修士論文 1999年度(平成11年度)

題目

ドメインネームシステムの耐障害性の向上のための研究

慶應義塾大学 大学院
政策・メディア研究科

土井 裕介 (89831921)

修士論文要旨 1999 年度 (平成 11 年度)

ドメインネームシステムの
耐障害性の向上のための研究

論文要旨

インターネット上で用いられている名前解決システムは、インターネット中に分散した名前サーバ群を利用する。しかし、それらの名前サーバは、必ずしも利用できるとは限らない。従って、目的ホストと送信者ホストとの間のネットワークが健全な場合でも、名前解決が失敗し、通信ができない場合がある。

本研究では、上記の問題をより明確に評価するための指標を作成し、原因を分析した。また、指標を用いて、問題が発生しやすい事例を発見した上で、それらの事例に対して有効な問題解決法である NSR3 を提案した。NSR3 では、名前サーバの位置情報を廃棄せずに記録しておき、名前サーバの障害に備える。さらに、NSR3 の一実装として、dnscap を作成し、実働実験を行った。平常時と障害時における実験結果の比較によって、障害時は平常時の 1.27 倍の効果を発揮する事が判明した。この結果は、dnscap が障害回避のための「代理応答」を、周囲の障害に適応して行っている事を示しており、これにより、NSR3 が有効に問題を解決する事を確認した。

キーワード: (1) インターネット (2) DNS (3) 名前キャッシュ (4) 耐障害性 (5) NSR3

慶應義塾大学 大学院 政策・メディア研究科
土井裕介 (89831921)

Abstract of Master's Thesis Academic Year 1999

A Research on the Improvement of the Robustness of the Domain Name System

Summary

The Domain Name System(DNS) is a name resolution system widely used in the Internet. To resolve each name-to-address mapping, DNS makes use of several sets of name servers dispersed throughout the Internet. Because these distant name servers are not always perfectly reliable, communications between two hosts which are connected by a trouble free network may fail due to name resolution failure.

In this research, the author analyzed the problem by defining a metric to evaluate it. As a solution to the problem, the author proposes the use of NSR3 along with an analysis of some outstanding cases in which it is most effective. An implementation of NSR3(dnscap) was tested and the results showed it to be 27% more effective during name lookup failure condition as when compared to normal operations. This shows that the dnscap makes "proxy answers" for failure avoidance, adapting to actual failure of the network/nameservice. Thus, the effect of NSR3 to solve the problem is proved by the result.

Keywords: (1) the Internet, (2) DNS, (3) name cache, (4) robustness, (5) NSR3

Keio University
Graduate School of Media and Governance
Yusuke DOI (89831921)

目 次

図目次

表 目 次

第1章　序論

本章では、研究の目的と意義を明確にした上で、本研究の進行について、論文の構成と共に述べる。

1.1　研究の目的

本研究の目的は、インターネットの名前解決システム (DNS: Domain Name System) の耐障害性の向上である。

利用者から見て通信ができない場合、利用者と接続相手との間のネットワークに障害がある場合が多い。しかし、両者間のネットワークが健全であっても、名前解決システムが正常に働かないという理由で、通信ができない場合がある。

名前解決システムは、インターネット上に分散している複数の名前サーバの集合である。ネットワークの障害によって、必要な名前サーバが利用者から利用不可能になる場合がある。多くの場合、ネットワークの障害は短時間で復旧する。しかし、DNS は構造上、広い範囲の障害の影響を受けやすく、一つひとつの障害が小さくても、大きな障害となる事がある。

そこで本研究では、遠隔の名前サーバの利用に障害が起きた場合、利用者からその障害を隠蔽し、あたかも障害を持つ名前サーバに到達できているように見せる事で、この問題を解決する。本研究により、DNS の構造に起因する問題を解決し、利用者の立場から見た「インターネットの接続性」を向上させる。

1.2　本研究で解決する問題

名前解決に利用するネットワークで障害が発生した場合に、名前解決に失敗することがある。この場合、送受信者間の通信に使われるネットワークが正常な場合でも、通信は成立しない。この問題を、DNS の設計の基本的な誤りである、「必要な名前サーバは、常に通信可能かつ利用可能である」という仮定を取り除く事で解決する。問題の解決については節 4.1 で述べる。

1.3　本論文の構成

本研究では、DNS が原因となる通信の失敗を問題と定義して、これを解決する。本論文の構成を以下に示す。

問題を明確にするための準備として、第 2 章では、例を通じて問題を提起し、その問題の背景にどのような理由があるかを示す。その上で、第 3 章にて問題解決の準備として名前解決の構造的分析を行う。同時に、問題がどの程度発生しており、潜在的にどの程度発生する可能性があるかについて述べる。ここまでで、問題の本質が DNS の再帰問い合わせの構造にある事を明らかにする。

問題を解決するためのアプローチとして、第 4 章では、いくつかの仮定を前提として、問題を解

決する手法 NSR3(Name Service Resource Record Recycle) を提案する。同時に、その仮定の検証を行い、DNS の技術的制約から NSR3 が前提とする仮定が成立する事を示す。

第 5 章では、NSR3 を現在用いられているインターネットアーキテクチャで実現するためにいくつかの設計を比較検討する。NSR3 の実装を作成して、実働試験を行った結果を、第 6 章で評価して、今回の研究により問題が解決できる事を示す。

いくつかの検討課題について、第 7 章にて考察を行い、最後に第 8 章にて、本研究の総括を行う。

1.4 関連研究

IETF DNSOPS 分科会 [23] は、IETF における DNS の運用を議論する分科会である。DNSOPS 分科会から、特にルートネームサーバに関する冗長性の確保のための運用案がいくつか提案・議論されている。

東京工業大学の太田昌孝氏、および Equinix, Inc. の Ted Hardie 氏によって提案されている方式 [25] は、ルートネームサーバの IP アドレスを共有する事で、技術的に 13 に制限されているルートネームサーバの数を増やし、数による信頼性を確保しようという提案である。この手法は、現在の技術を全く拡張する事無く、ルートネームサーバの数を増やす事ができる。しかし、ルートネームサーバの選択を経路制御機構に委ねる事によるシステムの複雑性の問題、管理主体のあいまいさによる障害時の連絡先 (責任者) の問題、中央集権型の解決策であり、ルートネームサーバに対してのみしか適用できないといった問題がある。

また、Sprint の C. Catalone 氏、R. Rockwell 氏によって提案された方式 [24] は、上記の方法と同様に共有された IP アドレスと経路情報を用い、複数の名前サーバに同じ IP アドレスを共有させる。この方法は各ネットワーク毎に自律分散的に適用できるが、上記の方式と同様に、経路制御に名前サーバの選択を委ねるので、複雑性の問題が残る。

特に、上記の技術は両方とも特定のドメイン名に対する信頼性を個別に調整するものであり、ドメイン名全体に効果が及ぶ技術ではない。

1.5 観測データに関して

本論文では、特に断りの無い限り、観測データは 1999 年 12 月時点のものである。

第2章　背景

本章では、名前解決が原因の問題が発生する具体例を挙げ、問題の原因について考察する。特にDNSが広域のネットワークに依存する構造を「DNS広域依存問題」と定義し、この構造が名前解決に関連する障害の原因である事を示す。

なお、DNSに関して、本研究と密接に関係する概念は付録Aにまとめた。

2.1　問題が発生する具体例: FreeBSD.org.

具体例として、`www.jp.freebsd.org.`を挙げる。

1999年12月現在、`www.jp.freebsd.org.`は日本国内ネットワークの中心部のひとつ(NSPIXP2)から、2hopという位置にある。ところで、ドメイン名を解決するには、ドメイン名空間を根から辿る必要がある。この例に対応するドメイン名空間と名前サーバの対応表を、表2.1に挙げる。表2.1から、`www.jp.freebsd.org.`という名前を解決するためには、いくつもの名前サーバを利用する事が判る。それら名前サーバの大多数は、米国に存在する。

表2.1: www.jp.freebsd.org.に関係するゾーンと権限を持つ名前サーバ

ゾーン	名前サーバ(群)	位置
.(root)	13個のルートサーバ群	米国(10) 日本(1) 欧州(2)
org.	12個のGTLDサーバ群	米国(10) 欧州(2)
freebsd.org.	ns1.crl.com.	米国西海岸: CRL
	ns2.crl.com.	米国西海岸: CRL
	ns1.iafrica.com.	米国東海岸: Teleglobe
	ns2.iafrica.com.	米国東海岸: Teleglobe
	ns1.root.com.	米国: ALTERNET
	who.cdrom.com.	米国西海岸: CRL
jp.freebsd.org.	ns.isoternet.org.	日本国内: NTT OCN内
	ns2.spnet.ne.jp.	日本国内: NTT OCN内
	hito.eccosys.com.	東京: NSPIXP2近傍(PSINET)
	monster.sfc.wide.ad.jp.	慶應大学SFC大学院棟
www.jp.freebsd.org.	(N/A)	東京: NSPIXP2近傍(ITJIT)

ここで、日本国内のネットワークから海外へのネットワークに到達できなくなったとする。この場合、`jp.freebsd.org.`の名前サーバの情報を名前解決の一時記憶(キャッシュ)中に持っていない限り、`www.jp.freebsd.org.`に到達できなくなる。

図2.4に、研究室のホスト(`adely.sfc.wide.ad.jp.` 203.178.140.37)から、表2.1にあるそれぞ

れの名前サーバ(ルートネームサーバは除く)、及び`www.jp.freebsd.org`までの経路[1]を図示した。送信者ホストは、図中左上`203.178.140.33`である。また、受信者ホスト`www.jp.freebsd.org`は、図中上側に位置する`203.139.121.132`である。全ての名前サーバと、送信者から受信者までの経路は色を変えて表示している。

図を見ると、`203.178.141.57`という中継ルータから、利用するネットワークがはっきり2つに分岐している。この分岐は、SFCから直接WIDE Los Angeles NOCへ渡る海外線と、国内のNSPIXP網との分岐である。この図から、近隣のホストのドメイン名にも関わらず、国際ネットワークを利用しないと名前が解決できない事が判る。つまり、仮に海外のネットワークが利用できなかった場合、`www.jp.freebsd.org.`との間のネットワークが正常だったとしても、名前解決に失敗してしまう(名前解決障害)。その結果、目的ホストとの間のネットワークが正常にも関わらず通信ができない。

2.2 名前解決への依存

現在のインターネットの利用モデルでは、通信の終点を指定する方法はほとんどの場合「ドメイン名」である。ドメイン名は記憶しやすい上、ネットワークの変更によるIPアドレスの変更があっても、DNSのデータベースを正しく更新すれば、同じドメイン名を識別子として使い続けられるという長所がある。

従って、利用者によるほぼ全てのコミュニケーションは名前解決に依存している。telnet等でホストを指定する際のみならず、URL/URI[27]やe-mailアドレス[26]などでも利用されている。

この事実は、名前解決の障害はコミュニケーションの障害の原因となる事を意味する。そして、節2.1で挙げた例のように、広域ネットワークを利用して名前解決を行わなければならない場合がある。従って、名前解決の障害は、コミュニケーションの信頼性を損う重大な問題である。

2.3 ドメイン名利用の国際化傾向

従来は、ドメイン名と地理位置の関係は強かった。`com.`や`org.`といったドメインの中は殆どが米国組織で、米国以外の組織はccTLD(国別トップレベルドメイン)内に存在していた。しかし、IAHC[21]によるgTLD-MOU[20]でも述べらているように、現在は`com.`等のドメインはgTLD (generic Top Level Domain: 地域や性質に依存しない一般的なドメイン名)としての役割を与えられている。また、トンガ等インターネット利用の後発国では、自国のccTLDを資源として広く世界中に開放し、gTLDに近い形態での運用を行っている。また、「ドメインネームの全て」[11]によると、日本のgTLD利用率は30%弱だが、全世界では75%程度の組織がgTLDを利用しており、世界的なgTLDへの偏向が存在する。

このような背景により、ドメイン名と、そのドメイン名を利用する組織のネットワーク上での位置との関係が薄くなっている。必然的に組織位置と名前サーバの位置との関係も薄くなり、特に重要なトップレベルドメインの名前サーバが遠く離れたネットワークに存在する可能性が高まる。また、国際的企業や組織の強い「ブランド力」を持つドメイン名についても同様の構造が存在する。国際的に共通のイメージを作り出す事がブランド力に対してプラスに働く場合、国別のドメイン名をccTLDではなくブランドドメインの下に置く事がある。

それぞれのドメイン名に関して具体的な例を挙げる。トップレベルドメインの国際化に関しての

---

[1] 1999年12月17日午後6時20分頃の経路の往路のみ

例は、個人ドメインやウェブホスティングサービス等に見られる。`www.itojun.org.`[18] は gTLD 中に存在する個人的なドメインであり、また `www.mizu.nu.` は Niue という国に与えられた ccTLD 内に存在するが、日本で活動する一個人 [19] が利用しているドメイン名の一例である。このような例は枚挙に暇が無い。国際ブランド力を持つ企業ドメイン名には、日本ヒューレット・パッカード (`jpn.hp.com.`)、日本 IBM (`jp.ibm.com.`) 等がある。またこれは企業ではないが、節 2.1 で例に挙げた FreeBSD(`jp.freebsd.org.`) 等の共同体も存在する。

これらのドメインは必ずしも国際的という枠組の中に存在するわけではない。これらのドメインの共通点は、名前サーバと実際のホスト・サイトのネットワーク位置が離れている事がある、という点である。

2.4 「DNS 広域依存問題」

本節では、現状の名前解決系が持つ問題を分析し、なぜ名前解決の障害が発生するのか、その原因を明確にする。

2.4.1 問題の一般化

DNS は、広い領域の障害に影響される。その理由は、遠隔の名前サーバが利用できない状況が、上位から下位への参照で成立している名前空間木 (図 A.1) を切断する原因になるからである。もしも、ある節を管理する名前サーバが利用できなくなったとする。すると、その節から末端方向の名前空間木が、全て利用不可能になる。逆に言えば、あるドメイン名の解決は、そのドメイン名の上位の節が一つでも利用不可能ならば、失敗する。

名前空間木の切断の例を、図 2.1 に示す。この図は、`jp.` ドメインの名前サーバが利用できなくなった場合を示している。仮に、使用しているリゾルバサーバから `jp.` ドメインの名前サーバが一つも利用できない状況が発生したとすると、近傍のネットワークは利用できるにも関わらず、`jp.` ドメインの名前が全て解決できなくなってしまう。

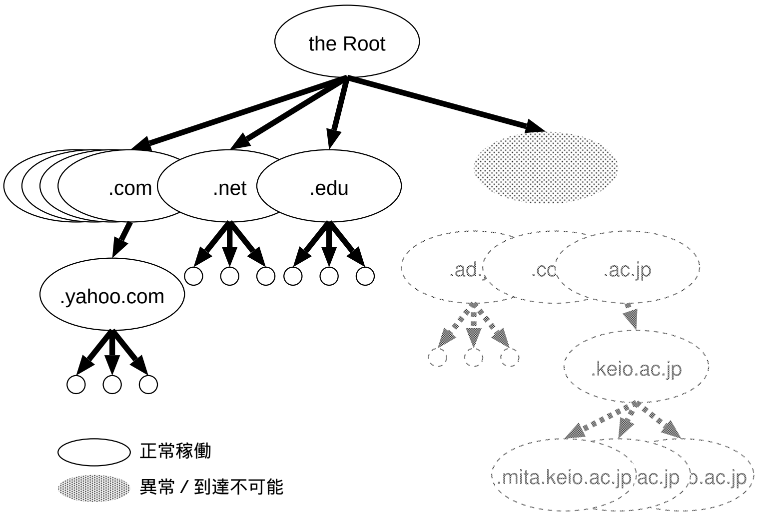

図 2.1: 名前空間木の切断

名前空間木の切断による障害の顕著な例が、節 2.1 で述べた `freebsd.org.` の例である。この場合、解決するドメイン名は `www.jp.freebsd.org.` であり、その名前を持つホストは地域ネットワーク、即ち日本国内に存在する。しかし、日本国外への到達性が失われた状態では、日本国内のホストと言えど名前が解決できず、実質的に通信が確立できない。

また、DNS に遠隔のネットワーク障害が影響を与えやすくする原因の一つに、節 2.3 で述べたようなドメイン名の国際化がある。国際化されたドメイン名においては、ネットワークの実体と名前サーバの位置が離れている事例が多い。従って、近隣の名前を解決するにも広範囲の名前サーバを利用しなければならない場合が増えている。その結果、通信障害の原因のうち、名前解決の障害の割合が増加している。

2.4.2 再帰問い合わせの広域ネットワークへの依存

名前解決障害は、再帰問い合わせの段階で名前空間木を下方向に辿る途中、名前空間が切断されている事から発生する。切断された名前空間において名前解決を失敗してしまう理由は、「再帰的名前解決」の枠組の不備にあり、名前解決障害が頻発する理由は、DNS が広域のネットワークへの依存にある。

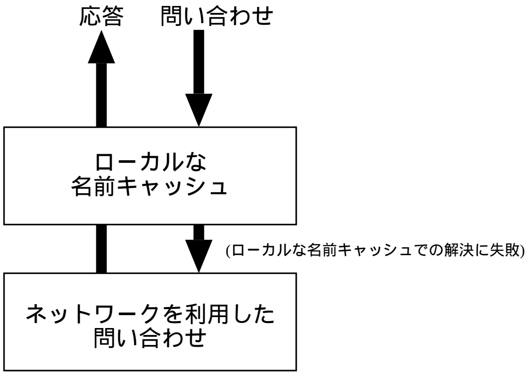

図 2.2: DNS の問い合わせの枠組

図 2.2 は、リゾルバサーバの問い合わせの枠組である。リゾルバサーバは、必要な情報を取得するために、まずキャッシュを参照する。そして、もしキャッシュからの情報取得に失敗した場合のみ、ネットワークを利用した問い合わせ (再帰的名前解決) を行う。

ここで、図 2.2 を見ると、遠隔の名前サーバへの問い合わせが失敗した場合に、取り得る代替手段が無い。これでは、名前空間木における節、つまり目的ドメイン名に対して上位のドメイン名の解決に失敗した場合、当然、目的ドメイン名を解決できない。

一方、上位のドメイン名を管理する名前サーバと下位の名前サーバ、そして目的ホストが同じような位置にあれば、そもそもこのような問題は発生しない。なぜならば、結局目的ホストとの通信に使うネットワークが健全ならば、名前解決も成功するし、そうでなければそもそも目的ホストへ到達できない。しかし、ここまでに述べたように、名前サーバの分布は必ずしも目的ホスト近傍とは限らない。

ここで、通信送信者と受信者との間のコミュニケーションに必要な通信領域を、「コミュニケーション領域」と定義する。同様に、名前解決に必要となる通信領域を「名前解決領域」と定義する。

2 つの領域を単純に図示すると、図 2.3 の様に描ける。この図の右側の円をコミュニケーション領域、左側の円を名前解決領域とする。

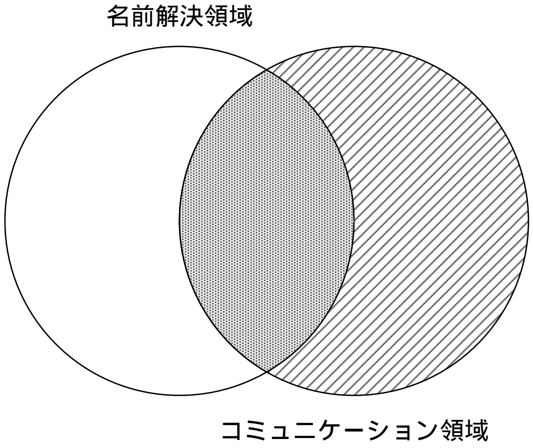

図 2.3: 2 つの領域 (模式図)

尚、本来はコミュニケーション領域 (右側の円) のみで行う通信を始めるためには、名前解決を成功させる必要がある。そして、名前解決は、広い領域に分散した名前サーバを用いて行う。その結果、本来右側の円で示される領域の内部で完結すべき通信を始めるために、広く分散した名前解決領域 (左側の円) の健全性が必要になる。DNS が広い領域に分散した名前サーバとネットワークに依存している点から、この問題を「**DNS 広域依存問題**」と呼ぶ。

正常な通信に影響を与える障害は、図中にランダムに発生する。障害がどちらかの円の中に発生した場合、通信が不可能になる。ここで、障害が右側の円の内部に発生した場合は、送受信者間の通信が正常に行えなくなる。逆に、障害が左側の円の内部に発生した場合は、名前解決が正常に行われなくなる。当然、両方の円の重なる部分に障害が発生した場合は、名前解決も本来の通信も正常に行えないものとする。

