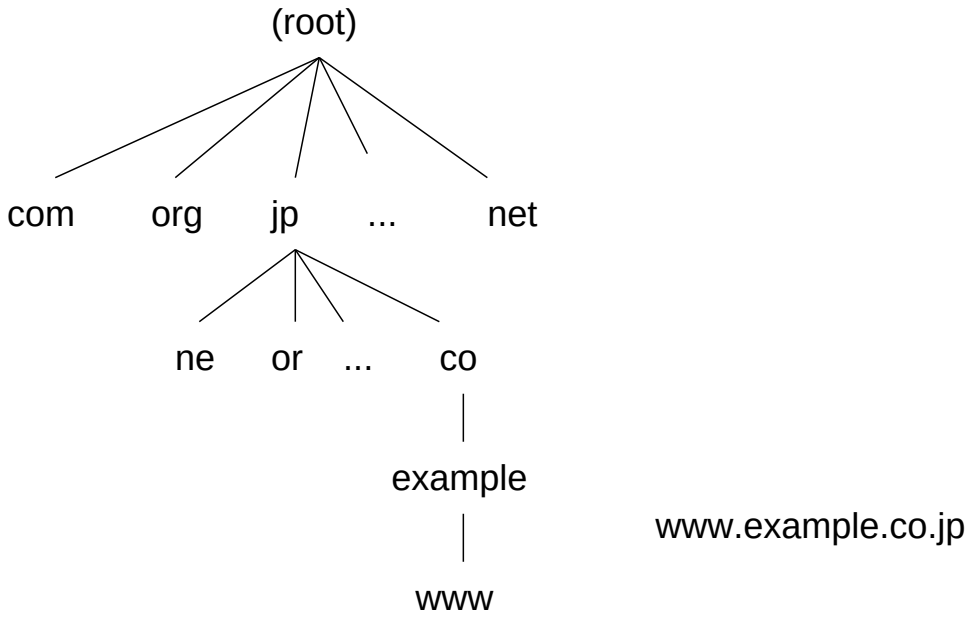

図 2.2: DNS tree

2.3 ルートネームサーバ

ルートネームサーバは、トップレベルの各ドメイン (.com, .net, .org など) の権威を持つネームサーバが、どこに位置しているのかを把握しており、名前の解決において非常に重要な役割を果たす。ドメイン名の問い合せに対し、ルートサーバは最低 1 つ以上のトップレベルゾーンの権威を持つサーバの名前と IP アドレスを返す。

ルートネームサーバには、その性質上多くの問い合せが殺到する。そのため DNS では、ネームサーバの負荷を軽減する機能 (キャッシュなど) を用意している。キャッシュ情報が存在しない場合、問い合せはルートネームサーバから開始される。そのため、インターネットのすべてのルートネームサーバが一斉に停止してしまったら、どの名前も解決できなくなってしまう。このような事態を避けるために、世界には同じデータを持ったルートサーバが 13 台あり、負荷分散している。

米 VeriSign 社に 2 台、ネットワーク管理団体 IANA、ヨーロッパのネットワーク管理団体 RIPE-NCC、米 PSINet 社、米 ISI (Information Sciences Institute) 、米 ISC (Internet Software Consortium) 、米 Maryland 大学、米航空宇宙局 (NASA) 、米国防総省、米陸軍研究所、ノルウェーの NORDUnet、日本の WIDE プロジェクトで各 1 台ずつを管理している。

以下に主なドメインの種類を挙げる。 (表 2.1, 2.2 参照)

表 2.1: TLD の種類 (gTLD)

TLD 名	登録資格要件	管理団体（レジストリ）
.com	制限なし	ICANN (VGRS)
.net	制限なし	ICANN (VGRS)
.org	制限なし	ICANN (VGRS)
.edu	米国内大学	Educause
.gov	米国政府機関	U.S. General Services Administration
.mil	米軍関係者	DoD Network Information Center
.int	国際機関	IANA

表 2.2: SLD の種類 (.jp ドメイン)

SLD 名	登録資格要件
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	ネットワーク管理組織（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

2.4 リゾルバ (resolver)

2.4.1 代表的な機能

- ネームサーバへの問い合わせ
  - ホスト名から IP アドレスへの変換
  - IP アドレスからホスト名への変換
  - その他の検索機能
- 応答の解釈 (資源レコードまたはエラーのどちらであるか)
- 要求側プログラムへの情報の返送

リゾルバはユーザプログラムと DNS サーバをつなぐプログラムである。多くの場合、ユーザプログラム（メールプログラム、 TELNET、 FTP など）からサブルーチンを呼び出し、システムコールなどの形で問合せを受け取り、ローカルホストのデータ形式と互換性のある形式で目的の情報を返す。このような挙動のリゾルバを厳密にはスタブリゾルバ (stub resolver) と呼び、関数やライブラリとして OS から提供されることが多い。

それに対しフルサービスリゾルバ (full-service resolver) は、 Windows の「TCP/IP のプロパティ」や UNIX 系 OS の /etc/resolv.conf などで指定するもので、スタブリゾルバの動作を制御するものである。ほとんどの UNIX では、代表的 DNS サーバである BIND [1] が提供するライブ