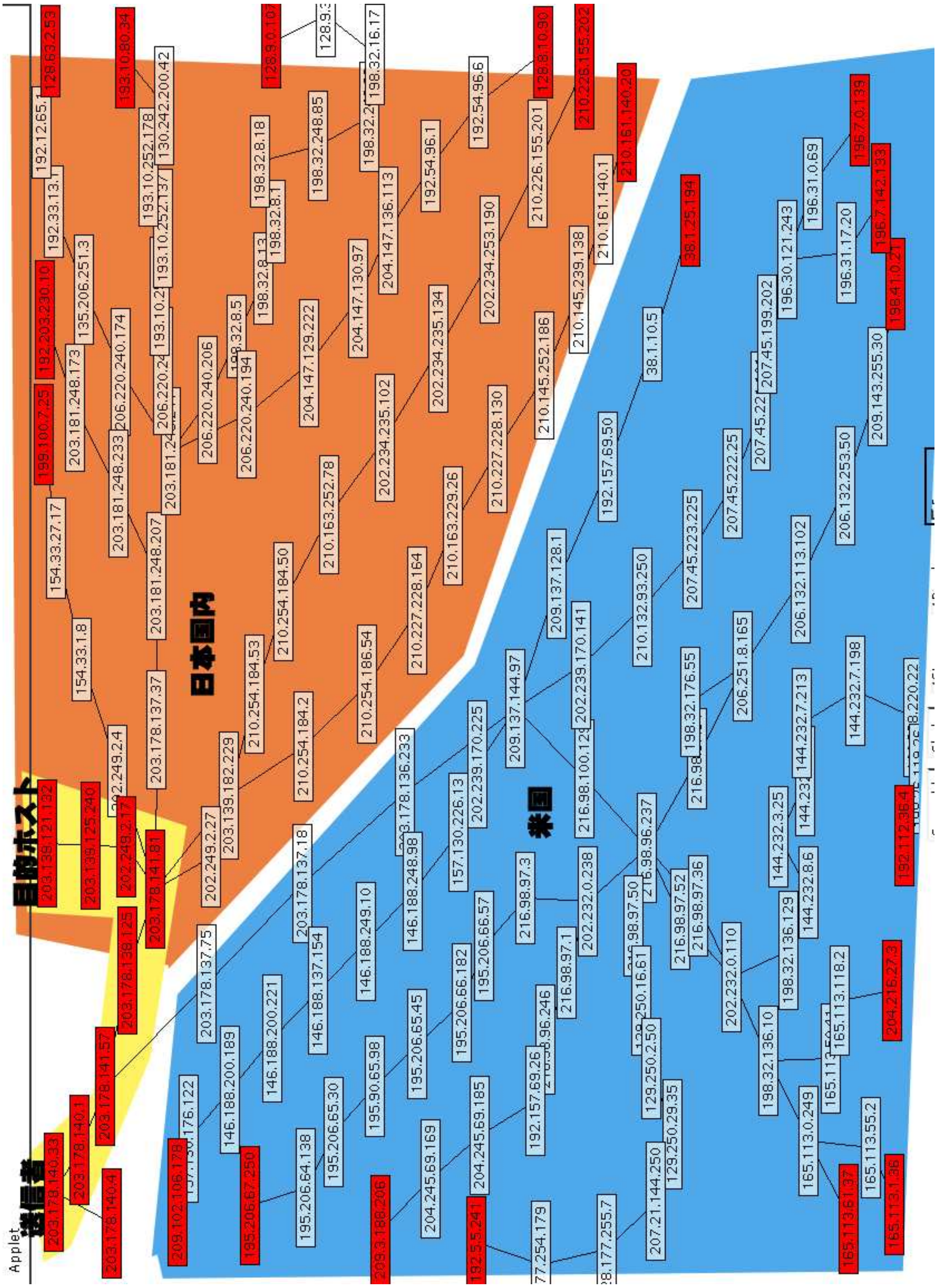

図 2.4: www.jp.freebsd.org. に関連する経路

第3章　問題の分析

本章では、DNS 広域依存問題を分析する。そのために、名前解決系の信頼性と、全体の信頼性の評価を行う指標を作成し、その指標を利用して実環境を観測、調査する。

3.1　指標化による問題の評価

DNS 広域依存問題の存在をより明確にするために、問題の程度を指標化する。指標化により、問題をおこしやすいケースを判別しやすくなり、また、問題の程度の比較が容易になる。DNS 広域依存問題を指標化するためには、コミュニケーション領域と名前解決領域との間の関係を指標化すれば良い。

必要な通信領域の和のうち、名前解決にのみ利用されている領域の大きさを、DNS 広域依存問題の指標とする。つまり、この問題の指標化は、図 2.3 の円の中の白抜きの部分の全体に対する比を求める問題に似ている。

3.1.1　モデル: 名前解決系

名前解決にのみ関係する機構を、一つの「名前解決系」と定義する。「名前解決系」は、与えられたドメイン名、そのドメイン名の解決に利用する可能性がある名前サーバの組み合わせ、そして問い合わせ開始地点 (ネットワーク位置) の組み合わせ毎に定義できる。

ドメイン名は、全世界で共有されている。しかし、名前解決を行うリゾルバサーバ、及び解決すべきドメイン名によって、ネットワークのどの部分を利用するかは変化する。従って、どの名前サーバを利用するべきなのかがドメイン名によって決定される。一方、それぞれの名前サーバへのネットワーク的相対関係は、検索の開始地点 (リゾルバサーバ) よって決定される。解決するドメイン名と、解決を行うリゾルバサーバの組み合わせ、一つ一つを本論文では「名前解決系」と定義し、評価の対象とする。

問題を指標化するために、名前解決系のモデルを定義する。名前解決系モデルの構成要素を表 3.1 に示す。

ここで、簡便のため、$src = rs$ と仮定する。一般に送信者ホストとリゾルバサーバは同一であるとは言えない。しかし、多くの場合はホストは近傍のリゾルバサーバを選択する。またリゾルバサーバと送信者ホスト間の通信が失敗する場合は、障害の原因は明白であり、評価の対象としない。

また、インターネットの経路は非対称であり、一般に $path(src, dst) \neq path(dst, src)$ である。ところで、指標化の目標は「問題を起こしやすいケースの判別」であり、往路と復路を別個に考察しても目的を達成できる。また、往路について考慮した結果は復路に関する考察にも活用できるので、本研究では往路についてのみ考察する。

表 3.1: 名前解決系モデルの構成要素

名前	記号	役割
送信者ホスト	src	通信の送信者
受信者ホスト	dst	通信の受信者
ドメイン名	d	検索されるドメイン名
リゾルバサーバ	rs	d を検索するリゾルバサーバ
名前サーバ	$NS(l, n)$	d に権限を持つ上位から深さ l の n 個目のサーバ
経路	$path(A, B)$	ホスト A からホスト B へ至る経路および経路上のルータ群

3.1.2 名前解決系の構造評価指標

問題の評価のために、名前解決系のネットワーク構造から、その名前解決系における『DNS広域依存問題』の発生のしやすさを指標化する。指標にはいくつかの形式が考えられる。中でも、本研究では、単純な確率論を適用した指標が最もDNSの特性を評価しやすい。指標化の段階で無視できない要素が、セカンダリ及びキャッシュの存在である。だが、本研究で提案する指標はこれら全てを網羅するものではない。

指標 a: 要素数の比較による問題の程度の概算

本研究では、障害の原因の中でどの程度名前解決障害が占めるかを示す指標として、以下の式で評価される指標 M_a を定義する。

$$M_a = \frac{|path_{ns}| - |path_{common}|}{|path_{ns}| + |path_d| - |path_{common}|} \tag{3.1}$$

ここで、$path_{ns}$ は名前サーバに関係する要素数、$path_d$ は送信者受信者間での通信に関係する要素数、$path_{common} = path_{ns} \cap path_d$ である。また、$|path_x|$ とは $path_x$ の要素数を表す。

指標 M_a では、単純にシステム要素の数が増えると障害の危険性が増えると仮定し、要素の数をそのまま指標に適用した。ここで、$path_{ns}$ は図2.3の左側の円、$path_d$ は同図の右側の円、$path_{common}$ は同図の円が重なっている部分に相当する。指標 M_a によって、図2.3の円全体の面積中の白抜き部分の割合が求められる。これは指標化の方針と合致し、問題の発生のしやすさに対する適切な近似を与える事が期待できる。

しかし、指標 M_a での『要素数』の定義には問題がある。仮にここで『要素数 = ホップ数』とする。つまり、$|path_d|$ は受信ホストまでのホップ数となる。ホップ数はルータの数であり、またルータ間の接続の数であり、さらに関係する設定の数でもあるので、要素数の代表としてホップ数を選択する事自体は適切である。

しかし、名前サーバに関して、$|path_{ns}|$ をどう数え上げるかという問題がある。一つのゾーンには複数の名前サーバが関わっている。複数の名前サーバはそのゾーンの信頼性を高めるが、その信頼性の向上は $|path_{ns}|$ の値に対応しない。近似として、それぞれのレベルのゾーンに対して、最も近い名前サーバへのホップ数の総和を $|path_{ns}|$ として採用する事ができる。

指標 **b:** 名前解決の信頼性評価

指標M_bは、セカンダリを含めた、名前解決のみの信頼性の指標である。従って、指標M_bは次に述べる指標M_cの準備であり、図2.3の左側の円の大きさを求める事に等しい。

M_bでは、冗長な名前サーバによる信頼性の向上を指標化するために、確率論を利用する。ここで、M_aと同様に「ホップ数」を用いて障害可能性の計算を行う。但しM_bでは、「1ホップあたりの信頼性」を定数として仮定し、確率論の手法を適用する。この仮定は、名前解決系の障害確率の導出が目的ならば適切でない。しかし、M_bの目的は指標の相対評価であり、実際の障害確率を推定する必要は無いので、1ホップあたりの信頼性を定数として適用する事で十分目的を達成できる。

M_bは以下の式で与えられる。ここで、

$$\Pi_{i=0}^{n}\{x_i\} = x_0 \cdot x_1 \cdot x_2 \cdot \cdots \cdot x_{n-1} \cdot x_n$$

とする。

$$\begin{align}
M_b &= P_s(\text{与えられたドメイン}\ d\ \text{に対する名前解決}) \tag{3.2}\\
&= \Pi_{l=0}^{l=|d|-1}\{P_s(d\ \text{のそれぞれの}\ l\ \text{の名前解決})\} \tag{3.3}\\
&= \Pi_{l=0}^{l=|d|-1}\{1-\Pi_{ns=1}^{|NS(l)|}P_f(d\ \text{のある}\ l\ \text{のそれぞれの名前サーバの利用})\} \tag{3.4}\\
&= \Pi_{l=0}^{l=|d|-1}[1-\Pi_{ns=1}^{|NS(l)|}\{1-P_s(NS(l,ns)\ \text{への問い合わせ})\}] \notag\\
&= \Pi_{l=0}^{l=|d|-1}[1-\Pi_{ns=1}^{|NS(l)|}\{1-R^{|path(rs,NS(l,ns))|}\}] \tag{3.5}
\end{align}$$

ここで、Rは定数として与えられた1ホップあたりの信頼性である。本論文では0.99を採用した。$P_f(X)$は事象Xの障害率/失敗可能性、$P_s(X)$は事象(X)の信頼性/成功可能性を表し、$P_f = 1.0 - P_s$の関係である。また、$|d|$でドメイン名dの深さ(名前空間木の根からの距離)、$|NS(l)|$でドメイン名dに権限を持つそのレベルの名前サーバの数を表すものとする。

式3.5は、与えられたドメインdへの名前解決の成功率を出発点(式3.2)とする。ある名前の解決が成功するためには、ドメイン名のうち、ホスト名以外の全ての階層($|d|-1$)の解決が成功しなければならないので、それらを積算(式3.3)する。それぞれの階層の名前解決が成功するためには、少なくとも一つの名前サーバによる名前解決が成功すれば良い。そのために、その階層の全ての名前サーバ($|NS(d)|$)での名前解決が失敗する確率を1から引く(式3.4)。それぞれの名前サーバへの問い合わせが成功する確率は、ここでは経路のホップ数に依存するとして、$R^{|path(rs,NS(l,ns))|}$、つまり定数Rを経路のホップ数乗したものにより求めた。

以上の結果をまとめた式3.5により、あるドメイン名dの名前解決が成功する確からしさを指標として求める事ができる。なお、式3.5は全ての深さのゾーンに対して解決を行う事を仮定しており、キャッシュの影響は考慮していない。

指標 **c: DNS** 広域依存問題の指標

M_cは、M_bをもとに導出する、「DNS広域依存問題」を表す指標である。まず、M_cでは「解決する目標となるドメイン名に対して権限を持つ名前サーバは、目標となる受信者ホストの近傍に一つある」という仮定を導入する(節4.2.4)。M_cによる指標化の段階では、上述の仮定があてはまらないいくつかの例外を念頭に置きつつ、一般的なケースに関して考察する。有名でかつ特殊な例に関しては、節3.2.2にて個別に解析する。

M_c は以下のように計算できる。なお、M_c は M_b を基礎にしているので、信頼性の指標である。問題を起こしやすい事例は 0 に近く、問題を起こしにくい事例は 1 に近くなる。

$$M_c = \Pi_{l=0}^{l=|d|-2}[1 - \Pi_{ns=1}^{|NS(l)|}\{1 - R^{|path(rs,NS(l,ns))|}\}] \cdot (1 - R^{|path(src,dst)|}) \quad (3.6)$$

式 3.6 は式 3.5 の拡張である。まず、M_c では M_b に比べて、コミュニケーション領域の存在を考慮している。受信者ホストとの通信の障害可能性 $(1 - R^{|path(src,dst)|})$ を掛ける事で、その領域のネットワークの信頼性が低ければ、M_c は M_b に比べて高い値になり、受信者との通信の信頼性が低ければ、M_c は M_b に比べて低くなる。また、M_c は M_b に比べて、名前解決の最終段階を評価しない。これは、その名前解決の経路は目的ホストとの通信の経路にほぼ等しい。つまり、図 2.3 中の、両方の円が重なる部分に相当する。従ってこの部分の信頼性は、DNS 広域依存問題の評価指標である M_c からは除外する。

指標 M_c により、それぞれの名前解決系に対する「DNS 広域依存問題」の発生しやすさを表す事ができる。

3.2 観測

本節では、実際の DNS トラフィックの測定や名前サーバ位置の調査等の、実環境の観察を通じ、実際にどの程度「DNS 広域依存問題」が発生していて、発生の可能性が高い場合がどの程度存在するのかを調べる。

3.2.1 名前解決系構造評価指標に基づく観測結果の分析

本節では、研究室で使われているリゾルバサーバ (`shonan.sfc.wide.ad.jp.`: 203.178.140.7) を観測地点として、その上で収集した実トラフィックを、節 3.1.2 で述べた指標に基づき調査する。

分析の目標は、DNS 広域依存問題が発生しやすい事例を実トラフィック中に発見する事である。いくつかの事例については、節 3.2.2 で個別に分析を行っている。

トラフィックの収集条件は以下の通り。

- 日時: 1999 年 12 月 11 日土曜日 14 時 05 分より 19 時 55 分
  活動の少ない土曜日を選んだ理由は、トラフィック収集による名前サーバの性能低下が他の活動に影響しないように、システムに余力を見込む必要があったためである。よって、システムの余力が確実に存在し、かつ、適当な量のネットワークの活動が認められる土曜日午後を収集のために選んだ。

- 場所: `shonan.sfc.wide.ad.jp.`(203.178.140.7) のネットワークインターフェース上
  `shonan.sfc.wide.ad.jp.` は、徳田村井楠本中村研究室で利用している `sfc.wide.ad.jp.` ドメインの名前サーバであり、研究室の多くのホストがこのサーバをリゾルバサーバとして利用している。

- 方法: `tcpdump`[32] による収集
  フィルタとして、`port 53 and udp` を与えた。また、パケット長は DNS の UDP メッセージの規格上の最大値である 512 バイトを指定した。これにより、`shonan.sfc.wide.ad.jp.` を利用する UDP の DNS メッセージ全てが記録される。

- 備考: 予め研究室内部にはパケットのモニタリングを行う可能性があることを伝えた上で、実験に協力する意思のある学生は `shonan.sfc.wide.ad.jp.` をリゾルバサーバとして利用するよう依頼してある。

上記の条件で収集したトラフィックに対し、以下の手順により指標 c を適用し、評価する。

1. DNS メッセージの QNAME フィールド (問い合わせドメイン) に現われるドメイン名を全て抽出
2. 抽出したドメイン名それぞれに対して、以下の処理を行う
   (a) ドメイン名に関係する名前サーバのリストを取得
   (b) それぞれの名前サーバについて、`traceroute` プログラムにより往路の経路を取得
   (c) ドメイン名が示すホスト (IP アドレス) が存在すれば、そのホストへの経路を同様に取得
   (d) 手順 2c に成功すれば、ドメイン名に対する名前解決系の指標 (M_c) の導出

手順 1 で抽出したドメイン名は 260967 個で、重複するドメイン名を除くと、一意なドメイン名は 3102 個あった。そのうち、手順 2c でドメイン名とホストとの対応付けが確認されたものは、1610 個であった。ホストとの対応付けが無いドメイン名のほとんどは、ホスト名以外への問い合わせ (MX/NS/PTR RR 等) か、ドメイン名の入力間違い、あるいはリゾルバライブラリが自動的に補完したドメイン名が誤っている事例である。

これらの 1610 個のドメインに対して、節 3.1.2 で述べた指標 c を適用する。この指標を 0 から 1 に向って 0.0001 刻みで累積したものを、図 3.1 に示す。

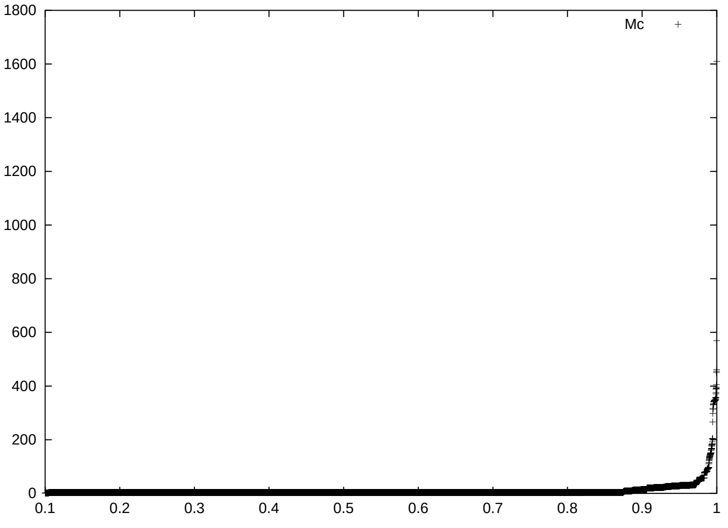

図 3.1: 1610 ドメインに対する指標 c の適用

図 3.1 は、横軸に指標 c を、縦軸に累積ドメイン数を示している。図 3.1 を、0.95 から 0.9999 まで拡大したものを図 3.2 に示す。

このグラフから読み取れるように、殆どのドメインに対する名前解決系の M_c は、1 に近い値を示す。1041 個のドメインの指標は、0.9999 を超える値を示した。一方、指標が 0.99 を超えるドメインは 128 個存在する。

指標が小さい値を示すドメインをおおまかに分類すると、以下のように分類できる。

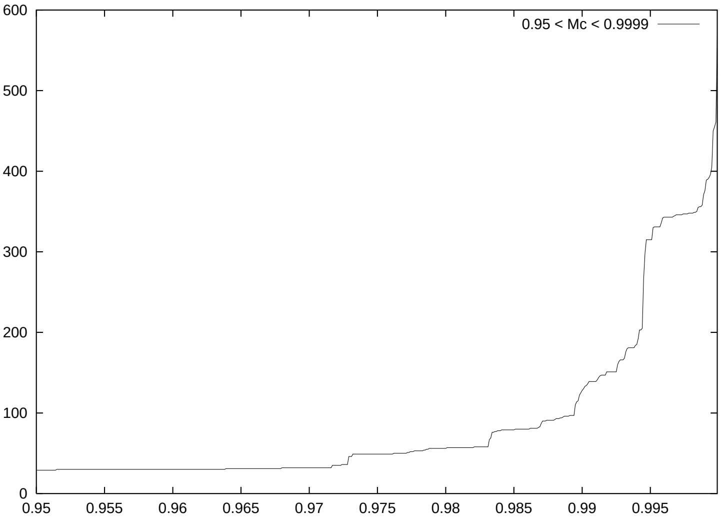

図 3.2: 1610 ドメインに対する指標 c の適用 (拡大)

1. 名前サーバも組織ネットワークも遠い海外
2. 国内サイトだが遠い (`ido.ne.jp.` 等)
3. セカンダリをプライマリの近傍に置く、冗長性の低い運用形態 (大学に多い)
4. 中間の名前サーバで冗長性が低い (`ntt.co.jp.`, `cst.nihon-u.ac.jp.`)
5. 名前サーバを一極集中運用 (`ibm.net.`, `aol.com.`)
6. ccTLD の名前サーバが集中している (`tw.`)

これらのドメイン名を含め、特徴的事例を節 3.2.2 にて分析する。

3.2.2 事例分析

本節では、いくつかの事例を分析する。分析に先立って、それぞれの事例について簡単にまとめる。

- **asahi.com.**
  asahi.com. の特徴は、分散したミラーサーバを持っている点である。この場合は指標 c の仮定から外れる。しかし、セカンダリ名前サーバを適切に配置する事で、コミュニケーション領域と名前解決領域がほぼ一致しているので、「DNS 広域依存問題」は発生しづらい。

- **アメリカ・オンライン**
  名前解決領域での信頼性が低い例。

- **NTT**
  下位ドメイン名の名前サーバは信頼性が高いが、上位ドメイン名の名前サーバは信頼性が低い例。これにより、下位ドメイン名を解決する際、上位ドメイン名の解決にのみ使われる領域が発生し、そこから「DNS 広域依存問題」が発生する可能性がある。

- **大学: スタンフォード大学**
  スタンフォード大学の名前サーバは集中運用されており、そのサーバに至るネットワークは

名前解決にのみ用いられる領域の例。

- **ccTLD: tw 等**
  特定の領域に TLD が依存している例 (`tw.`) と、そうでない TLD との比較。
- **OCN: itojun.org.**
  広い領域が名前解決のみのために必要な例。

asahi.com

`asahi.com.` とは、朝日新聞社が提供する WWW ベースの新聞である。1999 年 12 月現在では、情報発信元の `www.asahi.com.` 以外に、公式ミラーサーバとして `ntt.asahi.com.`、`iij.asahi.com.`、`spin.asahi.com.`、`tokyonet.asahi.com.` が存在する。また、`www.asahi.com.` 自体もラウンドロビン式に複数の非公開ミラーサーバへアクセスが振り分けられており、`wwwld[1-4],asahi.com.`(実体は全て同じ)、`mf02.asahi.com.`、`kdd.asahi.com.` が確認できる。これら `asahi.com.` に関連する名前サーバ群、HTTP サーバ群とそこまでの経路を図 3.3 に図示する。

この名前解決系のグラフを見る限り、DNS 広域依存問題に強い影響を与えるような、決定的な経路は存在しない事がわかる。経路は `203.178.141.57` で 2 方向に分岐しているが、そのどちらにもそれぞれ一つの名前サーバが存在する。従って、`203.178.140.33` から観測する限りは `asahi.com.` の名前解決系は良く機能すると言える。

`asahi.com.` に関して 1 段階だけの M_b を適用すると、名前サーバはそれぞれ 10 ホップ、14 ホップ向うに存在する事から、$M_b = 1 - \{(1 - R^{10}) \cdot (1 - R^{14})\} = 0.987$ となる。ここで、仮に遠い方の名前サーバが存在しなかったとすると、$M_b = 0.904$ となり、逆に名前サーバを一つ増やし、これを 10hop 先にあると仮定すると、$M_b = 0.999$ となる。ルートネームサーバの信頼性が 1 に限りなく近いと仮定すると、このような SLD(Second Level Domain) に直接存在するドメインは、そのドメイン自身の信頼性のみによって信頼性が決定し、名前サーバの適当な分散によって大きい信頼性を得る事ができる。

アメリカ・オンライン

`aol.com.` は米国の通信プロバイダ、アメリカ・オンラインのドメイン名である。`aol.com.` に権限を持つ名前サーバを表 3.2 に示す。

表 3.2: アメリカ・オンラインの名前サーバ

ホスト名	IP アドレス	ネットワーク位置
DNS-01.NS.aol.com.	152.163.159.232	IIJ America → alter.net. → atdn.net.
DNS-02.NS.aol.com.	205.188.157.232	quest.net. → atdn.net.