[1]Berkeley Internet Name Domain

ラリルーチンを用いることによって DNS での名前解決を行っている。ここで、問い合わせた情報の回答がローカルキャッシュ (第 2.6節参照) に存在することもあるので、処理を完了するまでの所要時間はミリ秒から数秒まで と大きな幅がある。このようにキャッシュを使う事によって、多くの問い合わせについてネットワークの遅延とネームサーバへの負荷を軽減できる事がある。このようなことを実現するのも、リゾルバの非常に重要な目的である。

キャッシュの有効性については、参考文献 [6], [7] を参照されたい。

2.4.2 タイムアウトについて

ネームサーバが 1 つしか設定されていない場合は、そのネームサーバにタイムアウトを 5 秒に設定し、問い合わせを送る。このタイムアウトとは、次の問い合わせを送るまでにリゾルバが回答を待つ時間である。リゾルバは、ネームサーバが本当に停止している、ないし到達できなくなっていることを示すエラーを受け取るか、またこの設定されたタイムアウト値になった場合、その値を 2 倍に設定し再び同じネームサーバに問い合わせを送る。この動作を起こすエラーには以下の 2 つがある。

- ICMP ポート到達不能メッセージの受信
  問い合わせ先のネームサーバで、問い合わせを待っているポートが無いことを意味する。
- ICMP ホスト到達不能メッセージ、ネットワーク到達不能メッセージの受信
  問い合わせを宛先 IP アドレスに届けることが出来ないことを意味する。

複数のネームサーバが設定されている場合

複数のネームサーバが設定されている場合には、リゾルバはまずリストの先頭のネームサーバに対してタイムアウトを 5 秒に設定して問い合わせを送る。ここまでは 1 つのときと同じである。リゾルバがタイムアウトを起こすか、ネットワークエラーを受け取ると同じタイムアウト値 (つまり 5 秒) でリストの次にあるネームサーバに問い合わせる。

しかしリゾルバは、発生し得るエラーの多くを受け取ることが出来ない。そもそもリゾルバは問い合わせを送ったすべてのネームサーバから回答を受け取る必要があるために「未接続の」ソケットを使っているが、この「未接続の」ソケットは ICMP エラーメッセージを受け取らないためである。リゾルバが設定されているすべてのネームサーバから回答を得られなかった場合、タイムアウト値を更新して同じサイクルを繰り返す。

この新しいタイムアウト値は、一般的には resolv.conf ファイルに設定されてるネームサーバの数によって決まる。 2 巡めの問い合わせで使われるタイムアウト値は 10 秒を、設定されたネームサーバの数で割った値となる (秒未満は切り捨て) 。それに続く巡回では、前回のタイムアウト値の倍の値が使われる。

問い合わせの再送を 3 周回行った後 (つまり設定されたネームサーバに対して各 4 回のタイムアウト) リゾルバはネームサーバへの問い合わせを諦める。

BIND 8.2.1 で ISC (Internet Systems Consortium) は、 resolv.conf の各ネームサーバに対して、リゾルバが 1 組のリトライ (最大で 2 回) しか送らないように変更した。これはどのネームサーバからも回答が得られない、応答が得られない場合にユーザがリゾルバからの返答を待つ時間を短縮することが狙いである。

表 2.3: BIND4.9 から BIND8.2 までのリゾルバのタイムアウト値

	ネームサーバの数		
リトライ	1	2	3
0	5 秒	5 秒	5 秒
1	10 秒	5 秒	3 秒
2	20 秒	10 秒	6 秒
3	40 秒	20 秒	13 秒
合計	75 秒	80 秒	81 秒

表 2.4: BIND8.2.1 以降のリゾルバのタイムアウト値

	ネームサーバの数		
リトライ	1	2	3
0	5 秒	5 秒	5 秒
1	10 秒	5 秒	3 秒
合計	15 秒	20 秒	24 秒

この表では最も最悪の場合を想定している。正常に動いていると思われるホストではリゾルバは 1 秒よりもかなり短い時間で回答を得ることが出来るはずである。設定されているすべてのネームサーバが非常に忙しいかまたは停止している時や、ネットワークが停止している場合に限り、リゾルバはこのように長い再送手順をすべて行った上で最終的に諦めることになる。

2.4.3 グルーレコード

グルーレコード (glue record) とは、コンテンツサーバが、委任されたサブドメインの NS を返す際に、追加の情報として必要となる情報のことである。具体的には、ネームサーバの名前に対

応する IP アドレスの情報のこである。サブドメインの委任をしている場合、上位のコンテンツサーバに委任先のドメイン名の問い合わせがあると、委任先を管理するネームサーバの名前を返すが、ネームサーバとの通信を行うためには、名前ではなく IP アドレスを知る必要がある。つまり、教えてもらったネームサーバの名前を再度 DNS で調べることになる。もしここで、返ってきたネームサーバの名前が委任先のドメイン名であった場合、この状況がループを起こし、どうやっても委任先のドメインの情報を得ることができなくなってしまう。これを防ぎ、上位のゾーンと下位の情報をつなぎ止めるために使われるもの。

例えば、 example.jp のネームサーバとして ns1.example.jp を指定してある場合、 jp のネームサーバは example.jp に関する問い合わせに対して NS として ns1.example.jp を答えるだけでなく、 ns1.example.jp の IP アドレスとして A レコードも同時に答える。このグルーデータはドメイン名のツリー構造から考えると、コンテンツサーバが管理するゾーンからは外れた情報であり、本来であれば不要なものである。しかし、リゾルバが IP アドレス による通信で情報を検索するためにはどうしても必要である。

2.5 名前解決の方法

本節では、ホスト名と IP アドレスの変換 (以下、名前解決と呼ぶ) が、どのように行なわれているのかを説明する。

まず、リゾルバから自分の属するドメイン (以下ローカルと呼ぶ) のネームサーバへの問い合わせを出す。次にローカルのネームサーバは問い合わせの内容に対する答えを持っているならばそれを返すが、無い場合はリゾルバに代わってそれに最も近いドメインのネームサーバに問い合わせを出す。ここで、最も近い答えが無い場合はルートネームサーバに問い合わせを出す。問い合わせを受けたサーバは、上記と同様にして、問い合わせに対する答えを持っていればそれを返し、そうでなければ最も近いドメインのネームサーバの IP アドレスを返す。

ここで、なぜローカルのネームサーバはリゾルバに次の参照先 (次に問い合わせを出すべきネームサーバ) を返さずに、自ら問い合わせを出すのかという疑問が生じる。これはリゾルバ自身がそのような再帰的に問い合わせる機能を持ち合わせていないためであり、これによってクライアントマシンの負荷を軽減し、１つのネームサーバだけに負荷をかけるようになっている。

以上のようにして、再帰的に答えが出るまで問い合わせを繰り返し答えを得る。この場合の「答えが出る」と言うことは、 IP アドレスが分かる、または問い合わせたホスト名は見つからないと言う結果に至ることである

例として、’ www.example.co.jp’ の名前解決の流れを示す。図 2.3参照。

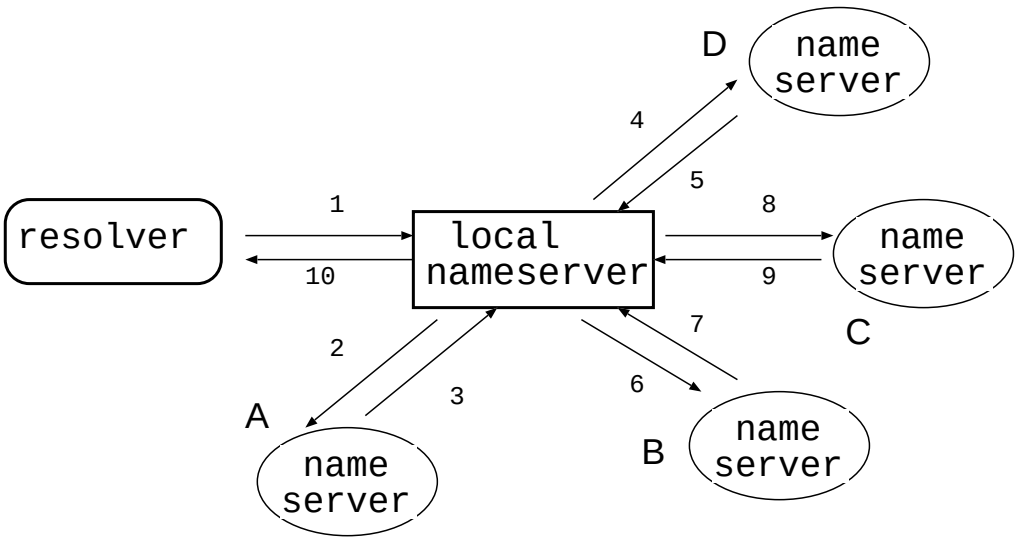

図 2.3: 問い合せから回答までの流れ

1. リゾルバが ’ www.example.co.jp’ の IP アドレスを
   ローカルネームサーバ (以下 NS) に問い合わせる。
2. NS がルートネームサーバ (以下 A) に ’www.example.co.jp’ の IP アドレスを
   問い合わせる。
3. A は、 jp ドメインのネームサーバ (以下 D) の IP アドレスを返す。
4. NS は、 D に ’ www.example.co.jp’ の IP アドレスを問い合わせる。
5. D は、 co.jp ドメインのネームサーバ (以下 B) の IP アドレスを返す。
6. NS は、 B に ’ www.example.co.jp’ の IP アドレスを問い合わせる。
7. B は、 example.co.jp ドメインのネームサーバ (以下 C) の IP アドレスを返す。
8. NS は、 C に ’ www.example.co.jp’ の IP アドレスを問い合わせる。
9. C は、’ www.example.co.jp’ の IP アドレスを返す。
10. NS は、リゾルバに’ www.example.co.jp’ の IP アドレスを返す。