`aol.com.` はネットワーク的に 2 つの場所に名前サーバを持っている。その 2 つは物理的にはワシントン近傍に位置すると思われるが、上流プロバイダとして 2 つとも異なるプロバイダが選択され、結果的にネットワーク的位置は分散している。

しかし、少なくとも観測地点である `adely.sfc.wide.ad.jp.` からは、これら 2 つの名前サーバはそれぞれ 21 ホップ、22 ホップと遠方に存在し、その間のネットワークの信頼性は高いとは言えない。

アメリカ・オンラインは、日本でもサービスを積極展開している。しかし、日本から見た名前解決系としては、アメリカ・オンラインの信頼性は相対的に低い。

NTT

`ntt.co.jp.` の名前サーバを表 3.3 に示す。表からも読み取れる通り、複数の名前サーバを持っているとは言え同じ物理ネットワーク上に位置すると思われ、名前解決系としての信頼性は名前サーバ一つの場合と大きな差が無い。

表 3.3: ntt.co.jp. の名前サーバ一覧

ホスト名	IP アドレス
ns1.ntt.co.jp.	210.163.32.51
ns2.ntt.co.jp.	210.163.32.52

しかしながら、`ecl.ntt.co.jp.` は表 3.4 に示す通り、ネットワーク的に分散した名前サーバを多数持っている。`tas.ntt.co.jp.`、`tnl.ntt.co.jp.` も同様である。

表 3.4: ecl.ntt.co.jp. の名前サーバ一覧

ホスト名	IP アドレス
tama55.ecl.ntt.co.jp.	129.60.39.103
tama3.tas.ntt.co.jp.	192.68.248.40
nacsis-10.sinet.ad.jp.	150.99.111.65
tama.tas.ntt.co.jp.	192.68.237.114
tama2.tas.ntt.co.jp.	192.68.237.113
tama5.ecl.ntt.co.jp.	129.60.39.102

第 3 段階 (`ntt.co.jp.`) と第 4 段階 (`ecl.ntt.co.jp.`) の名前サーバ群への経路をグラフ化すると、図 3.4 となる。第 3 段階はこの図中左下の `210.190.177.2` 方面のみに集中しており、この方面の接続性が失われると、`ntt.co.jp.` ドメイン全体の名前が解決できなくなる。

大学の例

大学等では、しばしば技術的見地からは不適切な運用を行う事がある。そうした大学のネットワークは、一箇所に集中している場合が多く、DNS 広域依存問題が顕在化する事は少ない。ここでは、米国スタンフォード大学の名前サーバを例に挙げる。

スタンフォード大学のドメイン `stanford.edu` に権限を持つ名前サーバは、表 3.5 の通りである。表を見れば明らかな通り、全て同じ物理ネットワークに接続されており、そのネットワークの接続性が失われれば `stanford.edu.` 以下の全ての名前が解決できなくなる。

スタンフォード大学等の、地理的・ネットワーク的に集中している組織の場合は、名前サーバが所属するネットワークへの到達性が失われる事と、サイト自身への到達性が失われる事はほぼ等しい。が、このネットワーク構成では、サイト自身への到達性が確保されているにも関わらず、サイト内部のネットワーク障害で名前サーバへ到達できない可能性が高まる。

表 3.5: スタンフォード大学の名前サーバ

ホスト名	IP アドレス
Atalante.stanford.edu.	36.56.0.150
Argus.stanford.edu.	36.56.0.151
Avallone.stanford.edu.	36.56.0.152

いくつかの ccTLD

ccTLD の中に、信頼性が不十分なものが存在する。特に顕著な例が、`tw.` (台湾) である。`tw.` に対して権限を持つ名前サーバと、観測地点からそれぞれの名前サーバまでの経路を、図 3.5 に示す。

図 3.5 を一見すれば明らかなように、`tw.` ドメインの名前サーバは殆んど同じネットワーク位置に存在する。図 3.5 の経路に障害があると、観測地点から、`tw.` ドメイン全てが利用できなくなる。代替経路がどの程度利用できるかは不明だが、経路情報の誤伝播などの障害に対して代替経路は無力な点に注意が必要である。

一方、ドメイン名を全世界に開放した ccTLD の例として、`ms.`、`cc.`、`nu.` 等が存在する。これらドメインについては、国内や米国西海岸にセカンダリサーバが存在しており、十分な信頼性を持つ。以上の ccTLD に対して、信頼性を表す指標 (指標 b) を求めた結果を、表 3.6 に示す。

表 3.6: ccTLD における指標 b の値 (一部)

ccTLD	指標 b
tw.	1.7×10^{-2}
ms.	9.9×10^{-5}
cc.	1.4×10^{-4}
nu.	2.9×10^{-5}

表 3.6 を見ると、`tw.` とその他の ccTLD の信頼性に大きな差が存在する事が明確である。

itojun.org.

`itojun.org.` は、日本の一個人が管理・運営するドメインで、ネットワーク的には NTT の OCN 網の中に存在する。ここで、`org.` と `itojun.org.` への経路を図 3.6 に示す。

図 3.6 中、送信者ホスト (203.178.140.33) は右下に存在し、受信者ホスト (210.160.95.97: `www.itojun.org.`) と `itojun.org.` に権限を持つ名前サーバは、図中左上に集中して存在する。送信者ホストから受信者ホストへの経路を境に、右上方向に広がっているのが WIDE LAX 海外線経由の米国のネットワークであり、左下方向に広がっているのが APAN 経由の米国のネットワークである。米国のネットワーク内に散っている名前サーバは、`org.` ゾーンに権限を持つ名前サーバ群である。送受信者ホスト及び名前サーバ群は、色を変えて表示してある。

この構造を観察すると、コミュニケーション領域の両脇に大きく張り出した名前解決領域の存在が判る。`org.` を解決するためには両側に広がる名前サーバのうち少なくとも一つが利用可能でなければならない。仮に、WIDE 海外線の先に存在する中継ルータ (209.137.144.97) 及び apnic の中継ルータ (203.181.248.207) の両方に障害が発生した場合は、`itojun.org.` の解決が不可能にな

り、正常な通信ができなくなる。

3.2.3 事例分析の結論

事例分析によって、名前サーバが少ない例、名前サーバが複数あっても同じネットワークに存在する例、下位ドメインの信頼性は高いが上位ドメインの信頼性は低い例などを見てきた。

`asahi.com.` の例のような、情報提供を目的とするドメインの場合は、ミラーサーバが多く存在するネットワークの近傍に名前サーバを設置する事で耐障害性が向上する事が判る。もしもミラーサーバが存在しても、その近傍に名前サーバが存在しなかったならば、ミラーサーバへの名前到達性が確保できず信頼性を損うものとなるだろう。

`ntt.co.jp.` の例のように、名前空間木で中間に位置する名前サーバの信頼性が、十分でない場合がある。名前空間の権限委譲は一方向であり、NS RR による参照も一方向にしか行われない。従って、下位ドメインが単独でいくら信頼性を確保しても、上位ドメインの解決に失敗してしまうと、下位ドメインの名前は一切解決できない。即ち、`ntt.co.jp.` のような例では、信頼性にボトルネックが生じている。

ccTLD の例では、台湾のドメインの名前サーバがネットワーク的に集中して存在しており、全ての名前サーバへの接続性が失われる危険性が高い事を見た。この場合は `ntt.co.jp.` と同じような、信頼性のボトルネックが TLD 全体に作用しており、より広範囲に影響する。また `itojun.org.` においては、`freebsd.org` の例と同じく、名前解決により広いネットワーク領域が必要である事を見た。`asahi.com.` 等も、TLD の解決においては同様の問題がある。

以上、名前サーバを分散させる事で、コミュニケーション領域と名前解決領域のずれを防いでいる例と、名前サーバの配置が理由で、コミュニケーション領域と名前解決領域のずれが生じ、「DNS 広域依存問題」を顕在化させやすい例を考察した。

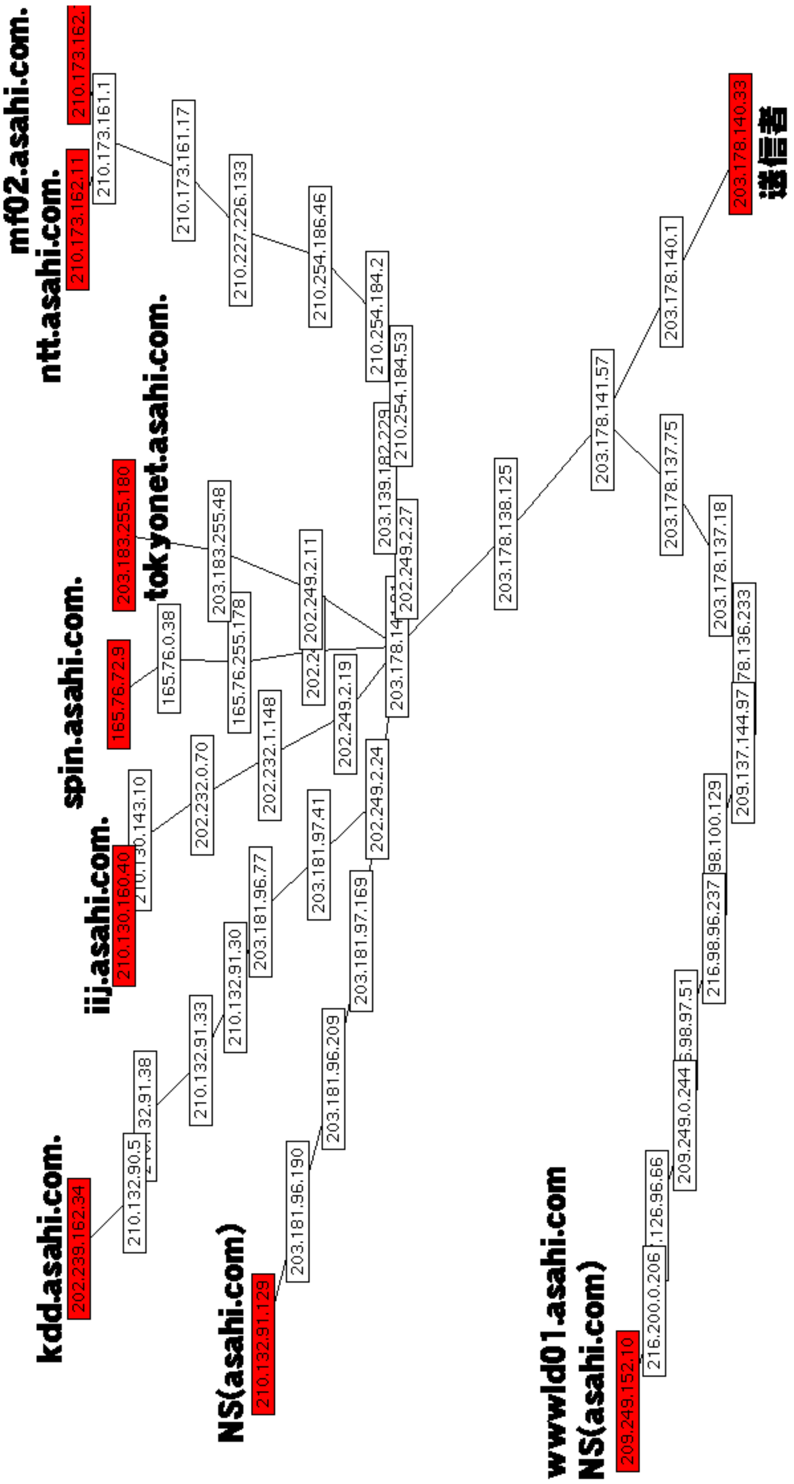

図 3.3: asahi.com. の名前サーバ及び http サーバ

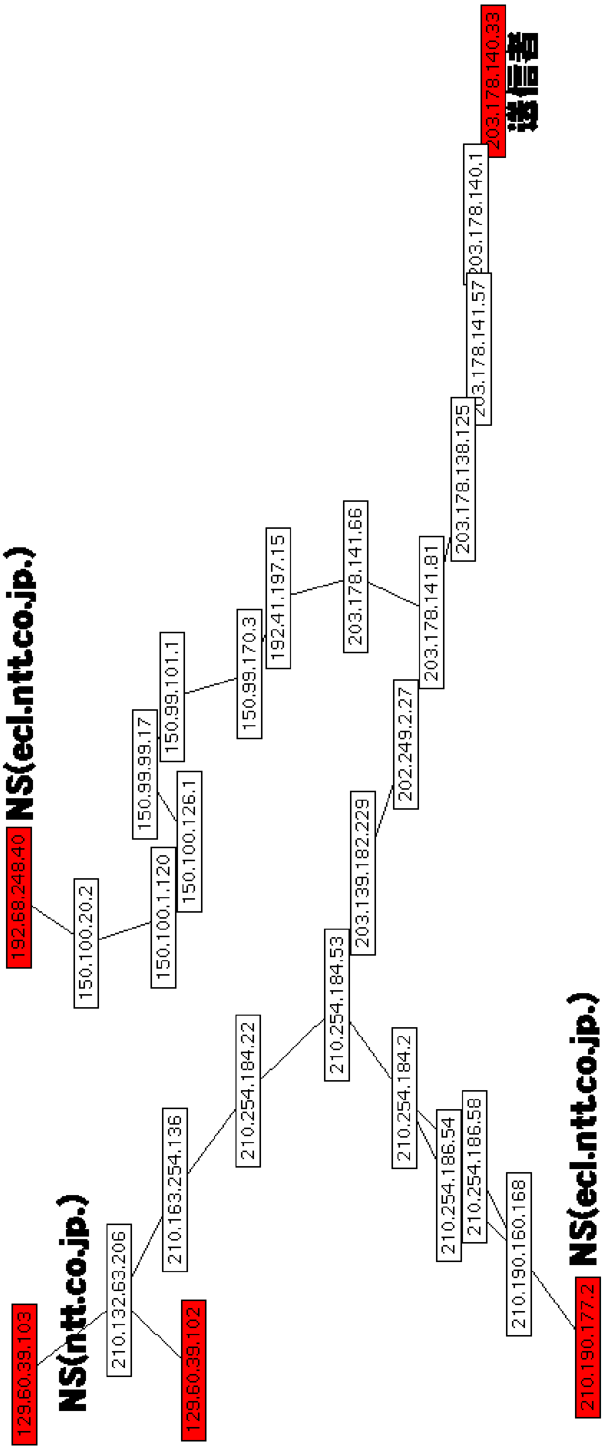

図 3.4: ntt.co.jp. 及び ecl.ntt.co.jp. の名前サーバ

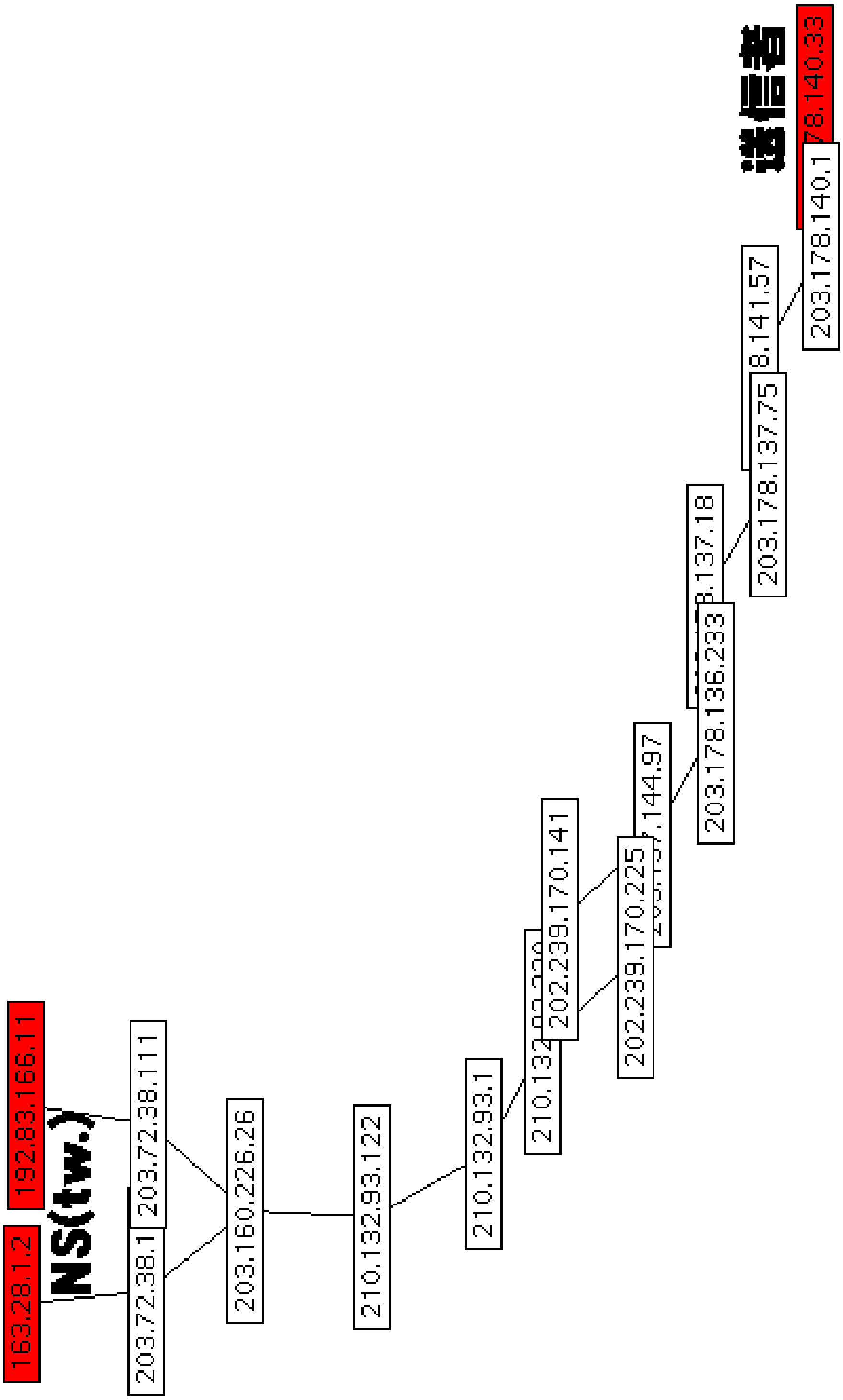

図 3.5: tw ccTLD の名前サーバ

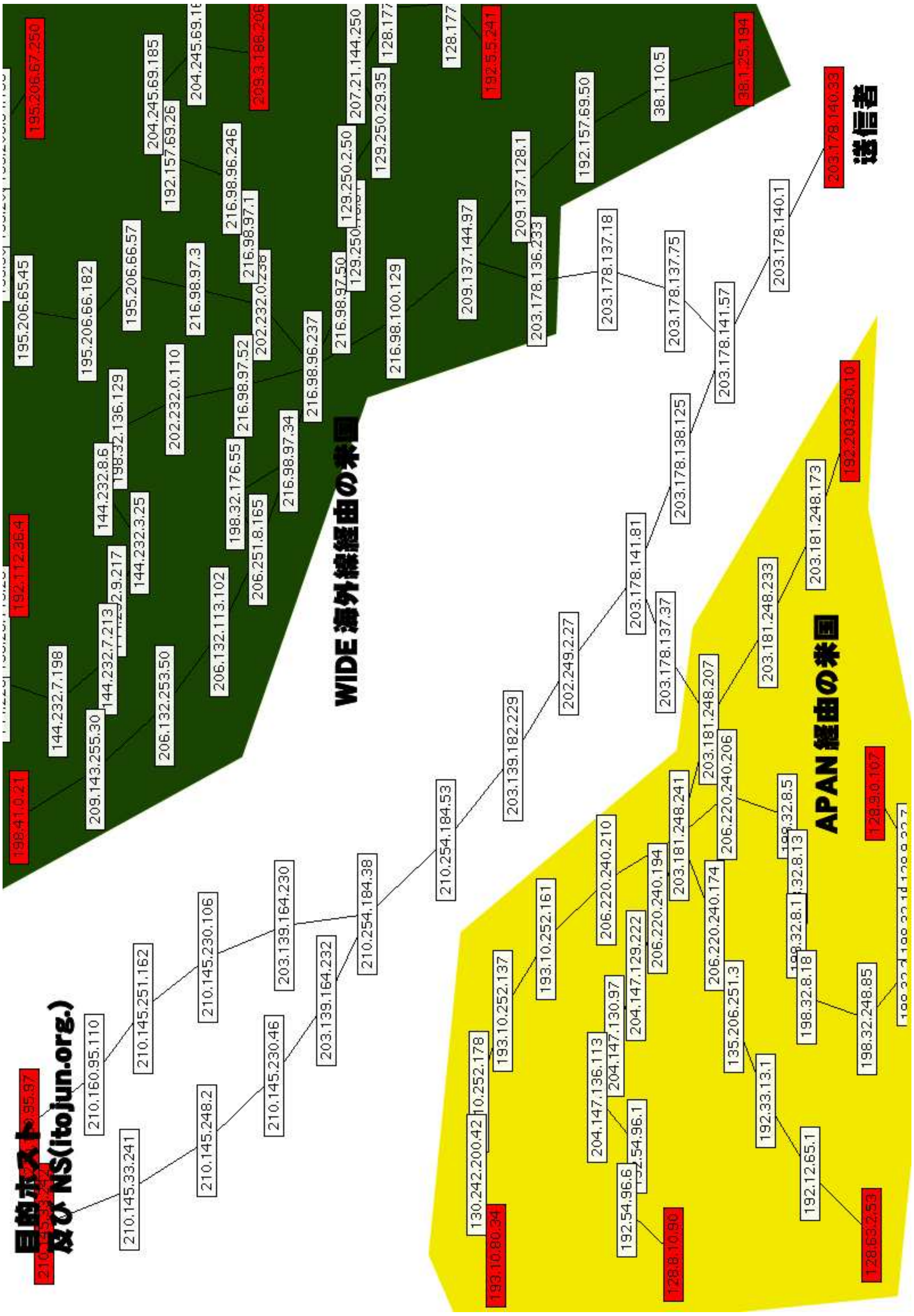

図 3.6: itojun.org に関連する名前サーバ

第4章　解決へのアプローチ

本章では、DNS 広域依存問題を解決するために、NSR3(Name Server Resource Record Recycle) を提案する。この手法は、DNS のいくつかの特性に着目し、問い合わせの過程で取得した名前サーバ情報を捨てずに、NSR3 名前キャッシュにほぼ無期限に保存する。そして、もしも名前解決に上位の名前サーバが必要にも関わらず、障害により利用できない場合、保存しておいた情報を再利用する。

4.1　NSR3: Name Server Resource Record Recycle

本研究で提案する名前サービスキャッシュ機構の拡張である NSR3: Name Server Resource Record Recycle について述べる。

NSR3 の目的は、DNS 広域依存問題の解決である。DNS の広域依存の原因となっている再帰問い合わせに着目し、ここまでに述べた仮定を出発点として DNS を拡張する。

NSR3 の要点は、NS リソースレコードの一時記憶を、期限が切れた後も保管しておく点にある。そして、もしも遠隔の名前サーバが利用できない時に、このレコードを再利用する事で、切断された名前空間木 (図 2.1) を再接続する。

4.1.1　NSR3 における仮定

NSR3 は、以下の仮定を出発点とする。

1. 名前サーバは一般にほとんど移動しない (節 4.2.3)
2. 腐った NS RR キャッシュは深刻な脅威ではない (節 4.2.2)
3. 解決すべきドメイン名の、「最後の 1 段階」に権限を持つ名前サーバは利用可能 (節 4.2.4)

4.1.2　再帰問い合わせの枠組みの拡張

NSR3 では、DNS 広域依存問題を解決するための自然な拡張として、図 4.1 のように名前問い合わせに失敗した場合の名前解決手段を追加する。この拡張された名前解決手段を、「NSR3 名前キャッシュ」と呼ぶ。NSR3 名前キャッシュは、名前解決系における「最後の手段」である。図 4.2 は DNS における名前解決の手段を、NSR3 も含めて相対化したものである。情報は常に権限を持つ名前サーバに対してネットワークを利用して問い合わせた結果は、常に正しく (完全)、キャッシュと NSR3 名前キャッシュは同等で、完全ではない。しかし、情報の強さ、すなわちどの情報が優先的に参照されるかは、図 4.1 の流れから判るとおり、名前キャッシュ> 名前サーバ >NSR3 名前キャッシュの順である。

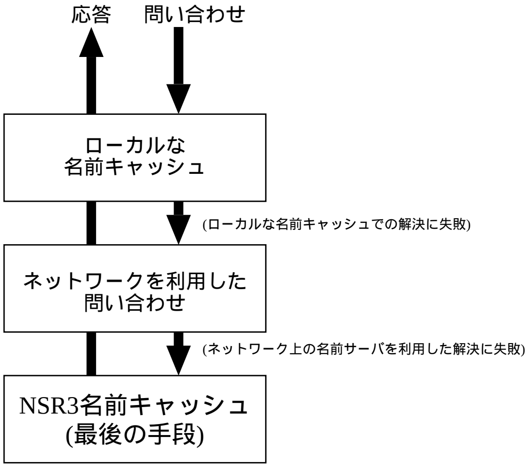

図 4.1: NSR3 による問い合わせの枠組の拡張

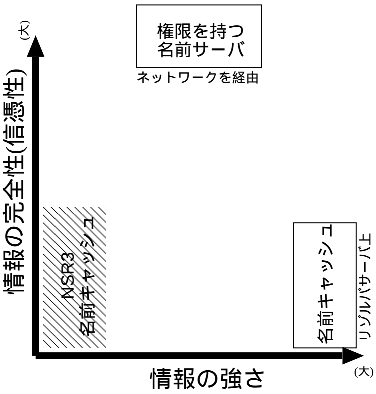

図 4.2: 情報の強さと信頼性

4.1.3 名前サービス情報の再利用

NSR3 により、図 2.1 のように何らかの障害によって名前空間木が切断された場合でも、図 4.3 のように切断された木の節を再発見でき、切断された名前空間を継続的に利用できる。この場合は、`ac.jp.` に対応する節を記憶しておく事で、`ac.jp.` 以下のドメインを再発見している。

図 4.1 で見た問い合わせの枠組の拡張 (NSR3 名前キャッシュ) における名前解決は、節 4.1.1 で述べた仮定を元に行う。この段階で、通常の名前キャッシュも名前サーバも利用できない事が確定している。

仮定 1「名前サーバは一般にはほとんど移動しない」仮定 2「腐った NS RR キャッシュは深刻な脅威ではない」を踏まえ、NSR3 では NS RR(Name Server Resource Record) を再利用 (Recycle) する。具体的には、図 4.1 の最上層である名前キャッシュで「期限切れ (expired)」となった NS RR、及び対応する名前サーバの A/AAAA RR を捨てずに、NSR3 名前キャッシュ中に保存しておく。そうした古い情報を、再帰問い合わせ中に名前キャッシュも名前サーバも利用できない事態に遭遇した場合に適宜再利用する。

また、仮定 3 により、名前サーバ以外の情報、特にホスト名を解決するための A RR は NSR3

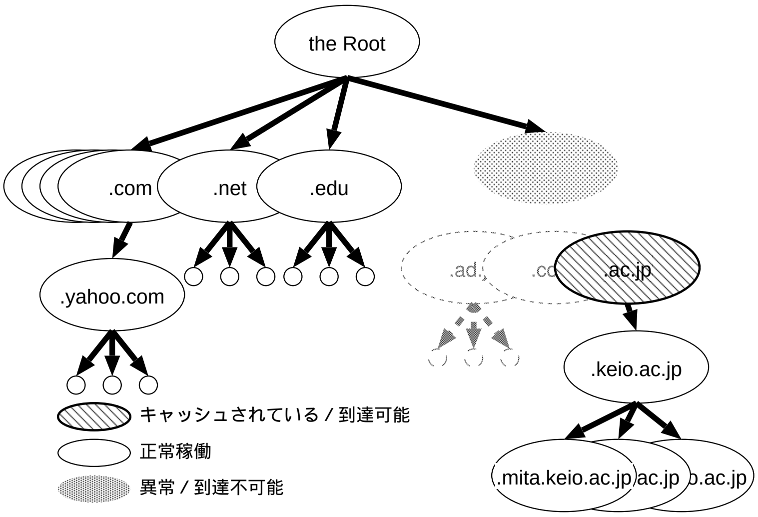

図 4.3: 切断された名前空間木の再発見

名前キャッシュに保存する必要はない。つまり、不要な情報を取っておいて記憶領域を圧迫する必要はない。逆に、仮定 1 および仮定 2 は名前サーバのみに対して成立するので、ホスト名に対する情報は NSR3 名前キャッシュに保存することは危険である。

但し、障害によって切断された名前空間木の再発見ができる場合は、NS RR が NSR3 名前キャッシュに存在する場合のみである。すなわち、これから解決する名前は、過去に一度でも解決した事のある名前でなければならない、という制約がある。しかし、ドメイン名空間が有限である以上、新規に登場する名前は時間と共に減っていく。

4.1.4 特長

NSR3 の特長を以下に述べる。

- 局所的に適用できる
  NSR3 は、リゾルバサーバ側の技術であり、局所的に適用することが可能である。DNS はインターネット全体で運用しているシステムであり、全体のシステムを変更する事が事実上不可能である以上、局所的に適用できる技術は普及の面で決定的に有利である。

- グローバルな名前空間を対象としている
  NSR3 の適用範囲が局所的である一方、NSR3 の効果はグローバルな名前空間を対象として作用する。つまり、NSR3 を適用したリゾルバサーバ、あるいはサイトにおける全ての名前解決が、NSR3 による恩恵を受けることができる。

- 情報の新鮮さや性能に全く影響を与えずに信頼性が向上する
  NSR3 は、節 4.1.1 で述べた仮定に基づき、NS RR のみに関して再利用を行う。その再利用は既存の名前キャッシュと名前サーバへの問い合わせの両方が失敗した時 (図 4.1) のみに、NSR3 名前キャッシュによって行われる。従って、情報の新鮮さは既存の枠組のまま変更されず、NSR3 は何ら悪影響を及ぼさない。一方、障害により名前空間木が切断された場合は、NSR3 により障害回復が試みられるので、結果的に信頼性が向上する。

4.2 名前サーバの性質

本節では、名前サーバの性質に着目し、整理する。同時に、名前サーバの性質から、NSR3 が前提としているいくつかの仮定について検証する。

4.2.1 再帰問い合わせの性能と情報の新鮮さとのトレードオフ

DNS 広域依存問題を低減させるために、各リソースレコードの TTL(有効期限) を長く取る事が考えられる。これにより、リソースレコードはリゾルバサーバに長い間残るので、その期間内は遠隔の名前サーバを利用する必要が無い。実際、`com.` などの大規模な TLD は TTL に 6 日間という長い期間を指定しており、その間は TLD の解決は不要になる。見方を逆転させると、名前サーバ上の情報の変更がリゾルバサーバに「伝わらない」時間を決定づける値が TTL であると言える。

TTL が長ければ、再帰問い合わせの回数は減り、名前解決系の性能は向上する。また、再帰問い合わせの回数が減少すれば、名前解決系の信頼性も向上する。逆に、TTL が 0 に近づけば再帰問い合わせの回数が増加し、性能、信頼性共に低下する。特に性能の低下は著しく、TTL を 0 に設定する事は一部の例外を除いて推奨されない。

しかし、TTL が長くなると、それだけ名前サーバ上の情報の変更が他のリゾルバサーバに伝わりにくくなる。つまり、キャッシュ内に長く情報を置いておくと、現実の変化に追随できず、結果的にキャッシュの情報が誤った情報となる可能性がある。このようなキャッシュを「腐ったキャッシュ(stale cache)」と呼ぶ。腐ったキャッシュは、コミュニケーションの成立を完全に阻害する危険が高く、「DNS 広域依存問題」よりも重大な問題である [1]。

従って、TTL の設定は、性能の低下と腐ったキャッシュの発生との間のトレードオフの関係にある。

4.2.2 NS リソースレコードの性質

腐ったキャッシュがコミュニケーションを阻害する可能性が高い。しかし、NS リソースレコード (NS RR) に関しては例外が存在する。

リゾルバサーバに、NS RR の腐ったキャッシュが存在する場合を考える。この場合、リゾルバサーバが「古い」名前サーバに対してアクセスしてしまう。古い名前サーバは以下の 3 つの状態のいずれかであると考えられる。

1. 対象となるゾーンについて権限を持たない
2. 非公式なセカンダリの設定を行い、セカンダリとして機能
3. 古い情報を持ったまま、依然名前サーバとして機能

このうち、上記の状態のうち、状態 1 については、ゾーンに権限を持つ事を期待して名前サーバに問い合わせたにも関わらず、ゾーンに権限を持たない回答が返って来る。従って、異常の存在を検知できる。実際に、インターネットで最も広く利用されている名前サーバの実装である BIND[9] では、このような権限を持たない名前サーバ (状態 1) を回避する機能がある。また、状態 2 に関しては、非公式ながらセカンダリとして機能しているので、マスタサーバ上にある正しいデータベースを参照できているものと見なせる。一方、状態 3 の場合は、古いサーバ上に残っている古い情報にアクセスしてしまう。

つまり、腐った NS RR キャッシュによってコミュニケーションが阻害される危険が残る場合は、状態 3 の時のみであり、それ以外の状態の場合は腐った NS RR キャッシュを持っていても確実にコミュニケーションを成立させる事ができる。そして、状態 3 の発生は名前サーバの移動を行う管理者が十分に注意深ければ防ぐ事ができる。従って、腐ったキャッシュの問題は、**NS RR** のみに関しては、大きな脅威ではない。

4.2.3 名前サーバの移動可能性

ここで、名前サーバの移動可能性、即ち NS リソースレコード (NS RR) の変更可能性について考察する。

名前サービスはネットワークサービスのうち根幹に最も近い。新規ネットワークの設計において、通信の成立、つまり経路制御の正確さや物理層での信号伝達が確保できたならば、次にすべき事は、ネットワーク上の要素に名前を付ける事だからである。名前サービスには DNS 以外にも NIS(Network Information System)[15] 等が存在するが、ここでは DNS のみを検討する。

名前サービスがネットワークサービスの根幹に近いという事は、他のサービスは名前サービスに依存している事を意味する。名前サーバに依存するサービスが多い以上、名前サーバの移動はコストが高い。従って、通常は、最初から名前サーバを動かさずに済むようにネットワークを設計する。

他のサービスへ影響を与えずに名前サーバを移動させるには、少なくとも以下のような手順を取る必要がある。ここで、移動元のホストを A、移動先を B と表記する。また、ホスト A が権限を持つゾーンに対して、直上に位置するゾーンに権限を持つ名前サーバを、親サーバ P とする。(ここでは、ホスト A が一つのゾーンに関してのみマスタサーバとして機能していると仮定する。)

1. ホスト A でのゾーンの内容更新を禁止する。
2. ゾーンファイルをホスト B にコピー
3. ホスト B で名前サービスを開始
4. ホスト A をホスト B のスレーブに設定
5. (以後のゾーン更新はホスト B に対して行う)
6. 親サーバ P 上の NS RR のうち、ホスト A を示すものをホスト B を示すものへと変更する。
7. ホスト A に対してゾーン転送を行っているスレーブサーバを全てホスト B に対してゾーン転送を行うように設定変更
8. 名前サーバのログを数日単位で観測し、ホスト A に対して問い合わせとゾーン転送要求が来ない事を確認
9. ホスト A の名前サーバを停止

また、セカンダリサーバの変更においても、マスタサーバ側で IP アドレスによるアクセス制御を行う場合があり、その設定変更などに注意しなければならない。

以上のように、名前サーバの移動には大きな工数がかかる。従って、技術的理由から、名前サーバは滅多に移動しない。

4.2.4 名前サーバとホスト名との関係

名前サーバとホスト名に関しては、多くの場合以下の関係が成立する。(説明の都合上、「サイト」を「一つの管理主体によって管理されるネットワーク」と定義する。)

「ホスト名を管理する名前サーバは、ホストと同じサイトに存在する」

当然、ホスト名、ホストへの物理的な接続、IP アドレスの割り当てなどの管理者は同じである事が多い。上記の仮定はむしろ、「サイト内には、ホスト名を管理する名前サーバとホストが存在する」と言うべきものである。

NSR3 の仮定 3 は、上記の仮定に基いている。解決すべき問題は、対象ホストへの接続性が保たれているにもかかわらず名前解決に失敗する場合である。一方、対象ホストと、ホスト名を管理する (ホスト名の 1 段階上の) 名前サーバは同じサイトに位置する事が期待できるので、ホスト名を管理する名前サーバへの到達性が確保できない場合の問題解決は、ホストへの到達性が確保できない場合の問題を解決する事にほぼ等しい。そしてこの問題 (ホストへの到達性が確保できない場合) は本研究の研究対象外である。

例外は、一部のミラーサーバである。例えば、世界中に分散している http サーバ群に対して一つの名前を割当てる事は可能である。しかし、このようなドメイン名はホスト名と言うよりもサービス名と言うべき名前であり、ドメイン名の例外的な利用法である。また、地理的な存在を持たない任意団体のドメイン等の場合は、そのドメインに含まれるホストが実際は様々な組織に所属する全く別のホストだという場合も存在する。このような場合、それぞれのミラーサーバの近傍に名前サーバを置く事で、名前解決領域とコミュニケーション領域のずれを防ぐ事ができる (節 3.2.2 の `asahi.com.` の例)。

4.3 NSR3 の適用

本節では、節 4.1 で述べた NSR3 の、実環境への適用手段について述べる。

インターネットは誕生して長い時間がたっており、その技術的社会的蓄積は膨大な量になる。この多大な蓄積が、DNS のようなインターネット基盤技術の改良を困難なものにする。何らかの問題を、歴史的蓄積になるべく影響を与えずに解決するためには、解決手法の適用は注意深く行わなければならない。

特に、既存の実装やシステムに悪影響を与えない事は第一の必要条件として挙げられる。同時に、既存の実装そのものに手を加える事はインターネットの規模の大きさから大変な困難を伴う事も念頭に置く必要がある。

4.3.1 適用の位置と範囲

まず、一般的なネットワーク構成のうち、NSR3 の適用に適した位置と適用領域に関して考察する。

NSR3 名前キャッシュの有効性を活かすためには、できるだけ大きい範囲に NSR3 を適用すれば良い。しかし大きすぎると NSR3 の影響範囲が広くなり、DNS そのものの見通しは悪くなる。この両者の均衡を取る適用範囲は、一つの「サイト」である。本論文では、サイトを「一つの管理主体で管理されるネットワーク」と定義している。

最適な適用範囲がサイト単位である理由は次の通り。サイトは単一の管理主体で管理される最大単位であり、一つのサイト内での効率的知識共有は一般に期待できる。そして、管理知識の共有は、NSR3 の対価として与えられる DNS の障害検知および復旧の困難さを低減する。もし、NSR3 の適用がサイト間となると、管理主体が異なるため、管理知識の十分な共有ができなくなる。管理者同士の連携がうまくいかないと、障害があった場合の障害検出および原因究明に手間取る可能性

が高まる。なぜならば、NSR3 は利用者に DNS の障害を検知させないための機構であり、障害を自動的に隠蔽する特性を持つからである。

サイト内で NSR3 が適用できる地点は次の 2 箇所である。

1. あらゆるホストが利用するリゾルバサーバ
2. サイト境界ルータ (サイト出入口が一つの場合のみ)

サイト内に、あらゆるホストが利用するリゾルバサーバが存在すれば、そのリゾルバサーバ上で NSR3 を適用すると最も効率が良い。なぜならばサイト内の DNS 問い合わせはそのリゾルバサーバに集約され、キャッシュや遠隔名前サーバの障害検出も同時に集約できるからである. リゾルバサーバ上の NSR3 キャッシュを、図 4.4 に示す。

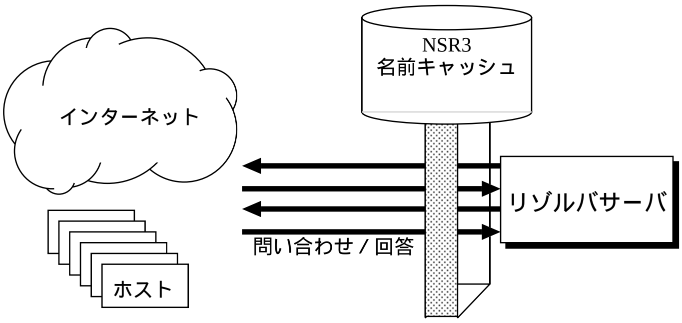

図 4.4: リゾルバサーバ上の NSR3

もしサイト内に、サイト内の名前解決を代行するための、集約化されたリゾルバサーバが存在しない場合は、他の点で NSR3 を適用した方が効率が良い。もし、上流ネットワークへの接続点が一箇所であれば、その点で NSR3 を適用できる。この適用の模式図を図 4.5 に示す。特に上流ネットワークへの接続の手前に Ethernet のような共有メディア LAN が存在した場合、その LAN 上でメッセージを 「見張る」ことができる。

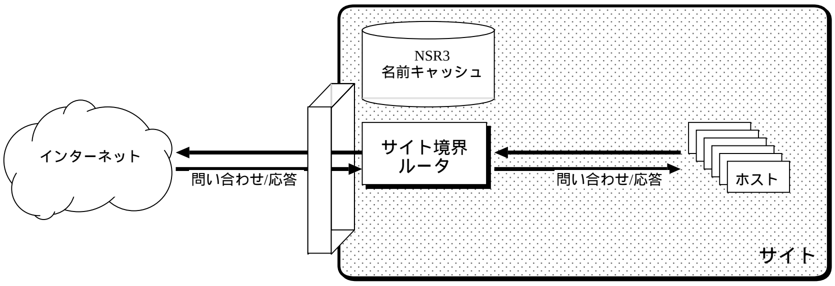

図 4.5: 上流ネットワーク接続点上の NSR3

サイト出入口が一箇所の場合の好例は、SOHO(Small Office/Home Office) 環境に見る事ができる。SOHO 環境は多くの場合、安価な専用線接続かダイアルアップによる間欠接続を利用している。特に間欠接続の場合、ネットワークが利用できない場合の内側のネットワークの名前解決系を調整するために、実際に嘘の DNS メッセージを作成し内側のリゾルバライブラリを 「騙す」実装が存在する。