上図ではネームサーバのキャッシュが全く働かない場合であった。
キャッシュが有効である場合、ネームサーバは問い合せをせずに回答をリゾルバに返す。問い合せたデータの不在情報 (ネガティブキャッシュ) が保持されている場合でもリゾルバにそのまま返

す。問い合せの対象となるゾーンの権威をもつネームサーバについての情報が一部でも残っている場合には、リゾルバはそれらのネームサーバに直接問い合せを出すことが出来る。これはネットワークトラフィックを大幅に減少させる。

2.6 キャッシュ (cache)

再帰的な問い合せを処理するネームサーバは、答が得られるまでに多くの問い合せを別々のネームサーバに送る。この過程で得られる多くの情報を、後で関連した問い合せ受けた場合に備えてデータを保存することをキャッシングという。また、この保存されたデータのことをキャッシュと呼び、名前解決の高速化・トラフィックの抑制・ネームサーバの負荷軽減といった様々なメリットを提供する。

また、名前解決に必要な一部のネームサーバがダウンしていても、キャッシュ情報で補完することができれば、ネームサーバのダウンに左右されずに名前解決を行う事ができる。
さらに、 BIND4.9 からは、解決したドメイン名が存在しなかった場合には、その不在情報もキャッシュされる。これはネガティブキャッシュ (negative cache) と呼ばれ、 RFC2308 で標準化されている。

キャッシュは非常に有用なものであるが、永久に残しておくわけにはいかない。実際のドメイン名情報が更新されても、ネームサーバは古い情報をリゾルバに返してしまい、古い情報がそのまま使われてしまう可能性がある。そこでネームサーバでは、 RR にある生存時間 (TTL: Time To Live) を用いて、キャッシュの有効期限を定めている。　もし保存したキャッシュが生存時間を超えてしまったらそのキャッシュを破棄することによって、データベースの整合性を維持している。

2.7 メッセージの形式

2.7.1 パケットの形式

DNS パケットの内容を以下に示す。(図 2.4参照)

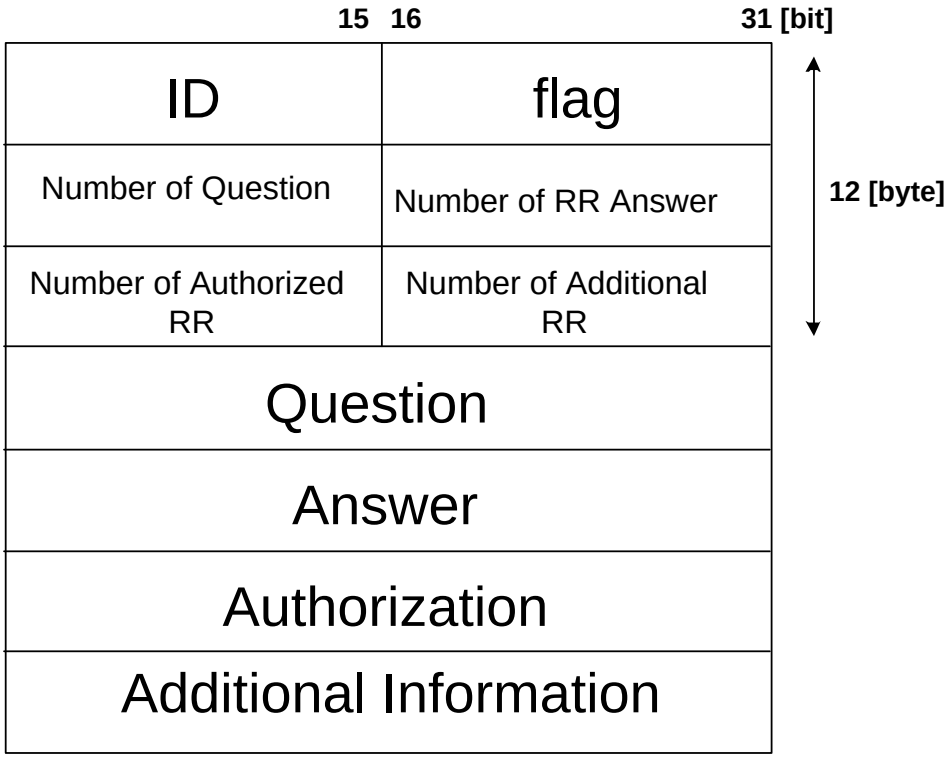

図 2.4: DNS パケットの内容

識別子 (ID) この数値で質問と回答の対応付けをする

質問数 (Number of Question) 質問部のエントリ数を示す符号無し 16bit 整数。

回答数 (Number of RR Answer) 回答部のエントリ数を示す符号無し 16bit 整数。

権威 RR 数 (Number of Authorized RR) 権威部のエントリ数を示す符号無し 16bit 整数。

追加 RR 数 (Number of additional RR) 付属部のエントリ数を示す符号無し 16bit 整数。

2.7.2 フラグの内容

DNS パケット内のフラグ (flag) 部分を以下に示す。

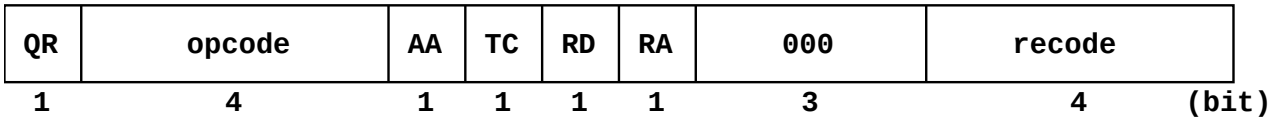

図 2.5: DNS パケット中の flag

QR： 1 ビットのフィールドで、 0 は照会であり、 1 は応答である事を意味している。

opcode： メッセージ中で問い合わせの種類を示す 4bit のフィールド。

0： 標準問い合わせ (query)