逆に、NSR3の適用にふさわしくない地点は、複数の出入口があるサイトの出入口である。NSR3は問い合わせと応答の組み合わせを観測して遠隔のサーバが健全に動作しているかを判断する。しかし、ネットワークの出入口が複数ある場合、問い合わせと応答のメッセージが同じ出入口を通るとは限らない。従って、NSR3が誤った判断を下し、ネットワークに無用な混乱を与える危険性が高い。但し、複数の出入口上のNSR3が協調動作できる場合はこの限りではない。

4.3.2 適用技法

NSR3の適用のためには、名前解決のメッセージをどこかで覗き見する手段を持ち、障害を持つ名前サーバのかわりに情報を問い合わせ元に伝える手段を持たなければならない。その条件を満たす技法には、次の3つが考えられる。

- BIND named の改造
  - 利点: 標準的なリゾルバサーバに導入できるので、普及が容易である。
  - 欠点: 改造コストが高い上に、実際に改造コードを開発者側が受け入れるための交渉が必要。リゾルバサーバでしかNSR3が適用できない。
- Y-splitter
  - 利点: 既存の実装の変更の必要がない。
  - 欠点: リゾルバサーバ以外の地点での適用ができない。
- Stealthy Cache
  - 利点: リゾルバサーバでも境界ルータでも、共有メディアLAN上でもNSR3を適用できる。
  - 欠点: パフォーマンスの低下が避けられない。トラフィックの覗き見とIP送信者アドレス偽造の問題。

named の改造

`named`あるいは`BIND`[9]とは、カリフォルニア大学バークレイ分校で開発された、インターネットで最も広く利用されている名前サーバの実装である。現在は開発はISC[22]に移管され、様々な発展を続けている。

NSR3の有効性の実証として、`named`を改造し、NSR3機能を実装する事は可能である。しかし、`named`自身はコア部分でも3万行を超える巨大なプログラム[17]であり、かつ古く伝統的なプログラムなので、内部の構造が複雑である。また、広汎に利用されているとは言え、実際に利用されているシステムはバージョン番号の違いなど様々なバリエーションが存在する。これら全てにNSR3を適用する事は非現実的である。

以上のように、`named`へのNSR3機能の実装はコストが高く、本研究における実証実験として適切ではない。しかし、NSR3の有用性が実証され、NSR3を普及させる段階になれば、この手法は最も有用である。

Y-splitter

Y-splitter は、DNS に関する技術的工夫の一つである。その目的は、2-faced DNS と呼ばれる問い合わせの送信者アドレスを見て問い合わせに対する回答の内容を変える名前サーバの実現である。ヤマハ株式会社から発売されているダイアルアップルータ、RT シリーズには、既に類似した機能 [16] が実装されている。

Y-splitter は、名前サーバの入出力の手前で DNS のメッセージの行き先を分岐させる。名前サーバは通常 TCP/UDP の 53 番 port を通信に利用する。ここで、実際の名前サーバを 53 番 port 以外に移し、53 番 port には DNS メッセージの中継を行うサービスを置く事ができる。この場合、既存の名前サーバは例えば 10053 番 port で Y-splitter からの入力を待つ事になる。

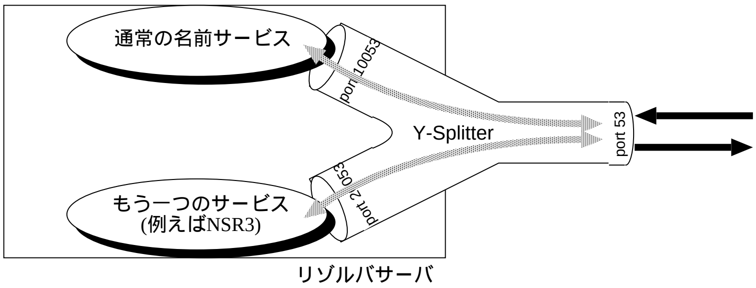

図 4.6: Y-splitter

図 4.6 は、Y-splitter の概念図である。Y-splitter は、53 番 port に与えられた入力を 2 方向に分岐させている。また、分岐させると同時に応答先 (入力したメッセージの送信源) を記憶し、もしも分岐した先が回答が返す場合は、対応する応答先に DNS メッセージを転送する。

Y-splitter を応用し、分岐の片方向に NSR3 の実装、片方向に既存のリゾルバサーバを接続する事が考えられる。実際には、NSR3 の実装がリゾルバサーバからの応答を検知するための仕組みが必要である。また、Y-splitter 内部に問い合わせと回答の対応付けを行う部分がある事から、NSR3 を Y-splitter 内部に実現する方法が有利である。

この手法の利点は、既存の実装を改変する必要が無い点である。その上、DNS メッセージ待ち受け port を 53 番 port から変更できる、あらゆるリゾルバサーバに適用できる。但し、この手法では、`named` の改造と同様に、リゾルバサーバ以外の地点では NSR3 が適用できないという欠点がある。

Stealthy Cache

Stealthy Cache とは、アクセス透過型の共有 DNS キャッシュである。つまり、ある通信路を共有するクライアント全体が、透過的にアクセスする名前キャッシュである。図 4.7 にその概要を示す。

Stealthy Cache は、通常はネットワークを流れる DNS トラフィックを覗き見しており、それらのトラフィックに記載されているリソースレコードを自前のデータベースに蓄えていく。そして、返答しない名前サーバを検出すると、その名前サーバへの問い合わせに対して代理回答 (図 4.8) を行う。つまり、返答しない名前サーバになりすまして問い合わせ元に回答する。問い合わせと応答の DNS メッセージのやりとりを観測し、名前サーバからの応答が無い時に、その名前サーバになりすまして割り込むという動作により、Stealthy Cache はアクセス透過のまま DNS キャッシュを

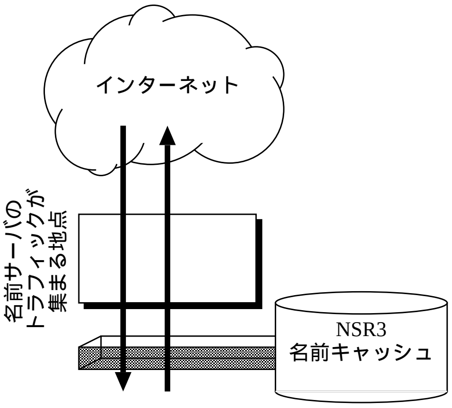

図 4.7: Stealty Cache の動作概要 (平常時)

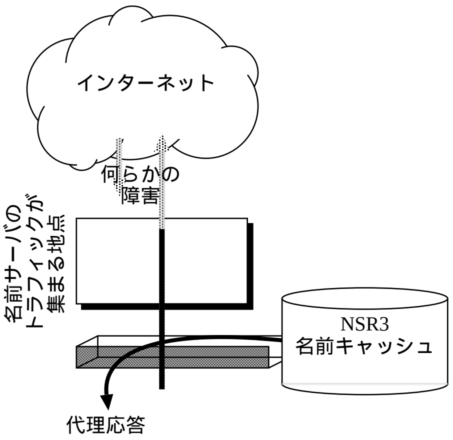

図 4.8: Stealty Cache の動作概要 (代理回答)

共有できる。

Stealthy Cache はリゾルバの存在するサーバ上で動作させれば Y-splitter と同等の機能を持つ。Y-splitter が明示的に port 番号を変更して DNS の通信に介入しているのに対し、Stealthy Cache は低レベルのパケット入出力を行い、直接 DNS の通信に介入する点が異なる。

Stealty Cache の利点は、サイトの境界ルータ等のリゾルバサーバ以外の場所にも適用できる点と、Ethernet 等の共有メディア LAN を利用すれば複数のホスト間でのキャッシュの共有が行える点である。

逆に欠点も存在する。Stealthy Cache では、ネットワークインターフェイスで、全てのパケットを取得してユーザプロセスに渡さなければならない。従って、システムの負荷は大きくなる。また、名前サービスのみとは言え、ネットワークのトラフィックを「覗き見」する事と、場合によっては代理回答の手段として IP 送信者アドレスを偽造する必要がある点は、継続的運用に際して道義的に問題となるかもしれない。

Stealthy Cache は DNS メッセージの通信に対する透過的な介入を行うので、その機能を利用して NSR3 を実装する事ができる。Stealthy Cache のキャッシュ管理ルーチンに対する実装を行えば、NSR3 を実現できる。

第5章　設計

本章では、NSR3 を実現した Stealthy Cache の実装である、dnscap の設計と、その構造について述べる。

5.1　dnscap: Stealthy Cache with NSR3 の設計

dnscap(DNS CAPture) の目標は、DNS 広域依存問題の解決である。この目標を達成するための設計を、以下に述べる。

5.1.1　要求仕様

dnscap の要求仕様は以下の通り。

- 様々な環境での実験が可能
  dnscap の目的は、NSR3 の実証実験である。そのために、なるべくいろいろな環境の実験を可能にする必要がある。
- 既存のリゾルバサーバの実装をそのまま利用
  既存の実装に影響を与えたり、既存の実装を置き変える必要が有る仕様では、実験にならない。実働環境での実験が難しくなるばかりか、変更する箇所が増える程、実験結果の分析が困難になり、NSR3 の影響なのか他の変更点の影響なのかの判別が困難になる。
- 二次記憶上のキャッシュ
  実証実験である事を考慮すると、キャッシュした情報はハードディスクなどの二次記憶に蓄え、分析に備える必要がある。
- 詳細な動作記録
  動作解析のために、動作に関する詳細な記録を残す必要がある。

以上の条件を考慮すると、節 4.3.2 で述べた Stealthy Cache としての実装が適切だと考えられる。

5.1.2　実現に必要な情報

NSR3 の実現のために管理すべき情報は以下の通り。

- 問い合わせと回答の対応表 NSR3 が「遠隔名前サーバの障害」を検出するために、問い合わせと回答の対応づけを行う。つまり、どの問い合わせに対する回答が帰って来ないのか、どの名前サーバが利用できないのかを記録し、検出する。

- 遠隔名前サーバとの通信状態利用できない名前サーバが判明したならば、そのサーバを記録する必要がある。
- 名前キャッシュ

NSR3 機能はこれら 3 つの情報によって実現する。

5.1.3 名前サーバの状態検出と状態保持

NSR3 の動作を決定付ける要素の一つに、遠隔の名前サーバの状態検出がある。名前サーバが利用できるかの判断条件の設定で、NSR3 の挙動は大きく変化する。

節 4.3.2 で述べた技法を利用する場合、名前サーバの状態検出に利用できる情報は、観測した DNS メッセージのみであり、特に問い合わせに対する回答の有無を利用する。その際、「回答が無い」と判断する条件を決める上で、一定時間待ち、「回答が観測されない」事を、無回答の判断規準にできる。

NSR3 がどの程度敏感に反応するかは、無反応の判断と、無反応からの回復の取り扱いによって決まる。無反応の判断とは、無回答の検出を、名前サーバ自身が反応できない状態にある、という判断に結びつけるかという規準である。また、回復の取り扱いとは、無反応と判断されている名前サーバからの回答があった場合どう判断するかという規準である。

無回答の条件としての待ち時間は、DNS の挙動に対して決めるべきである。基本的に名前サーバは 5 秒を基本にして問い合わせに対する回答を待つので、それよりも長いか短いかで動作が変わる事になる。また、無反応かどうかの判断は、保守的なアプローチを取る。つまり、名前サーバが無反応だと判断する時は慎重に、いくつかの連続した無回答を観測してはじめて無反応だと判断する。一方、名前サーバが反応したと判断するには、少数の回答を観測するだけで十分とする。

検出した状態は、IP アドレスで表現される名前サーバと、名前サーバに対応する「無反応レベル」の組で保持する。無反応レベルは、名前サーバがどの程度連続して反応がなかったかを示す値である。

5.1.4 処理の流れ

dnscap の処理では、データグラムが入力され、出力されるまでを、主に以下の 5 つの流れに分類する。

1. 未知の回答
2. 通常の問い合わせと回答
3. 通常の問い合わせと無回答、それによる代理回答
4. 無反応名前サーバへの問い合わせと即時代理回答
5. 無反応名前サーバからの回答と通信回復処理

いずれの場合でも、問い合わせを観測した段階で、その問い合わせが無反応名前サーバとして登録された名前サーバ宛でなければ、時間待ち行列に問い合わせメッセージと送受信者アドレスを格納する。また、問い合わせが無反応名前サーバ宛だった場合は、即時代理回答を行う。

これらの流れをまとめた、各問い合わせ/回答の組み合わせ毎の状態遷移図を、図 5.1 に示す。また、以下にそれぞれの流れについて述べる。

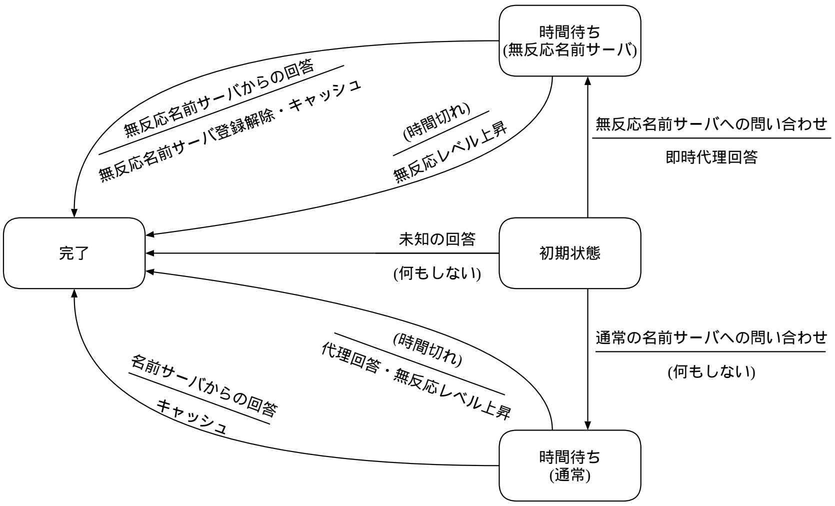

図 5.1: 各問い合わせ/回答の組み合わせに対する状態遷移図

未知の回答　回答の入力時に、時間待ち行列に対応する問い合わせが無かった場合、その回答は未知の回答であり、無視される。

問い合わせと回答　対応する問い合わせが時間待ち行列中に存在する回答メッセージを観測した時は、対応する時間待ち行列を無効化した上で、回答メッセージの内容をキャッシュに蓄える。

問い合わせと無回答、それによる代理回答　問い合わせから時間が経ち、問い合わせに対応する回答が得られなかった場合は、時間待ち行列に残っている。その問い合わせに対しては、代理回答を行った上で、問い合わせの目的ホストである名前サーバの無反応レベルを上げる。

無反応名前サーバへの問い合わせと即時代理回答　問い合わせメッセージが無反応名前サーバとして登録されていたら (無反応レベルがある閾値を超えていた場合は)、上記の通り即時代理回答が行われる。同時に、時間待ち行列に対して問い合わせメッセージを登録する際は、フラグを立てて代理回答が完了している事を示す。

無反応名前サーバからの回答と通信回復処理　もしも受信した回答メッセージの送信者アドレスが無反応名前サーバとして登録されていたら、その名前サーバの無反応レベルを一定値下げる。下げ幅は運用ポリシーを反映する。つまり、下げ幅が大きい程保守的な、つまり代理回答を抑えた運用になる。

5.2 dnscap の構成

本節では、dnscap の構成について述べる。

5.2.1 モジュール構成

dnscap の機能は、以下のモジュールに分割する。

- ネットワーク入力ハンドラ
- データ解析部
- ネットワーク出力部
- 名前データ操作部
- 無反応名前サーバデータベース部
- 時間待ち行列

モジュールおよび入出力の関係図を図 5.2 にて示す。dnscap あるいは関連プロジェクトで開発したモジュールはそれぞれ角の丸い四角形で示し、外部モジュールは二重の長方形で示している。また、矢印はモジュール間の呼び出しを表す。最下層のネットワークインターフェースとネットワーク入力ハンドラとの間には、ネットワークインターフェースの種類だけの関係が存在し、特にネットワーク入力ハンドラではデータリンク層のヘッダ解析を行っている。

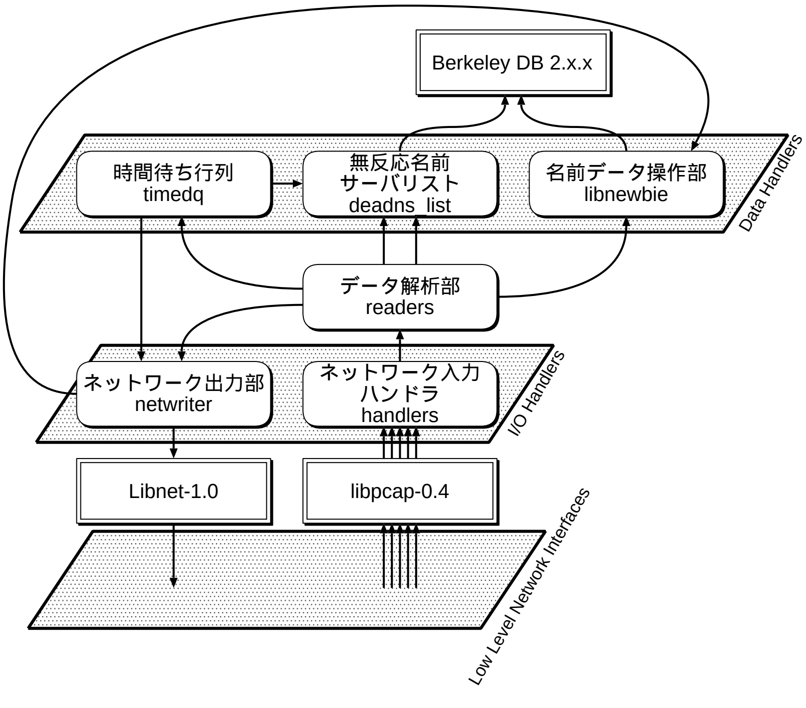

図 5.2: dnscap モジュール関係図

5.2.2 各モジュールの役割

dnscap のモジュールの役割を述べ、全体の機能の中でどのように作用するかを明確にする。

ネットワーク入力ハンドラ

ネットワーク入力ハンドラの役割は、データリンク層のヘッダと共に与えられるメッセージの、データリンク層ヘッダ部を解釈し、データリンク毎の差異を吸収する事である。

図5.2にある通り、ネットワークのメッセージは、下位モジュールのlibpcapから呼び出されたネットワーク入力ハンドラから、データ解析部に渡される。

データ解析部

データ解析部は、ネットワーク入力ハンドラから渡されたIPヘッダから始まるメッセージを解釈する。このモジュールは、入力メッセージが問い合わせか回答かによって役割が変わる。

問い合わせならば、このモジュールの役割は即時応答をすべきかのチェックと、問い合わせの時間待ち行列への登録である。まず、問い合わせを受ける名前サーバのIPアドレスを鍵にして無反応名前サーバデータベース部を参照して、無反応レベルが閾値を超えているかを調べる。もしも閾値を超えていたら、即時回答を返すためにネットワーク出力部を直接呼び出す。次に、即時回答を行ったかどうかのフラグをセットして、メッセージを時間待ち行列へ追加する。

メッセージが回答の場合は、対応する問い合わせを時間待ち行列から削除する。また、もしも回答の発信者IPアドレス、即ち名前サーバのアドレスが無反応名前サーバデータベースに登録されていたら、この回答をデータベースに反映するために、無反応レベルを下げておく。最後に、回答に含まれる情報を名前キャッシュに含める。

ネットワーク出力部

ネットワーク出力部には、データ解析部から直接呼び出される場合と、時間待ち行列が回答待ち時間切れで呼び出す場合がある。どちらの場合でも、ネットワーク出力部の役割は同じである。即ち、与えられた問い合わせに対する代理回答の作成と送出を行う。代理回答の作成のためにNSR3キャッシュを参照する。そのため、ネットワーク出力部は名前データ操作部を参照する。また、名前データ操作部は実際のDNSメッセージの作成機能を併せ持つので、メッセージの作成も名前データ操作部を利用して行う。また、低レベルネットワークインターフェースを利用した送出を行うために、Libnetモジュールを呼び出している。

なお、メッセージ作成段階で、dnscapによる回答と正規の回答を区別するために、「fingerprint情報」と呼ぶTXT(文字列)リソースレコードをメッセージの補足部に追加する。

名前データ操作部

名前データ操作部は、ネットワークメッセージ中の名前データを蓄え、検索するサービスを提供する。データベースとしての下位モジュールに、Berkeley DBを利用する。名前データ操作部は、データ解析部からデータベースの登録の形で呼び出される場合と、ネットワーク出力部からデータベース検索及びメッセージ作成の形で呼び出される場合とがある。

無反応名前サーバデータベース部

無反応名前サーバデータベース部は、反応が無い名前サーバをIPアドレスで管理し、連続して無反応だった問い合わせの数(無反応レベル)を併せて記録し、問い合わせに対して反応の無い名前サーバのデータベースを一元管理する。

無反応名前サーバデータベース部は、データ解析部から名前サーバの状態の問い合わせの形、通信回復通知(登録解除)の形、時間待ち行列の回答待ち時間切れによる無反応登録の形、の3種類の形で呼び出される。

時間待ち行列

時間待ち行列は、データ解析部から渡された問い合わせを一定時間保持し、その一定時間の間に対応する回答によって登録の無効化が行われなかった問い合わせに対して、タイムアウト処理を行う。タイムアウト処理は、名前サーバの無反応レベルの操作と、データ解析部によって即時代理回答が行われていない問い合わせに対する代理回答である。

時間待ち行列は、タイムアウト処理のために、無反応名前サーバデータベース及びネットワーク出力部を呼び出す。

5.2.3 dnscapで利用している外部モジュールについて

dnscapで利用している下位モジュールについて述べる。libpcap及びlibnetは、OSやアーキテクチャに特に依存したパケットキャプチャリングや低レベルパケット出力に関するモジュールであり、dnscapの移植性を高めるために採用した。また、Berkeley DBは高性能・高信頼性のデータベースライブラリであり、多くのDNSメッセージを処理する際に必要な性能を確保する為に採用した。

libpcap-0.4

libpcap[7]はLawrence Berkeley National Laboratoryによって開発されたパケットキャプチャリングライブラリである。libpcapは、OSの種類やアーキテクチャによって異なるパケットキャプチャリングの手法の差異を吸収する。これにより、統一されたプログラミングインターフェイスによってパケットキャプチャリングが可能となり、プログラムの移植性が高まる。

dnscapでは、libpcapによりネットワーク入力ハンドラを呼び出させている。入力ハンドラにより、入力メッセージのうちネットワークインターフェイス依存の部分が解釈された後、残りの部分(IPヘッダから先)がデータ解析部に渡される。

libnet-1.0

libnet[33]は、Mike D. Shiffman `<mike@infonexus.com>`が開発した、アーキテクチャ毎に低レベルなパケット出力を行うライブラリである。libnetは、OSの種類やアーキテクチャによって異なる低レベルパケット出力の手法の差異を吸収する。統一されたプログラミングインターフェイスによって低レベルパケット出力が可能となり、プログラムの移植性が高まる。

dnscapでは、遠隔の名前サーバの反応が無い場合呼び出されるネットワーク出力部で、出力す

べきメッセージを作成した後にそのメッセージを実際のネットワークインターフェイスに出力するために利用している。

Berkeley DB

Berkeley DB[8] は現在 米国の Sleepycat Software で開発が続けられているデータベースライブラリである。いろいろな方式でのデータベースの作成やアクセスが行える上に、sendmail 等の有力アプリケーションで長い間利用されて来たため、信頼性が高い。

dnscap では、無反応名前サーバデータベース部のデータベース、及び、名前データ操作部での名前キャッシュデータベースに利用している。

第6章　評価

本節では、NSR3による名前解決系の変化について、分析・評価した後に、dnscapを利用したNSR3の実証実験の結果についてまとめ、dnscapが名前解決障害時に作動して、周囲の障害を隠蔽する効果があった事を示す。

6.1　NSR3による名前解決系の変化

本節では、NSR3により名前解決系がどのように変化したか評価する。

6.1.1　名前解決系の性質変化

dnscapの利用による名前解決系の性質変化を以下にまとめる。

- 定常時の動作
  定常時の動作は、基本的に何も変更が無い。従って性質は既存の名前解決系の性質を全てそのまま受け継ぐ。
- 性能
  リゾルバの段階でdnscapを適用する以上、性能は劣化する事が予測できる。なぜならばdnscapはインターフェイスで観測できるパケット全てを観測しようとするからである。
- 障害時の動作
  遠隔の名前サーバの障害を確認したら、dnscapは障害を隠蔽しようと試みる。その試みが成功すれば、NSR3名前キャッシュに含まれている情報が古く腐っている場合を除き、dnscapが動作している地点からは障害を検知できなくなる。すなわち、名前解決系が遠隔名前サーバも含めて正常に動作しているかのように振舞う。

NSR3は、DNSが正常に動作している限りはDNSに影響を与えない。従って、定常状態では従来のDNSと何ら変わらない動作を行う事が期待できる。

dnscap/NSR3が効力を発揮するのは、特定の名前サーバが利用できなくなった場合である。以下の条件が成立した場合、dnscap/NSR3は、通常のDNSでは解決できない名前を解決する。

- ある名前dに対して直接権限を持つサーバへ到達可能である
- 名前木の根から名前dへ至る段階のうち、最下層(名前に対して直接段階に権限を持つサーバ)以外のどれかが利用不可能である、あるいはRTTが異常に長い
- dnscapを導入してから少なくとも一度、名前dを解決した事がある

上記の条件が成立する可能性については第3章にて述べた。

6.1.2 NSR3の効果の指標による予測

節3.1.2「指標c: DNS広域依存問題の指標」で示したように、M_cはDNS広域依存問題の起こりやすさの指標である。dnscapによりNSR3を適用した名前解決系では、M_cがどのように変化するかを以下に示す。

ここで、NSR3に関して$t \to \infty$とする事で、NSR3名前キャッシュのキャッシュヒット率を定数R_cで近似する。NSR3名前キャッシュは通常のキャッシュと異なり、名前解決の過程で得た遠隔名前サーバの情報(NS RR)を破棄しないので、$t \to \infty$とする事で、新しい名前解決で得られる情報が既知の情報と異なる確率は0に近付き、R_cは変化しなくなる。

厳密には、NSR3に関してもう一つ係数を導入する必要がある。その係数とはキャッシュの正誤率である。当然、NSR3キャッシュの内容と実際の名前サーバが食い違う危険性は存在する。しかし、ここでは、本稿では名前サーバの変更は滅多に行われないという仮定(節4.2.3)に従い、キャッシュは常に正しい情報を含んでいるものとする。また、変更が頻発する名前サーバに対する対応はNSR3の枠組の外で行われるべき機能であり、ここでは考慮しない。

さて、上記の条件を考慮して、M_cを元にdnscap/NSR3適用時の指標を導出する。NSR3名前キャッシュに情報が有る場合は名前サーバが利用できなくても名前解決が可能である。ここで、各名前レベルの名前サーバ利用可能性に対して定数$1 - R_c$、つまりキャッシュミス率を掛ける事で、この特性を指標に繰り入れる。

$$M_{NSR3} = M_c \cdot (1 - R_c) \tag{6.1}$$

もしもNSR3名前キャッシュに対するアクセスが理想的な場合、$t \to \infty$とした時に$R_c \to 1$となる。新規に登場する名前は次第に0に近づくので、理想状態では上位名前サーバが利用できなくとも、名前解決には影響しない事が予想できる。

つまり、**NSR3**の理想状態では、「**DNS**広域依存問題」は発生しない事が予想できる。

6.2 dnscapの実働実験

本節では、実働実験から判明したdnscapの効果について述べる。

1999年12月11日、1999年12月22日の両日にdnscapの実働実験を行った。12月22日は`jp.`ドメインのセカンダリ名前サーバでもある`jp-gate.wide.ad.jp.`というマシンの一時停止があった。12月22日の実験では、その停止による影響を観察した。実験環境を表6.1に示す。

表6.1: dnscap実験環境

ハードウェア	Sun Ultra30 (sun4u)
OS	SunOS 5.6
ネットワーク	1000Base-T
周辺環境	gcc 2.8.1 libpcap-0.4 libnet-1.0 Berkeley DB 2.7.7

6.2.1 テストケースの解決による動作の検証

ドメイン名`test-a.sub.doitest.sfc.wide.ad.jp.`を、ドメイン名広域依存問題のテストケースとして設定した。以下テストケースドメイン名をd_tと表記する。d_tの名前空間構造を図6.1に示す。

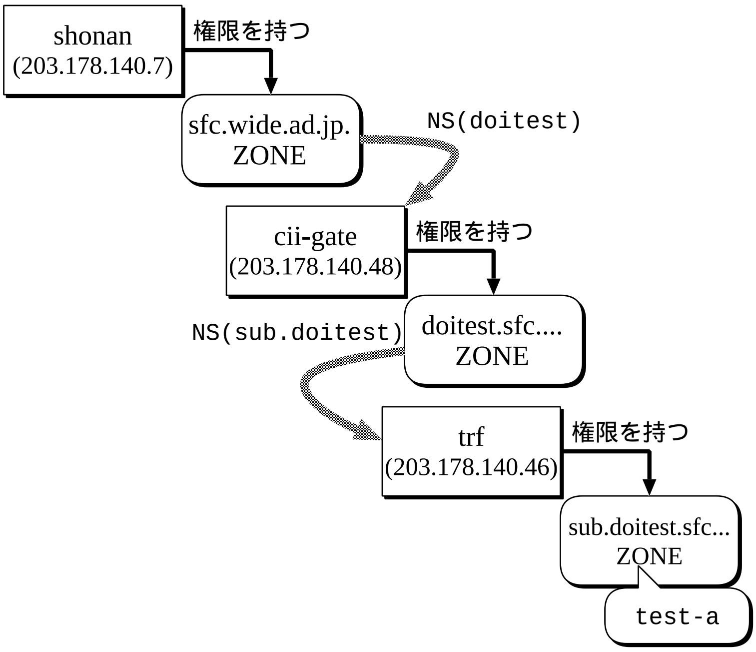

図6.1: テストケースドメイン名の名前空間構造

図6.1は、ZONEで示される名前空間のゾーン(図中の丸い長方形)とゾーンに権限を持つ名前サーバ(図中の長方形)及びそれらの関係(矢印)を示したものである。但し、`sfc.wide.ad.jp.`ゾーンに関してはプライマリ名前サーバのみを示している。ここで、関係とは上位ゾーン内に記述されたNSリソースレコードによるゾーンと下位名前サーバの関係、そして権限(authority)を持つことによる名前サーバとゾーンの関係の2つである。

この構造の狙いは、2つの名前サーバを経由しないとテストケースドメイン名を解決できない点にある。ここで、`cii-gate`上の名前サーバを制御する事で、中間名前サーバが利用できない状態(「障害状態」)及び障害状態からの復帰を再現できる。

12/11の実験中、以下の手順でテストケースの解決を行い、dnscapがd_tに対して正常に機能する事を確認した。dnscapは図6.1上の名前サーバ`shonan`上で動作しており、`shonan`はリゾルバとしての機能を併せ持つ。また、d_tおよび関連するレコードのTTLは1秒に設定したので、キャッシュの影響を排除するために各動作の間には1秒以上の間隔を空けた。

1. 初期状態(dnscap停止)
2. `cii-gate`の名前サーバを停止(障害発生)
3. d_tを`shonan`を利用して解決。正しく解決できない事を確認
4. `cii-gate`の名前サーバを再起動した上で、dnscapを`shonan`で起動
5. d_tを`shonan`を利用して解決。正しい情報が得られる事を確認
6. `cii-gate`の名前サーバを停止(障害発生)

表 6.2: 11 日の実験の統計結果

入力メッセージ (`io.inmesg`)	160016
代理応答 (`io.outmesg`)	512
処理待ち (`io.waiting`)	0
内部エラー (`err.internal`)	5
truncate(`err.trunc`)	0
キャッシュミス (`err.cacheMiss`)	0
不正な形式 (`err.invalidFormat`)	0
未知の形式 (`err.unknown`)	0
無反応名前サーバ (`ns.dead`)	114
データベースエラー (`ns.dberr`)	0

7. d_t を `shonan` を利用して解決
8. 手順 7 を数度繰り返し、結果を確認
9. `cii-gate` の名前サーバを起動 (障害回復)
10. 手順 7 を数度繰り返し、結果を手順 8 と比較

試験の結果、手順 7 では dnscap はまだ代理応答を返さず、先にリゾルバライブラリがタイムアウトする事を確認した。しかし、その過程でリゾルバライブラリが再試行を行い、結果として dnscap が `cii-gate` を無反応名前サーバデータベースに追加した (これはログから確認)。無反応名前サーバへの `cii-gate` の追加後は、dnscap の即時代理応答により `sub.doitest` の名前サーバが `trf` である事を示す NS RR が `shonan` に与えられる事を確認した。その後、リゾルバライブラリが d_t を解決して、d_t が指し示す IP アドレス (この場合は `127.0.0.1`) にクライアントプログラムが接続 (`connect(2)`) する事を確認した (図 6.4)。

6.2.2 動作ログ及び dnscap 統計情報からの分析

12 月 11 日の実験では、14 時 07 分にログの取得が開始されて、19 時 54 分に実験が終了するまでの統計情報を表 6.2 に示す。ここで、代理応答は、入力メッセージに対して約 $\frac{3}{1000}$ であり、名前解決系としては概ね問題無く動作していた様子である。代理応答の発生と問い合わせ数の推移を図 6.2 に示す。

図 6.2 は、横軸に実験開始からの秒数 (最終的に 20816 秒)、縦軸に問い合わせ処理数 (“# of received queries”) と代理応答処理数 (“# of answers out”) を取っている。但し、比較のために代理応答処理数を $\frac{160016}{512}$ 倍して、見掛け上の最終的な位置が一致するように変更している。

一方、12 月 22 日の実験は、12 月 11 日の実験と異なり、WIDE 藤沢 NOC に存在する名前サーバ `jp-gate` がネットワークから切り離されており、また同時に経路が不安定になっている状況での実験である。22 日の実験の統計結果を表 6.3 に、また受信メッセージと代理応答数の推移を図 6.3 に示す。図 6.3 は、図 6.2 と同様、横軸に経過時間 (7731 秒)、縦軸に問い合わせ処理数 (“`# of received queries`”) と代理応答処理数 (“`# of answers out`”) を取っている。但し比較のために、代理応答処理数は $\frac{70458}{239}$ 倍している。

22 日の実験では、入力メッセージに対して代理応答が $\frac{4}{1000}$ 程度存在した。表 6.2 に対する表 6.3 での入力メッセージに対する代理応答の比を求めると、1.27 倍程度である。また、図 6.2 と図 6.3

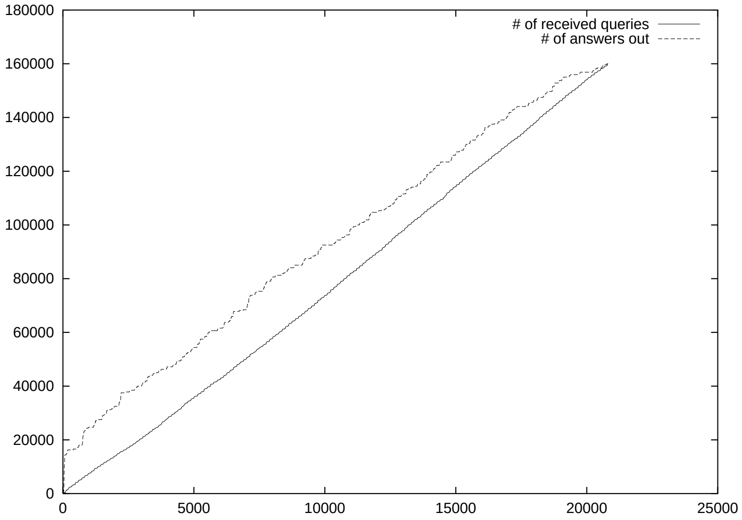

図 6.2: 11 日の問い合わせ処理数と代理応答処理数

表 6.3: 22 日の実験の統計結果

入力メッセージ (`io.inmesg`)	49415
代理応答 (`io.outmesg`)	201
処理待ち (`io.waiting`)	0
内部エラー (`err.internal`)	2
truncate(`err.trunc`)	1
キャッシュミス (`err.cacheMiss`)	0
不正な形式 (`err.invalidFormat`)	0
未知の形式 (`err.unknown`)	0
無反応名前サーバ (`ns.dead`)	64
データベースエラー (`ns.dberr`)	0

を比較すれば明らかなように、近隣のネットワークが不安定な場合、dnscap の挙動を示す代理応答処理数の変動が大きくなっている。これはつまり、近隣のネットワークに不具合や利用不可能な名前サーバがある場合に、名前解決系が不安定になっている様子を示している。

代理応答の増加、および代理応答処理の挙動は、dnscap がより活発に活動している事を示す。つまり、これらの結果は、dnscap が周囲の障害に対応して、名前解決の障害を隠す効果をあげている証拠である。

6.2.3 ネットワークの観測結果からの動作解析

dnscap を動作させた時の通信ログを分析し、意図した動作をおこなっている事を確認する。また、意図しない動作や想定外の影響が無い事を確認する。

実働実験中取得した通信ログは、12 月 11 日の実験では 15 時 30 分より 19 時 55 分までであり、12 月 22 日の実験では 12 時 00 分より 14 時 08 分までである。ここで、通信ログは `tcpdump` により

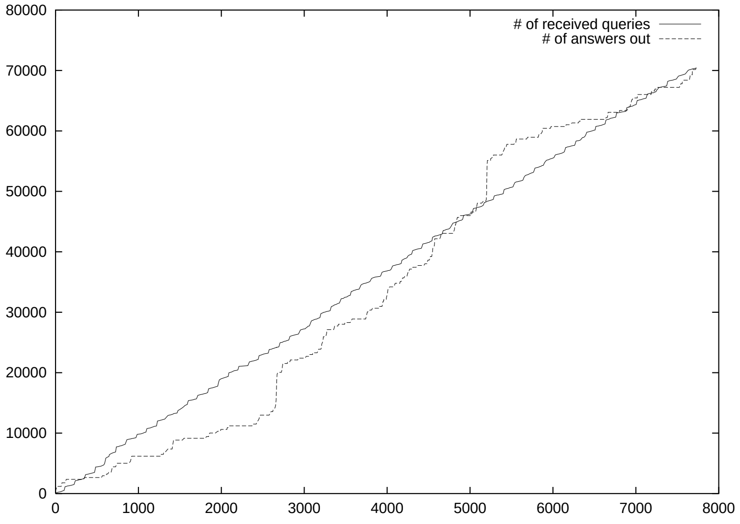

図 6.3: 22 日の問い合わせ処理数と代理応答処理数

取得したものを、`dnsdump`(付録 B) により解析している。dnscap による代理応答には、fingerprint 情報が埋め込まれている (節 5.2.2)。ここでは、取得した通信ログ中の fingerprint 情報を検索することで、代理応答を発見する。

図 6.4 はテストケース: t_d に関する試行のログである。一番上のブロックでは正しい手順で再帰問い合わせが行われる事を確認している。中央のブロックでは `203.178.140.48` の名前サーバ停止して、正しく代理応答を得られる事を確認している。一番下のブロックでは、`203.178.140.48` の名前サーバを再起動して、正しく定常状態に復帰する事を確認した。

図 6.5 は 22 日に観測されたトラフィックである。あるゾーンの名前サーバに対して全くアクセスができない状態になり、リゾルバサーバが可能な全ての名前サーバに対して問い合わせを行っている。最終的に dnscap による代理応答が観測されている。

図 6.6 の例では、`210.154.149.35` 上の名前サーバが応答しない状態であった。そこでリゾルバサーバは同じドメインに対するもうひとつの名前サーバ (`192.50.77.195`) に対し問い合わせを行い、回答を得た。最後に dnscap が、`210.154.149.35` の代わりに代理応答を行っている。これは、NSR3 名前キャッシュの前に、障害を持っているサーバとは別の名前サーバが利用できたので、そちらの情報が優先された例である。

図 6.7 の例は、正しく動作しない名前サーバに対して dnscap が作用した例である。図の上で最も右側の `129.117.147.130` の名前サーバが、何らかの理由で名前問い合わせに対する回答メッセージの送信者 IP アドレスに不正確な IP アドレスを使っている。`203.178.140.7` 上のリゾルバサーバは送信者アドレスを厳密にチェックするので、その回答を受け取る事ができない。そこで、dnscap が持っていた情報を元に代理応答を行った。

6.2.4 リゾルバサーバの問い合わせ処理性能の変化

パケットの「覗き見」による性能の低下を明確にするために、dnscap が動作している時としていない時の、名前サーバの問い合わせ処理にかかる時間をホスト A(Solaris 5.5.1/sparc/Gigabit Ether) およびホスト B(FreeBSD 2.2.8/i386/100BASE-T) で計測した。計測の条件は以下の通り。

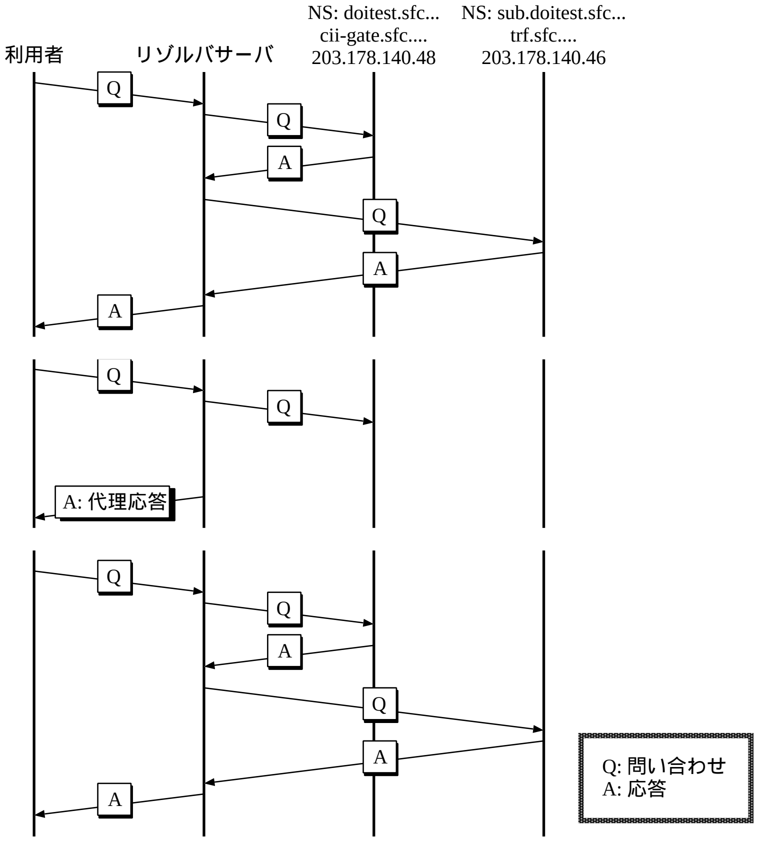

図 6.4: テストケースにおける動作

- 再帰問い合わせはしない: 再帰問い合わせにかかる時間は不確定であり、これによる誤差は名前サーバの性能の劣化に比べてはるかに大きい。
- 同じ物理セグメントの他のホストからの時間計測: 問い合わせを送信し、回答を受信した直後に次の問い合わせを送信するという動作を 1000 回繰り返す。