1： 逆問い合わせ (iquery)

2： サーバの状態の要求 (status)

3： 予約

4： ゾーン更新の通知 (notify)

5： 動的更新 (update)

6-15： 拡張のための予約

AA： 権威付きの答え (Authoritative Answer) 。
応答したネームサーバが、問い合わせたドメインについて権威があるかどうかを示す。

TC： UDP パケットの最大値である 512byte を越えてしまい、情報が欠落したことを示す。

RD： 再帰要求を意味する。ネームサーバに名前解決ができるまで、再帰的に問い合わせを繰り返すように要求していることを示す。照会の時にセットされる。

RA： 再帰要求の回答として再帰可能を示す。応答の時にセットされる。

recode： リターン・コードの 4bit のフィールド。

0： エラー無し

1： フォーマット不正

3： 照会されるドメイン名が存在しない

2.8 資源レコード (Resource Record)

DNS では、情報を (Resource Record) という形で保存している。

リゾルバが問い合わせる際に指定するものである。 DNS 資源レコードは 識別子 (ラベル)、クラス、型、値 (データ) からなる。これらがすべて同じであるような二つのレコードが存在することは無意味なので、サーバはもし重複に出会ったら、重複をとり除いた返事をすべきである。ただし、値だけが異っていて、識別子、クラスと型が同じレコードはほとんどのレコード型について認められている。このように、識別子、クラス、型が同じレコードの一群を資源レコード集合 (Resource Record Set, RRSet) と定義する。

A： IP アドレスを定義する。 32 ビット・バイナリ値として格納される。
　このレコードが最も一般的に使われる。

PTR： ポインタの照会に用いられる。逆引きの際に指定される。

CNAME： 「基準名 (canonicalname) 」の略である。これは別名のことである。

MX： メール交換レコード。
　インターネットに接続されているホスト上のメールの送り手によって用いられる。

NS： ネームサーバレコード。
　ドメインの権威あるネームサーバを指定する。ドメイン名として示される。

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

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。
- ・更新間隔―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―ゾーン転送に失敗時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―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―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

2.9 DNS における UDP の制限

DNS の名前解決では、問い合わせおよび応答メッセージは、 DNS が利用する UDP の制限で 512byte までと決まっている。この 512byte という制限は、 IP パケットがフラグメンテーションを起こさないことが保証される最大サイズから設定されている。この制限がないと、サイズの大きな UDP パケットは IP パケットがフラグメンテーションを起こし、パケットがロスした場合の処理やパケットの到着順番が変わるなどの複雑な処理が必要となる。 DNS ではこのような複雑な処理を実装することはせず、必要に応じて UDP の代わりに TCP を利用することになっている。

これはリゾルバが問い合わせクエリを発行し、ヘッダのフラグフィールド (図 2.5参照) の TC ビットが設定されて返信されることで確認できる。 TCP はいわゆるセグメントにユーザ・データを分割するため、どのような大きさのデータでも複数のセグメントにし転送することが可能である。

このメッセージの大きさが制限されていることにより、当然やりとりする情報の量も制限を受ける。ルートサーバの数が全世界で 13 台というのは、実はこの制限によりやりとりできる NS レコードの数の上限が決まっているためである。また DNS サーバが IPv6 のレコードである AAAA を持つ場合においては、 IPv6 のアドレス長が 128bit と IPv4 に比べて 4 倍になっているので、このサイズ制限をさらに圧迫する。

現在、 DNSsec や IPv6 の名前解決などの、より多くの情報量が必要となるものに対しては、DNS の拡張プロトコルである DNS 拡張機能 (edns0) を使用すること[2]で、より大きなサイズの UDP パケットを扱うことが可能となる。世の中で使用されている DNS サーバソフトウェアがすべてこの edns0 に対応しているわけではないが、これが広まれば UDP のサイズ制限問題が緩和される。

DNS は基本的に UDP を用いるため、リゾルバもネームサーバも独自のタイムアウト及び再転送を行わなければならない。というのも、 UDP を用いる他のアプリケーション (TFTP, BOOTP, SNMP) の多くはほとんどがローカルエリアネットワーク (LAN) で実行されるが、DNS はそれとは異なるからだ。 DNS の問い合わせと回答はワイドエリアネットワーク (WAN) でやり取りされるので、一般的にパケットの損失率や往復時間の変動が大きい。そのため優れた再転送、タイムアウトのアルゴリズムが必要である。