以上の条件で計測した結果の、問い合わせ送信から回答受信までかかった時間の平均を表 6.4 に示す。表を見ると、B は A に比べより大きな性能劣化を示している。

表 6.4: dnscap 動作時と非動作時での問い合わせ応答性能

ホスト	dnscap 有り (A)	dnscap 無し (B)	性能劣化 $(\frac{A-B}{B})$
A	0.001912	0.001800	0.0622
B	0.001075	0.000898	0.1971

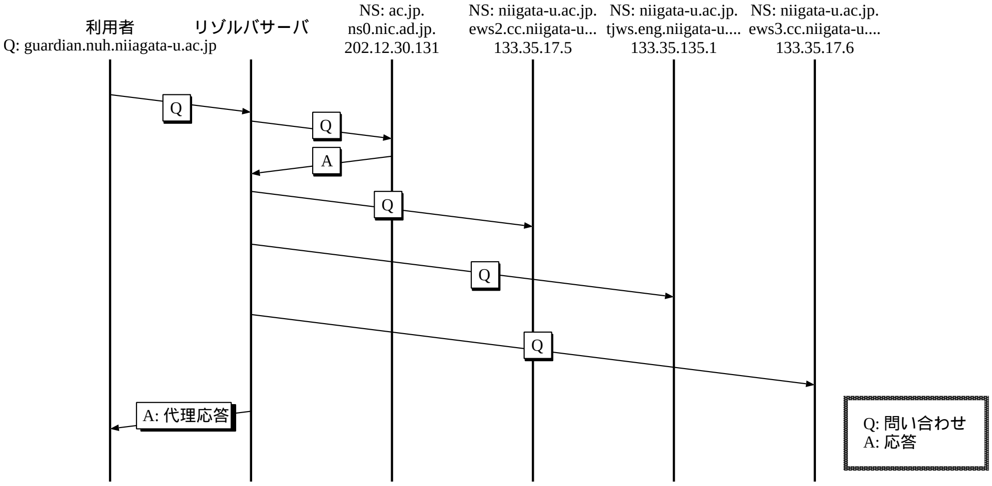

図 6.5: 意図した動作

6.2.5 実験のまとめ

実験の結果、テストケースに関して正しく名前解決が行われる事を確認した。また、テストケースに関しては、`telnet` 等のクライアントプログラム上のリゾルバライブラリが、dnscap による代理応答により DNS 広域依存問題を回避する様子を確認した。同時に、通信ログの解析により、dnscap によって通信に悪影響は生じてない事を確認した。また、名前サーバの障害に対して、期待した通りに代理応答を返す事を確認した。

また、ネットワーク安定時 (11 日) 及び不安定時 (12 日) を比較し、dnscap の働きに、1.27 倍という明らかな差がある事が判明した。その結果、dnscap は障害対策として有効に動作している事が確認できた。

dnscap による性能劣化が観測された。しかし、実験環境である Solaris/sparc/1000BASE-T の組み合わせにおいては、性能劣化による問題は生じなかった。

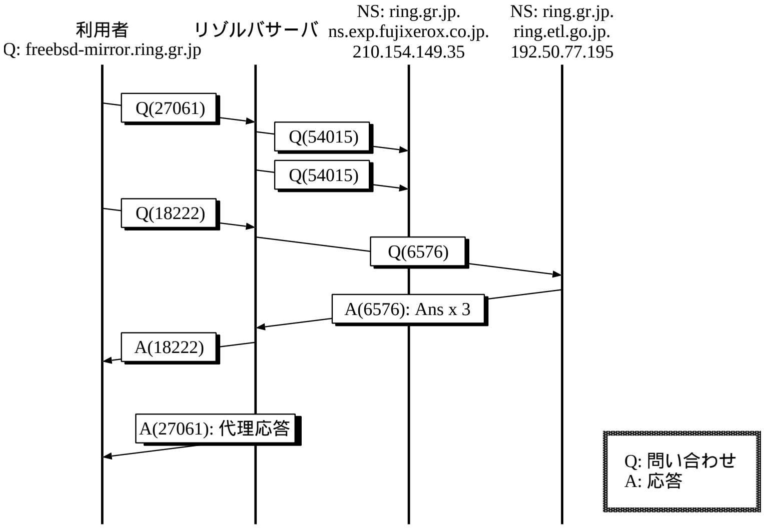

図 6.6: 動作例 – retry

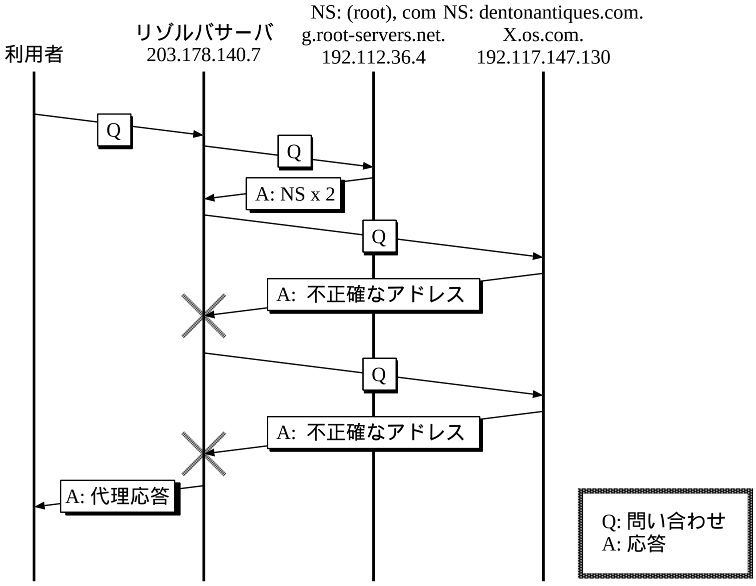

図 6.7: 動作例 – 不正確な IP アドレス

第7章　考察

本章では、dnscap/NSR3 に伴ういくつかの検討課題について考察する。

7.1　NSR3 適用環境における DNS の障害検出と復旧

dnscap/NSR3 は DNS の障害を隠す。節 6.2.3 にもその例がある。ユーザの立場からすれば、dnscap/NSR3 の効果により、障害に気がつかないままに通信の目的ホストに接続できるので、なんら問題は発生しない。しかし、障害を解決しなければならない管理者の立場からみると、dnscap の存在は障害の特定を困難にする可能性がある。

dnscap はそのために、管理者向けの情報を残す必要がある。節 5.2.2 で述べた Fingerprint 情報はその一つである。つまり、dnscap による代理応答には、dnscap による代理応答である事を示す TXT リソースレコードを付加する。

DNS の障害を追跡する場合は、`dig` や `nslookup` といった専用のクライアントソフトウェアを用いる事が多い。これらのソフトウェアを利用すると、DNS メッセージ中の Fingerprint の存在を見つける事ができる。また、これらのソフトは利用するリゾルバサーバを任意に選択できるので、Fingerprint 情報によって dnscap の影響が明かになれば、dnscap に影響されない遠方のリゾルバサーバを利用する事で障害解決を容易にする。

従って、dnscap の運用を行う管理者は、DNS の障害追跡にあたって以下の点に注意すれば良い。

1. Fingerprint 情報
2. リゾルバサーバの選択

7.2　IP ソースアドレス偽造とスイッチデバイス

dnscap は実装上、DNS 通信への介入手段として、IP ソースアドレスの偽造を選択した。一方、IP ソースアドレスの偽造はセキュリティの観点から危険な行為とされる。従って、一部のスイッチデバイスや防火壁等のネットワーク装置が、ソースアドレス偽造を検出してパケットを転送しない可能性がある。

この問題は、NSR3 の実装方法を named の改造や Y-splitter の利用といった手法に変更する事で根本的に解決できる。一方、Stealthy Cache の技法を利用する場合は、Stealthy Cache の設置場所と実装に注意する必要がある。基本原則は「どの空間をどの空間に対して偽造するか」を明確にする事である。

IP ソースアドレスの偽造はコミュニケーションへの介入手段である。このコミュニケーションとは、リゾルバサーバとサイト外の名前サーバとのコミュニケーションである。従って、Stealthy Cache の設置には以下の 2 形式が考えられる。

1. リゾルバサーバ中に設置

2. サイト境界ルータに設置

形式 1 は、リゾルバサーバ単位での NSR3 技法の適用である。節 6.2 では、形式 1 による実験を行った。この形式の場合、IP アドレス偽造はリゾルバサーバに対してのみ行われる。つまり単独のホストの内部で発生する事象であり、ネットワーク機器はこれを関知しない。これは、リゾルバサーバ外の名前空間をリゾルバサーバに対して偽造する事で、障害を隠蔽する方法である。

また、サイト境界ルータへの設置する場合 (形式 2) は、境界のどちらの IP アドレスをどちらに対して偽造するかに注意する必要がある。サイトの境界の内側に向うコミュニケーションについて「のみ」外側の IP アドレスを偽造する事が許される。

この条件は、以下の 2 点より導かれる。dnscap が影響すべき空間はサイト内である点と、サイト内の全てのリゾルバサーバに対して、DNS の障害を隠蔽し NSR3 名前キャッシュに蓄えた名前空間を代りに示す事を目的とする点である。つまり、方法 2 は、サイト外の名前空間をサイト内に対して偽造する方法である。

7.3 dnscap を利用した攻撃

以下のような dnscap を利用した攻撃が考えられる。

1. サイト X は、なんらかの手段でサイト A の dnscap に、「サイト B の名前サーバはサイト X にある」という偽情報を記憶させる。
2. サイト B の名前サーバをサービス拒否攻撃 (Denial-of-Service Attack) 等で沈黙させる。
3. サイト A からのアクセスが dnscap の偽情報に従ってサイト X に導かれる。

NSR3 をこのような攻撃から守るためには、攻撃の第 1 段階で防がなければならない。つまり、ネットワークを流れている情報のそれぞれに対して信憑性の評価を行う必要がある。

信憑性の評価には以下のような手段が考えられる。

1. 名前空間木の権限委譲を信頼
2. 出現パターンを観測
3. DNSSec を利用

手段 1 は、名前空間の根、あるいは任意の節に対応する名前サーバを信頼できる既知の知識として持つ。その上で、それら信頼できる名前サーバから権限を委譲された名前サーバのみを、信頼する名前サーバに繰り入れていく方法である。

この手法は、キャッシュした名前空間木が十分な大きさに成長するまでは NSR3 名前キャッシュが十分な効果を持たないという欠点がある。また、実際の名前空間木の構成変更をどのように検出して、どのようにキャッシュ上の名前空間木に反映させていくかという問題も残る。しかし、権限委譲に基づいた名前空間木の構成を行えば、DNS 本体とほぼ等しい信憑性を得る事ができるという利点がある。

手段 2 には、侵入探知 (ID: Intrusion Detection)[13] の手法を応用できるだろう。ID は、攻撃を意図したパケットには一定のパターンが存在するという仮定の下、そのパターンを検出する技術である。もしも、攻撃者の取り得る手段が明らかになれば、疑わしい DNS メッセージの情報はキャッシュしないという選択が可能になる。単純な例では、DNS の枠組の中で再帰問い合わせと回答という枠組が存在する。その際、問い合わせとの対応関係が無い回答や、DNS メッセージに不正があるメッセージを検出する事は可能である。

手段 3 は、DNSSec を認証のために利用するリゾルバサーバにおいてのみ利用できる。既に DNSSec を利用できる BIND の実装を基礎に、NSR3 機能を実装した安全なリゾルバサーバを作る事が可能である。

7.4 より広範囲な応用

NSR3 の応用範囲を広げる事には一定の意義を見いだす事ができるが、そのためにはいくつかの困難な問題を解決する必要がある。

NSR3 の応用範囲を例えば、「日本全体」に広げたとする。すると、国外の名前サーバの不調のほぼ全てが dnscap によって隠蔽される。これによって、国内からの名前解決の信頼性は飛躍的に上がる。なぜならば日本国内での通信の障害可能性のみにしか影響されないからである。

以下の説明で、「境界が定義できる任意のサイトの集合」を「クラスタ」と定義する。

日本国内というクラスタの「定義」の問題を度外視して、ある一定の組み合わせの「クラスタ境界ルータ (CBR)」が定義できたとしても、依然問題は残る。節 4.3.1 でも述べたが、ネットワークトラフィックの出入口が複数存在する場合、NSR3 は現状のままでは適用できない。インターネットの経路は非対称なため、問い合わせと応答が各地点で正しく対応付けられないからである。

もしも CBR が十分に少なく、かつそれぞれの CBR が近傍に存在すれば、協調型 NSR3 を考える事ができる。協調型 NSR3 の一例には以下のような物が考えられる。

図 7.1 に示すように、NSR3 エッジがそれぞれの CBR 上に存在し、クラスタ内の中央部に一つの NSR3 コアが存在する。NSR3 エッジ上で DNS メッセージの出入りを観測し、一定の粒度で NSR3 コアに情報を集約する。NSR3 コアは全ての NSR3 エッジ上の情報から外界の無反応名前サーバのリストを作成し、NSR3 エッジに送る。NSR3 エッジはリスト上にある名前サーバへの問い合わせのみに対して代理応答を行う。

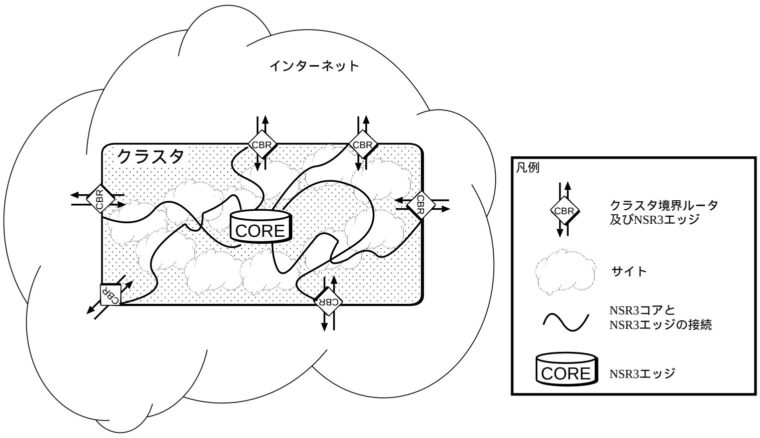

図 7.1: 協調型 NSR3

しかし、大きなネットワークに対して協調型 NSR3 を適用した場合、CBR のリストをきちんと管理しなければならない。上記の方法では、CBR リストが完全でないと正常な運用ができないので、クラスタに含まれるサイト (管理主体) の数が多くなればなるほど現実性が無くなる。

また、無反応名前サーバ検出に一定の知識が適用できれば、非協調型 NSR3 をクラスタ全体に適用できる可能性がある。例えば、リゾルバサーバ、あるいはリゾルバライブラリの問い合わせ

で、反応が無ければ一定時間置いて再び問い合わせるという性質が普遍的性質だとすると、そのパターンを観測する事で反応が無い事は検出できる。問題は、問い合わせに代理応答を返してしまった後、他の経路から本来答えるべき名前サーバからの回答が来ても検出できない点である。

第8章　結論と今後の展望

本章では、本研究の結論として、問題の解析結果と解決に関してまとめ、将来の展望について述べる。

8.1　結論

本研究では初めに、ネットワーク層は接続可能な近傍のネットワークでも、名前解決に失敗すると接続できない問題から、DNSが広域の名前サーバに依存している問題『DNS広域依存問題』の存在を明らかにした。この問題に対し、現在の名前解決系の観測結果及び指標から、様々な事例を分析した。同時に、分析結果とDNSの特性に着目して、問題を解決するための手法NSR3を提案した。

また、評価において、NSR3によって問題が解決すれば、理想的な状態では『DNS広域依存問題』が発生しなくなる事を、モデルと指標から示した。これは、名前空間木が切断されて、切断点から先が利用できなくなる状況に対して、NSR3が効果的に作用する事を示したものである。

NSR3の動作検証のため、dnscapを実装した。実環境における実験結果により、dnscapがDNS広域依存問題を吸収し、解決している事が確認できた。具体的には、障害時は定常時の約1.27倍の比率で代理応答を行っている事による。つまり、ネットワーク障害時と定常時を比較すると、障害時の方が明らかに代理応答の件数が多く、これは周囲のネットワークの障害に対してdnscapが作動し、障害を隠蔽している事を示す数値である。

以上、モデルと実環境での実験での両面から、本研究で提案したNSR3の効用を証明した。

8.2　DNS研究に関する今後の展望

NSR3は、DNSの通信が不安定になる場合に備えて、特殊なキャッシュを確保しておく技術である。仮にインターネット自体が大きく、そして深くなりすぎた時に、NSR3とその周辺技術がインターネットの信頼性向上のために必要になる。

今回の研究で提案した指標は、NSR3の実用性を証明するのには十分であった。しかし、問題の分析に関しては、複数の名前サーバでの共有経路の考慮等、指標をより現実の問題に近づける余地がある。より正確な指標が導ければ、リゾルバサーバ側の立場から名前解決系を評価でき、DNSそのものの評価に貢献できる。また、多くの観測点の指標を合成する事で、名前解決系の問題点や根本的な障害対策に貢献できる。

今後、NSR3に関する様々な問題の解決を含めて、DNSそのものに関しての分析や研究が少ないため、本研究を継続して行う必要がある。

謝辞

本研究を行なうにあたり、様々な面からの御指導と御支援を頂きました慶應大学環境情報学部教授 村井純博士に感謝します。同時に、お忙しい中多大な御指導を頂きました、慶應義塾大学環境情報学部教授 徳田 英幸博士と慶應義塾大学環境情報学部助教授 萩野 達也博士、分析手法等の面で温かい助言を頂きました、慶應義塾大学環境情報学部教授 西岡 啓二博士、日頃から研究に関する多くの助言、叱咤激励を頂きました、慶應義塾大学環境情報学部の楠本 博之博士、中村 修博士に感謝します。

本研究を通じて丁寧に御指導頂いた、慶應大学政策 ·メディア研究科の渡辺 恭人氏、石橋 啓一郎氏、関谷 勇司氏、株式会社東芝 研究開発センターの南 政樹氏、及び、研究初期の方向づけに関して相談にのって頂き、迷いに手をさしのべて頂いた、株式会社東芝研究開発センターの石山 政浩氏をはじめとするWIDEプロジェクトメンバーの方々に感謝します。

本研究の基礎となるアイディアを与えて頂き、また議論の中から様々な知見を与えて頂いた、NTT移動通信網株式会社 ワイヤレス研究所の岡島 一郎氏、並びにStanford University Assistant Professor Mary G. Baker博士をはじめとする、Stanford University MosquitoNet Groupの皆さんに感謝します。また、新しいアイディアを常に厳しい目で評価して頂いた、慶應義塾大学村井研究室のメンバー、慶應義塾大学大学院 政策 ·メディア研究科のモービル広域ネットワークプロジェクトの皆さんに感謝します。

また、執筆中に精神の潤いとゆとりを与えてくれた、スヌーピーとその仲間達、そして彼らの生みの親であるCharles Schulz氏に感謝します。

最後に、ありとあらゆる面から私を助け、支えになってくれた、妻 折田 明子をはじめとする家族の皆に感謝します。

尚、本研究は、1999年度森泰吉郎記念研究振興基金(研究育成費)の支援を受けて行なわれた事をここに記すと同時に、その支援に対して感謝します。

平成12年1月17日
土井裕介

参考文献

[1] 土井裕介, *“An Analysis of the Name Cache Effect on the Nomadic Computing”*, インターネットコンファレンス’98 論文集, 日本ソフトウェア科学会研究会資料シリーズ No.9(ISSN 1341-870X), 1998年12月, pp.11-pp.20

[2] 土井裕介, 「次世代ネットワーク環境をサポートする名前サービスの実現方法に関する研究」, 慶應大学環境情報学部卒業論文 (村井研究室), 1998年3月

[3] 尾上 淳, 「広域開放型分散環境における名前サーバの設計と実装」, 慶應義塾大学大学院 平成元年度修士論文

[4] Peter B. Danzig, Katia Obraczka, and Anant Kumar, *“An Analysis of Wide-Area Name Server Traffic”*, ACM Computer Communication Review 22(4), Oct. 1992

[5] Craig Labovits, G. Robert Malan, and Farnam Jahanian, *“Internet Routing Instability”*, Proceedings of ACM SIGCOMM 97, September 1997

[6] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, *“The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”*, ACM SIGCOMM, volume 27, number3, July 1997

[7] Steven McCanne, Van Jacobson, *“The BSD Packet Filter: A New Architecture for User-level Packet Capture”*, Proceedings of USENIX Winter Conference, January 1993

[8] Margo Seltzer, Ozan Yigit, *“A New Hashing Package for UNIX”*, Proceedings of USENIX Winter Conference, January 1991

[9] Douglas B. Terry, Mark Painter, David W. Riggle, and Songnian Zhou, *The Berkeley Internet Name Domain Server*

[10] 井庭 崇, 福原 義久, 「複雑系入門 知のフロンティアへの冒険」, NTT 出版, 1998年6月

[11] 郵政省電気通信局・監修, 「ドメインネームの全て」, インターネット・ドメインネームに関する研究会・著 財団法人 日本データ通信協会 編, クリエイト・クルーズ, 1998年9月

[12] 「商用ネットワークサービスプロバイダー接続マップ」, Internet Magazine 2000年2月号 付録, 株式会社インプレス, 2000年1月

[13] *“Special Issue: Instrusion Detection”*, USENIX `;login:`, September 1999

[14] Paul Albitz, Cricket Liu, *“DNS and BIND”*, O’Reilly & Associates, INC., September, 1998

[15] Hal Stern, *“NFS & NIS”*, O’Reilly & Associates, INC., July 1991

[16] 「*RT200i/RT140[iefp]/RT103i Rev.4.00.18* リリースノート」, `http://www.rtpro.yamaha.co.jp/RT/docs/relnote/Rev.04.00/relnote_04_00_18.txt`, ヤマハ株式会社, 1999 年 7 月