[2]BIND9 では実装されている

第 3 章

実験の環境

3.1 測定の環境

本研究で測定に用いたネットワークトポロジは図 3.1のようになる。早稲田大学と外部とは回線速度 4Gbps で接続されている。図 3.1のように測定用の PC を設置し、そこを流れるパケットを収集した。

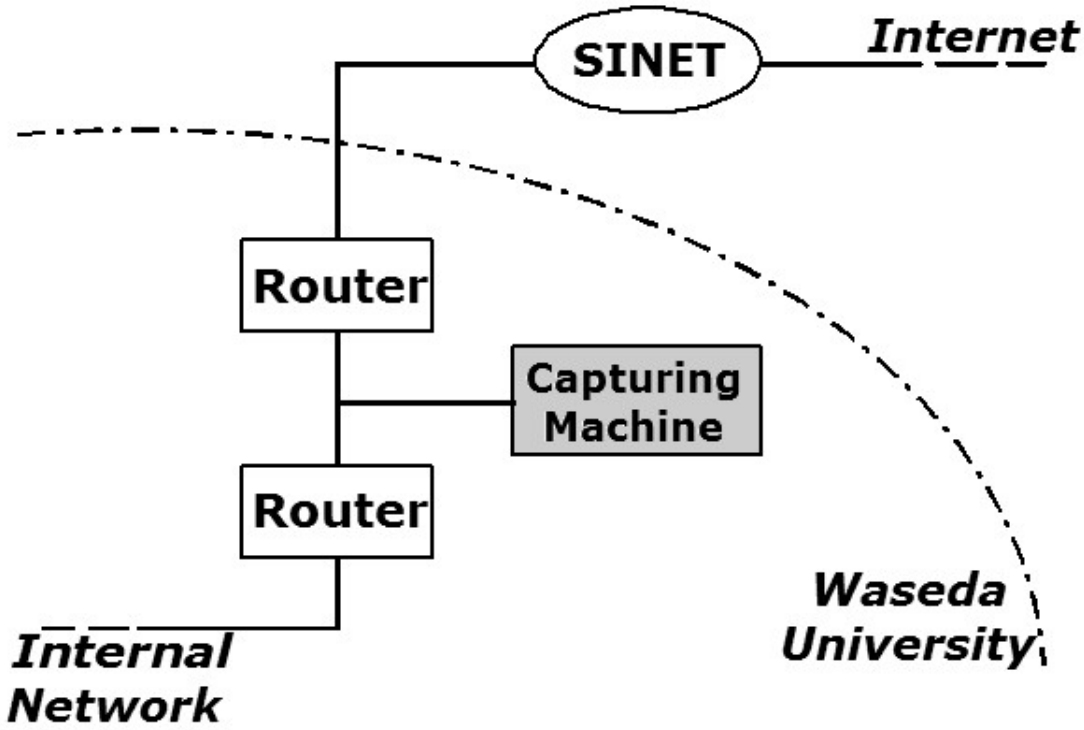

図 3.1: Network Topology

3.2 測定の対象

本研究で測定の対象としたのは以下の通りである。

- 大学内のホストから学外のネームサーバへの問い合せパケット
- 大学外のネームサーバから学内のホストへの回答パケット

これは大学に出入りするパケットの内、送信元または問い合わせ先のどちらかに port 53 を含むパケットということなる。測定用のプログラムとして tcpdump を用いて、パケットを取り込んだ。その出力を自作プログラムにより解析した。

3.3 測定マシン

本研究では、以下のマシンを用いて測定・解析を行った。

表 3.1: 測定環境

CPU	Pentium 4 2GHz
Memory	256MByte
OS	FreeBSD 4.10-RELEASE

第 4 章

実験の概要

実験の概要

DNS がその重要性とは裏腹に、実際の挙動はあまり研究されていないことは第 1章で述べた。
本研究ではその一端を知る目的で、実質的な名前解決をするリゾルバについて考察する。
学内から学外へ向かう経路で DNS 問い合わせメッセージと回答メッセージを採集し、
以下のような観点で調査した。

1. DNS 問い合わせに対して、回答メッセージが一定時間内に返って来ないものを抽出する
2. 送信元 IP アドレス, 問い合わせ先 IP アドレス
   送信元 port 番号, 問い合わせ先 port 番号がすべて一致する組を抽出する
3. その問い合わせた内容を参照し、照合する
4. その結果から問い合わせメッセージの再送と思われるものを特定する

送信元 IP アドレス、問い合わせ先 IP アドレスが一致するものなので resolv.conf に設定されているいくつかのネームサーバすべてをまとめて集計するのではなく、そのネームサーバ毎に集計をしている。つまり、一連の問い合わせの全過程を追うわけではなく、問い合わせ先のサーバ単位で考えるということである。

実験の観点

- 再問い合わせの時間間隔
- 再問い合わせ回数

これらに注目することによって、リゾルバの実際の動作状況を分析する。

第 5 章

実験の結果

5.1 再問い合わせの時間間隔

実験結果として、時間間隔を示す。結果は以下の 3 つに大別された。
出力形式を以下に記す。この形で代表的な 3 つのパターンを以下に示す。

> 送信元 IP アドレス.port 番号 > 問い合わせ先 IP アドレス.port 番号
> 再送の回数 ： 一番最初の問い合わせ (再送 0 回目) からの時間間隔（単位：秒）

1. `133.9.x.xxx.1025 > 198.zz.z.4.53:`

   `01 : 0.000028`

   `02 : 12.000123`

   `03 : 26.000463`

   `04 : 54.000947`

   `05 : 120.001445`

   `06 : 170.002256`

2. `133.9.xxx.xx.61756 > 193.zzz.zzz.149.53:`

   `01 : 0.000309`

   `02 : 0.000920`

   `03 : 0.005178`

   `04 : 0.011191`

   `05 : 0.017641`

3. 133.9.x.xxx.1025 > 198.zz.zz.12.53:
   01 : 63.991491
   02 : 179.983428
   03 : 308.975283
   04 : 498.968232
   05 : 748.961922

3 つの共通している点は大学内のホストから大学外へのネームサーバ (port 53) へ DNS 問い合わせメッセージを送信していることである。順番に見ていく。

1. BIND の仕様やデフォルトの設定と比較してもっとも理想的に近い形である。
   この例では、タイムアウト値は 4 秒、ネームサーバ数は 3 つと推測される。
   最初の問い合わせを 4 秒のタイムアウト値で送信する。そこでは回答が得られない。そして設定されている 2 番目、3 番目のネームサーバにそれぞれタイムアウト値 4 秒で問い合わせるが、それでも回答が得られない。そこで最初の問い合わせから約 12 秒経過する時に、設定された最初のネームサーバへ 1 回目の再送（問い合わせ）を行う。これはタイムアウト値 8 秒で設定され、若干の誤差はあるもののほぼ第 2、第 3 はそれぞれ 第 2 章の表 2.3 に従っている。

2. 今回の実験では、再送が 1 回以上確認された内の 4.5% がこのパターンに該当した。
   サーバ数やタイムアウト値で恐らく設定されてない結果が出ている。本来、1 つの通信が終わればしばらく使用されることのないエフェメラルポート[1]がこれほど使われるのはソフトウェアのバグの可能性が大きい。実際に、どのソフトウェアのどのバージョンという詳細までは今の段階では掴めていない。
   この結果のように 0 秒台での再送が非常に多く検出された。

3. 特異な例である。最初に問い合わせるネームサーバに対するタイムアウト値が大きすぎる。
   設定されているネームサーバ数や、タイムアウト値の予測は不可能だが、このように 1 回目の再送から数十秒のタイプは問い合わせ回数も 20 回を超える事が多くあった。
   明らかに異常である。再送回数については次節 5.2で詳しく見る。

[1]Ephemeral port (一時的な通信のために自由に利用できるポート)

5.2 再問い合わせ発生回数

ここで実験で得られたネームサーバへの再送（問い合わせ）の回数の結果を表 5.1で示す。

表 5.1 での割合は、デフォルト設定で正常とされる再送 3 回までの全体に占める割合である。3 回で収まらないものの内、多くはこれ以後にも再三に渡る多重問い合わせを行っており、ユーザの設定かネットワーク経路、ソフトウェアの実装に何らかの問題があると思われる。

考察

デフォルトでは第 2 章 表 2.3のように回数が決まっているがオプションとして問い合わせ試行回数を設定することが出来る。(最大 5 回) また、タイムアウト値も同様に変更する事が可能である。(最低 2 秒) さらに、 resolv.conf に設定されている第 2、第 3 のネームサーバに問い合わせを出さないようにすることもできる。今回の実験ではネームサーバ毎に注目しているためその問題は回避できている。試行回数とタイムアウト値については、様々な場合を想定している。

この結果から明らかになったように、再送（問い合わせ）の発生状況が本来の DNS の規格に従わないことがある。微少時間内に同一 IP アドレスからある特定のネームサーバへ多重に問い合わせている。それにより、回答率、再送率に変化が出ると思われる。一般的に DNS の回答率を考える場合、その問い合わせ先ネームサーバまでの経路や混雑状況に注目するが、これは問い合わせを出すローカル側の問題である。

5.3 結果のまとめ

- 再問い合わせの発生時間間隔は、本来の規則を無視しておりどちらかと言えば乱数に近い。
- 再問い合わせの送信回数は、 90% 以上が 3 回以内に収まっていた。

表 5.1: リゾルバの行った再送回数

時刻	2 回	3 回	割合	合計
0	33,678	1,892	95.288%	37,329
1	11,987	1,734	90.904%	15,094
2	14,786	1,989	91.258%	18,382
3	14,508	1,846	91.882%	17,799
4	12,844	1,486	91.152%	15,721
5	11,876	1,527	89.526%	14,971
6	12,946	1,607	90.831%	16,022
7	14,913	1,812	91.569%	18,265
8	17,378	2,210	91.010%	21,523
9	22,751	2,780	91.812%	27,808
10	24,966	2,980	91.127%	30,667
11	28,611	3,167	91.693%	34,657
12	28,752	3,124	91.199%	34,952
13	29,301	3,198	88.221%	36,838
14	27,896	3,232	85.509%	36,403
15	21,820	2,599	90.327%	27,034
16	34,671	4,163	88.963%	43,652
17	25,175	2,989	90.142%	31,244
18	21,741	2,752	91.028%	26,907
19	20,737	2,445	91.368%	25,372
20	18,830	2,286	91.933%	22,969
21	17,558	2,365	91.731%	21,719
22	16,422	2,130	91.529%	20,269
23	16,724	2,083	91.554%	20,542

第 6 章

まとめ

6.1 結果のまとめ

- 再問い合わせの発生時間間隔は、本来の規則を無視しておりどちらかと言えば乱数に近い。
- 再問い合わせの送信回数は、 90% 以上が 3 回以内に収まっていた。

6.2 今後の課題

今後の課題としては、微少時間内に同一 IP アドレス（port 番号）からの多重問い合わせの原因解明が挙げられる。 0 秒台での再送 (問い合わせ) が多かった。これは回答率、再送回数に大きく関係する。
これを解決するために考えられるのは

- resolv.conf の調整
  例えば、設定されているネームサーバが応答しない場合に第 2、第 3 のネームサーバとその優先順位を自動的に切り替え、次回以降での問い合わせに最適化する。
  これにより、問い合わせの効率化が図れる。
- ソフトウェアでの規制
  問い合わせを出すリゾルバ側ではなく、問い合わせを受け付けるネームサーバ側での工夫。パケットが複製されたかのような間隔での問い合わせは異常であると判断し、最初の問い合わせ以外は応じないなどの調整が必要かも知れない。
- BIND8.4.3 の使用禁止
  これを使用している DNS キャッシュサーバに IPv4, IPv6 アドレスの両方が割り当てられていて、かつ名前解決が正しくできないドメイン名を、その DNS キャッシュサーバ経由

で検索すると無駄な問い合わせパケットを発生し続ける。

これはパケットを調査するだけではバージョンまでは把握できない。

今回の実験では、 resolv.conf に設定されているある 1 つのネームサーバ単位で再送を考察したが、問い合わせ内容と時刻データに則し、第 2、第 3 のネームサーバへの問い合わせ、そしてその挙動までの一連の様子を完全に追跡したい。そしてネームサーバまでの経路や、問い合わせる時間帯によって最適な再送回数やタイムアウト値などを導きだし、現状よりさらに良い DNS 環境が構築の助けになれば幸いである。

謝辞

本修士論文の作成にあたり日頃より様々な御指導を頂いている早稲田大学理工学部情報・ネットワーク工学科の後藤滋樹教授に深く感謝致します。そして貴重な助言、アドバイスをくださった後藤研究室の諸氏、また、トラフィックの収集をするに当たって多大なご協力を賜った早稲田大学メディアネットワークセンターの方々に感謝致します。

参考文献

[1] W.Richard Stevens 著, 橘 康雄 訳, 井上 尚司 監訳: 『詳解 TCP/IP VoL.1』, ピアソン・エジュケーション, 2000.

[2] W.Richard Stevens 著, 橘 康雄 訳, 井上 尚司 監訳: 『詳解 TCP/IP VoL.2』, ピアソン・エジュケーション, 2000.

[3] Paul Albitz ,Cricket Liu, 「DNS & BIND 第 4 版」, オライリー・ジャパン, 2002.

[4] Philip Miller 著, 苅田幸雄訳『マスタリング TCP/IP 応用編』オーム社, 1998.

[5] 荒井祐一, 「インターネットにおける DNS のトラフィック解析」 早稲田大学理工学部情報学科 2001 年度卒業論文, 2002.

[6] 田中政史, 「DNS におけるキャッシュヒット率の解析」 早稲田大学理工学部情報学科 2002 年度卒業論文, 2003.

[7] 竹谷賢二, 「階層キャッシングによる効率的な名前解決」 早稲田大学理工学部情報学科 2003 年度卒業論文, 2003.

[8] RFC1034, "DOMAIN NAMES - CONCEPTS AND FACILITIES" `http://ietfreport.isoc.org/rfc/rfc1034.txt`

[9] RFC1035, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION" `http://ietfreport.isoc.org/rfc/rfc1034.txt`

[10] RFC2181, "Clarifications to the DNS Specification" `http://www.ietf.org/rfc/rfc2671.txt?number=2181`

[11] RFC2308, "Negative Caching of DNS Queries (DNS NCACHE)" `http://www.ietf.org/rfc/rfc2671.txt?number=2308`

[12] RFC2671, "Extension Mechanisms for DNS (EDNS0)" `http://www.ietf.org/rfc/rfc2671.txt?number=2671`

[13] JPRS, ”DNS 関連技術情報 ”
http://jprs.jp/tech/

[14] Internet Systems Consortium (ISC)
http://www.isc.org/

[15] 総務省,”情報通信白書平成 16 年度版 ”
http://www.johotsusintokei.soumu.go.jp/whitepaper/ja/cover/index.htm

[16] FreeBSD.org, ”FreeBSD Hyper Text Man Pages”
http://www.freebsd.org/cgi/man.cgi