[17] Tim Vann, William McFall, Mark McNeill, and Duke Power, *An Introduction to BIND Source Code*, IBM Unclassified Documents

[18] 萩野純一郎, 「*itojun.org*」, `http://www.itojun.org/`

[19] 水島広子, 「コミュニケーション正常化作戦」, `http://www.mizu.nu/`

[20] *"Generic Top Level Domain Memorandum of Understanding"*, `http://www.gtld-mou.org/`

[21] *"Internet International Ad Hoc Committee"*, `http://www.iahc.org/`

[22] *Internet Software Consortium*, `http://www.isc.org/`

[23] *IETF Domain Name Server Operations Working Group*, `http://www.ietf.org/html.charters/dnsop-charter.html`

[24] G. Catalone, R.Rockell, *"Implementation of a High Availability DNS System"*, Internet-Draft `draft-catalone-rockell-hadns-00.txt`, work in progress

[25] M. Ohta, E. Hardie, *"Distributing Root Name Servers via Shared Unicast Addresses"*, Internet-Draft `draft-ietf-shared-root-server-01.txt`, work in progress

[26] Jonathan B. Postel, *"Simple Mail Transfer Protocol"*, IETF Request For Comments: 821, August 1982

[27] T. Berners-Lee, L.Masinter, and M McCahill, *"Uniform Resource Locators (URL)"*, IETF Request For Comments: 1738, December 1994

[28] P. Vixie, *"A Mechanism for Prompt Notification of Zone Changes"*, IETF Request For Comments: 1996, August 1996

[29] D. Eastlake 3rd, C. Kaufman, *"Domain Name System Security Extensions"*, IETF Request For Comments: 2065, January 1997

[30] B. Manning, P. Vixie, *"Operational Criteria for Root Name Servers"*, IETF Request for Comments: 2010, October 1996

[31] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, *"Dynamic Updates in the Domain Name System"*, IETF Request For Comments: 2136, April 1997

[32] *"tcpdump 3.4"*, Lawrence Berkeley National Laboratory, Network Research Group, `tcpdump@ee.lbl.gov`

[33] Mike D. Schiffman, *"Libnet 1.0"*, `http://www.packetfactory.net/libnet/`

付 録A　DNS: Domain Name System

A.1　Domain Name System

A.1.1　ドメイン名とDNSの役割

現在のインターネットアーキテクチャにおける「名前」、即ちドメイン名の役割の一つは、コミュニケーションの終端識別子である。

一方、インターネットプロトコルの終端識別子は、IPアドレスである。IPアドレスとドメイン名を結びつけるために、インターネット上に分散している名前サーバを用い「名前解決」を行う。このサーバ群を含むシステム全体を“Domain Name System(DNS)”と呼ぶ。

ドメイン名は、アルファベット、数字、一部の記号から成り立つ。アルファベットの大文字と小文字は同一視される。また、記号には-(ハイフン)のみが許されている。ドメイン名は木のように分岐する階層構造を持っており、その階層の区切りは.(ピリオド)で表記される。

ドメイン名の表記は、`www.sfc.keio.ac.jp.`や`www.icann.org.`の様に行う。末尾に.(ピリオド)が付くドメイン名の表記をFQDN(Fully Quolified Domain Name)と呼ぶ。末尾の.(ピリオド)はルートドメイン(後述)を示している。本論文では、一部の例外を除いて、全てのドメイン名をFQDNで表現した。

A.1.2　DNSとドメイン名に関係する基本的概念: 名前空間木と権限

名前サーバ群とドメイン名を密接に関係づける概念が、「名前空間木」である。そして、名前空間木は、「ゾーン」により構成され、その構造は「権限」および「権限委譲」の概念で構築される。そして、ゾーンは「リソースレコード(RR)」の集合である。

名前空間木とはドメイン名の定義域であり、全てのドメイン名は名前空間木に属する。この木の一部を抽出した例を図A.1に示す。ドメイン名は名前空間木の「節」あるいは末端である「葉」に相当する。名前空間木においては、「上位ドメイン」とは根により近い節の事を意味し、「下位ドメイン」とは葉により近い節、あるいは葉そのものを意味する。

名前空間木は節を境界として分割する事ができる。例えば、`keio.ac.jp.`という分割すると、`keio.ac.jp.`以下の枝が名前空間木の中で分割される事になる。分割した枝をそれぞれ「ゾーン」と呼ぶ。

ゾーンには、その枝の頂点に位置する節のドメイン名を代表として付ける。例えば、`sfc.keio.ac.jp.`を頂点とするゾーンは、`sfc.keio.ac.jp.`ゾーンと呼ぶ。また、ゾーンは入れ子構造を持つ。`keio.ac.jp.`ゾーンは`ac.jp.`ゾーンに、`ac.jp.`ゾーンは`jp.`ゾーンに含まれている。慶應大学に関連する一部のゾーンの包含関係を図A.2に示す。

名前サーバは、一つあるいは複数のゾーンの情報を管理するディレクトリサーバである。名前サーバとゾーンは、「権限(Authority)」によって関連づけられる。ゾーンの情報を管理する名前サーバを、そのゾーンに対して「権限を持つ名前サーバ(Authoritative Name Server)」と定義す

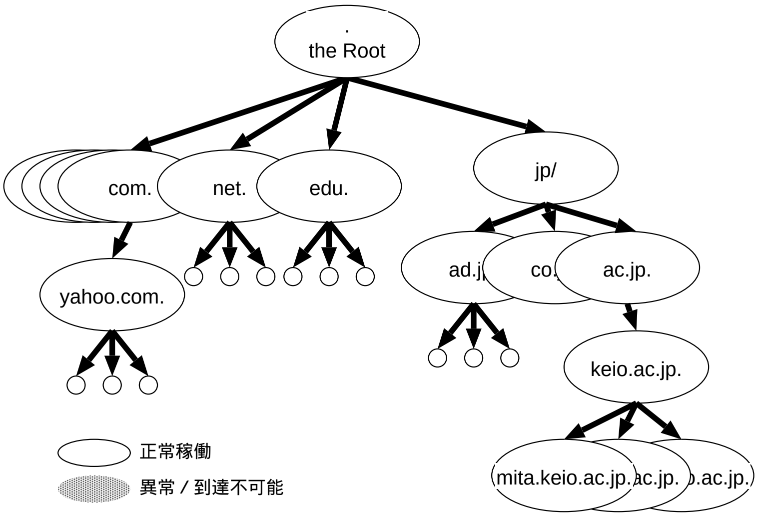

図 A.1: 名前空間木の一部 (例)

る。ゾーンに対して権限を持つ名前サーバは1つ以上存在する。

あるゾーンの直上に位置するゾーンの名前サーバが、そのゾーンに対する「権限」を持つ名前サーバを決める。これを「権限委譲(Delegation)」と呼ぶ。図A.2に示された包含関係は、そのままドメイン名に対する権限の委譲の関係に相当する。この場合、円で表現されたドメイン名に対応する名前サーバ群が存在し、その名前サーバ群は三角形で表現されるゾーンの名前を管理する。

権限委譲によって構成される名前空間木の頂点に位置するゾーンは、「ルート(root: 根)」と呼ばれ、一つの.(ピリオド)で示す。ルートに対して権限を持つサーバ群を特に「ルートネームサーバ」と呼び区別する。ルートネームサーバは、1999年12月時点で世界に13個存在する[30]。そして、ルートネームサーバからの直接的・間接的な権限委譲によって、世界中の全ての名前空間が構成されている

リソースレコード(RR)は、ゾーンに含まれる情報の単位である。ゾーンは一つの「ディレクトリ」に相当し、ディレクトリのエントリに相当する情報が、RRである。RRにはいくつかの形式があるが、ここではA/AAAA/NS RRのみについて述べる。

A RRおよびAAAA RRはドメイン名に対応するIPアドレスの情報である。A RRはIPv4に、AAAA RRはIPv6にそれぞれ対応する。「ホスト名」と呼ばれるドメイン名には大抵A/AAAA RRが対応付けられる。一方、NS RRは、ゾーンに対応する名前サーバのホスト名を示す。ある名前に対してNS RRが定義されているという事は、その名前を頂点とするゾーンに対し、NS RRによって指定された名前サーバが権限を持つ事を意味する。つまりNS RRは、権限の委譲の表現である。

例えば、`keio.ac.jp.`ゾーン中に含まれる名前`sfc.keio.ac.jp.`に対して、ホスト名`ns.sfc.keio.ac.jp.`が権限を持つ旨のNS RRが存在する。この場合、`sfc.keio.ac.jp.`から下の名前空間は`ns.sfc.keio.ac.jp`に委譲されている事を意味する。

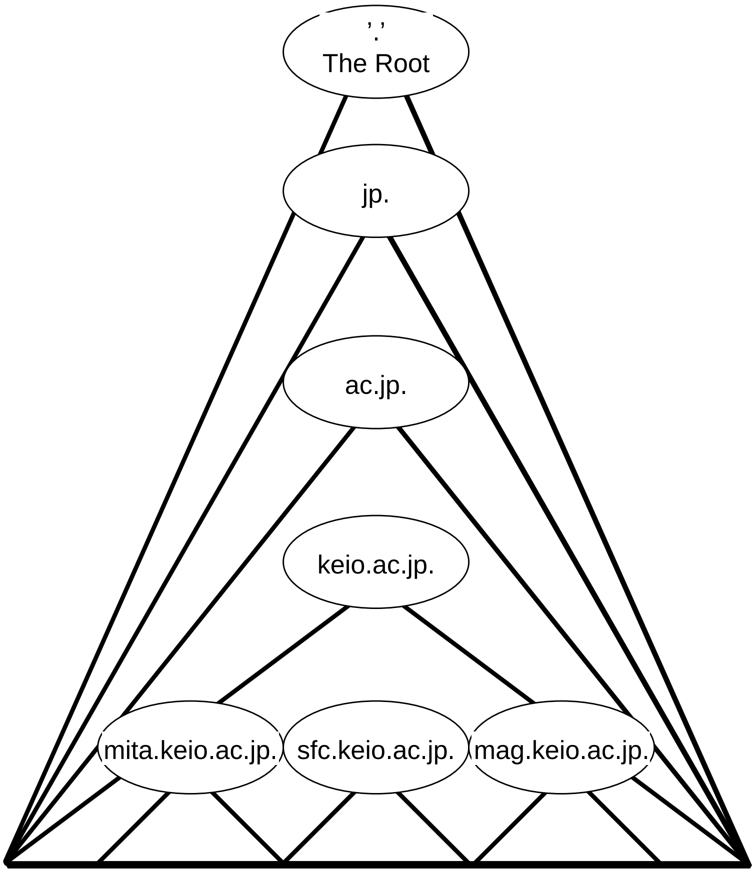

図 A.2: 慶應大学に関連するゾーンの包含関係 (一部)

A.1.3 再帰的名前解決

再帰的名前解決とは、ある名前を与えられた時に、名前空間木を再帰的に辿りながら名前に対応する情報を得るまでの手続き全体を指す。名前解決には一般に 3 種類の機構が関係する。

- 権限を持つ名前サーバ (Authoritative Name Server):
  ある名前空間に対して権限を持つ名前サーバ群。本論文では、単に「名前サーバ」と言った場合は権限を持つ名前サーバを指すものとする。
- 名前解決を行う名前サーバ/リゾルバサーバ:
  ネットワークを経由した名前問い合わせを受け付け、再帰的名前解決を代行するサーバ。本論文では、「リゾルバサーバ」と呼ぶ。
- 名前を問い合わせるライブラリルーチン/リゾルバライブラリ:
  ユーザが利用するプログラムに直接リンクされているライブラリルーチン。ネットワークを利用して、リゾルバサーバに対して問い合わせを行う。本論文では「リゾルバライブラリ」と呼ぶ。

名前解決は、名前空間木全体に対して、以下の手順で行われる。(詳細は略している)

1. ユーザアプリケーションがリゾルバライブラリを呼び出し、知りたい情報のドメイン名と情報の種類 (IP アドレス:A や、メールボックス情報:MX 等) を与える。
2. リゾルバライブラリは、与えられた問い合わせを DNS のメッセージに変換し、予め設定された近傍のリゾルバサーバにメッセージを送る。
3. リゾルバサーバは問い合わされた情報を得るまで以下の手続き (再帰的名前解決) を繰り返す。
   (a) リゾルバサーバは既知の名前サーバのリストの中から、問い合わされたドメイン名に最も近い名前空間に対して権限を持つ名前サーバ群を探し出す。もしも既知の名前サーバが何もなくても、リゾルバサーバはルートサーバ (群) を前提知識として知っている。
   (b) 手順 3a で得たサーバ群のうち任意のサーバにリゾルバライブラリからの問い合わせ (query) を送る。もしも一定時間内に返答が得られない場合は、同じサーバ群の中の違うサーバに問い合わせを再送する。
   (c) 名前サーバからの返答を調べて、問い合わされたドメイン名に関する情報があれば、ここで再帰的名前解決を終了。前回手順 3a で得た名前サーバよりも、問い合わされたドメイン名に近いドメイン名リストを得られたら、そのリストに対してもう一度手順 3b を実行する。もしも問い合わされたドメイン名に近付けない場合、再帰的名前解決は失敗する。
4. 問い合わされた情報を得たリゾルバサーバは、要求元のリゾルバライブラリに DNS メッセージの形で情報を伝える。
5. リゾルバライブラリは、DNS メッセージを解釈してユーザアプリケーションに必要な情報を与える。

具体例を以下に示す。図 A.3 は、キャッシュに記録されている既知のドメインが無い場合の、`sfc.keio.ac.jp.` ドメインの解決の例である。実線で書かれた情報はリゾルバサーバにとっての既知の情報で、破線のものは未知の情報である。また太い矢印は、下位ゾーンに権限を持つ名前サーバの位置を示し、実際には NS RR である。図の問い合わせの流れは、以下のようになっている。

1. 最上段左: ルートネームサーバについては、「ヒント」という形式で予め与えられている。
2. 最上段右: ルートネームサーバに `sfc.keio.ac.jp.` を問い合わせる。ルートネームサーバは既知の情報の中から `jp.` ゾーンの名前サーバ一覧を回答する。
3. 中段左: `jp.` ゾーンの名前サーバに同様に問い合わせると、その名前サーバはたまたま `ac.jp.` ゾーンの名前サーバも兼ねていた。`ac.jp.` ゾーンには `keio.ac.jp.` の名前サーバ一覧があるので、その情報を回答する。
4. 中段右: `keio.ac.jp.` の名前サーバによって `sfc.keio.ac.jp.` の名前サーバを知る。
5. 下段: `sfc.keio.ac.jp.` の名前サーバによって、求める情報を得る。

また、図 A.3 の解決を行なった後に、`mag.keio.ac.jp.` の解決を行う例を図 A.4 に示す。キャッシュ情報の利用により、ルートネームサーバからの再帰的名前解決の手順を大幅に省略できる。

1. 上段左: 既に `jp.` と `keio.ac.jp.` と `sfc.keio.ac.jp.` の名前サーバを知っている。
2. 上段右: 求める名前 `mag.keio.ac.jp.` を含むゾーンの名前サーバの中で、その名前に最も近いゾーンに権限を持つ `keio.ac.jp.` に対して問い合わせ、`mag.keio.ac.jp.` の名前サーバの一覧を得る。
3. 下段: `mag.keio.ac.jp.` の名前サーバにより、求める情報を得る。

A.1.4 DNSの冗長機構

DNSは、ネットワークの障害に対処するために、システムに冗長性を持たせている。つまり、ネットワークに何らかの障害が発生して、一部の名前サーバが使えない状況が発生しても、障害が一定の条件の範囲内に収まる限り、名前解決を継続できる。また、不確定なネットワークを常に利用するのではなく、情報を一時記憶(キャッシュ)する事で問い合わせの性能を確保している。主な冗長性の機構として、名前キャッシュとセカンダリサーバが存在する。

- 名前キャッシュ
  名前キャッシュとは、リゾルバサーバ上におけるリソースレコードの一時記憶である。全てのリソースレコードに対して、有効期限(TTL)が設定されている。リゾルバサーバは、名前解決を行った結果得た情報を有効期限で指定された時間記憶しておく事ができる。そして、以後の問い合わせで名前キャッシュ中に記憶されたリソースレコードを利用する。この一時記憶を利用しないと、毎回の名前検索を空間木の根から出発しなければならなくなり、性能、信頼性、規模性共に著しく低下する。また、キャッシュは、権限を持つ名前サーバ上の情報が変化しても、有効期限の間は古い情報を保持しつづけるという欠点がある。

- セカンダリサーバ

  セカンダリサーバとは、名前サーバの複製である。冗長性の確保は、空間木のそれぞれの節に対して、ネットワーク的に分散した複数の名前サーバを割当てる事で行われる。それらの名前サーバは、ゾーンの情報を管理するプライマリサーバ一つと、プライマリサーバの複製である複数のセカンダリサーバから構成される。

  あるゾーンを管理する名前サーバ群が分散していれば、再帰問い合わせの段階で名前サーバ群の一部が利用できなくても、少なくとも一つが利用できれば名前解決が成功する。全ての名前サーバが適切に分散させる事で、そのゾーンの信頼性を向上させられる。

  名前サーバの複製手続きを、ゾーン転送と呼ぶ。セカンダリ名前サーバは他の権限を持つ名前サーバ(必ずしもプライマリサーバとは限らない)に「ゾーン転送要求」を送り、ゾーンの内容を取得する。特にゾーン転送要求を行う側をスレーブ、ゾーンを転送する側をマスタと呼ぶ。プライマリサーバは常にマスタなので、プライマリマスタと呼ばれる事もある。

  セカンダリサーバを設置する際に問題となりうる点は、どの名前サーバに対してセカンダリの設定を行えば適切に名前サーバが分散しているのかが不明な点と、名前サーバを分散させるためには、他の名前サーバの管理者にセカンダリの設定を引き受けてもらう必要がある点がある。

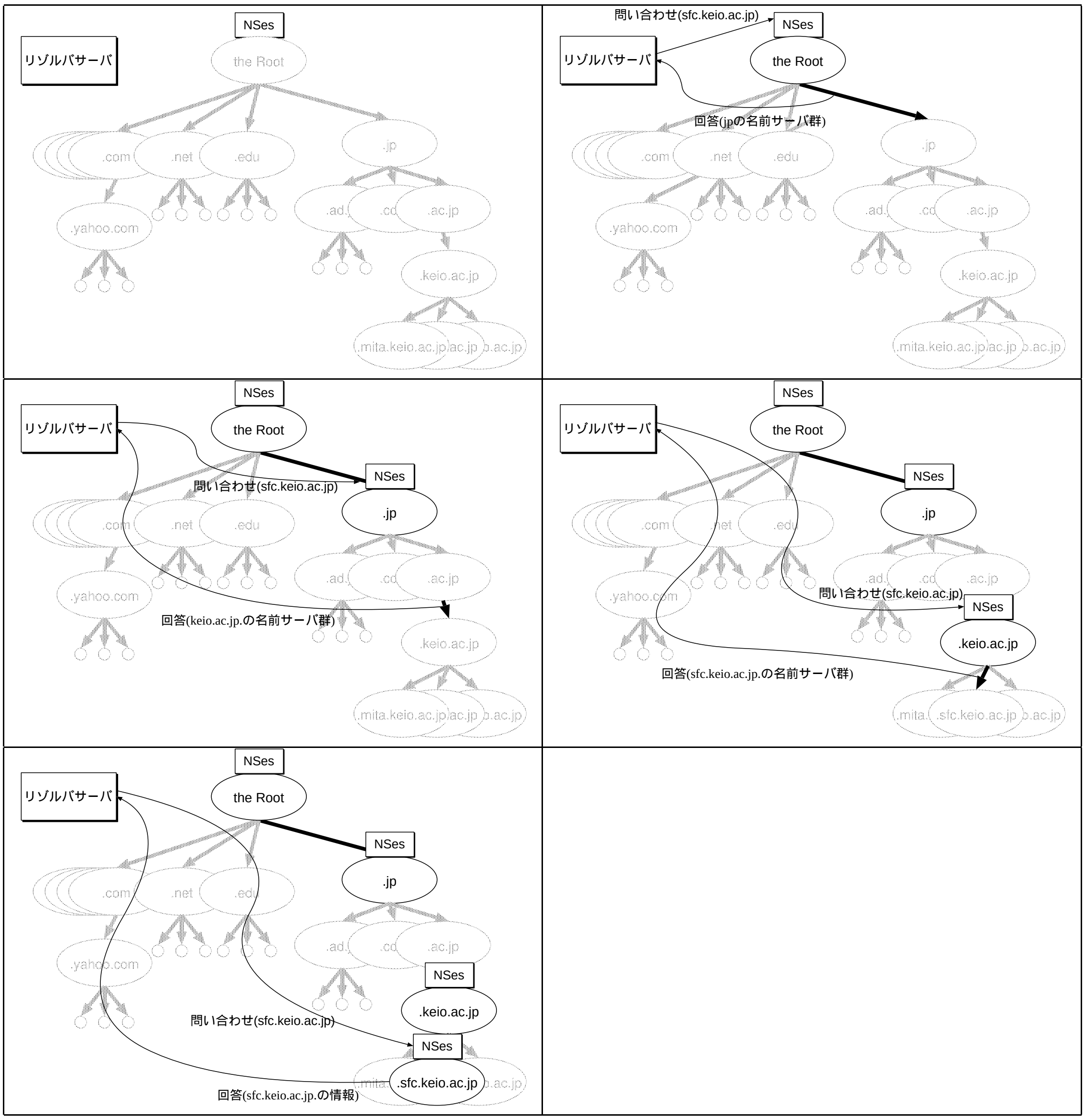

図 A.3: sfc.keio.ac.jp ドメインの解決

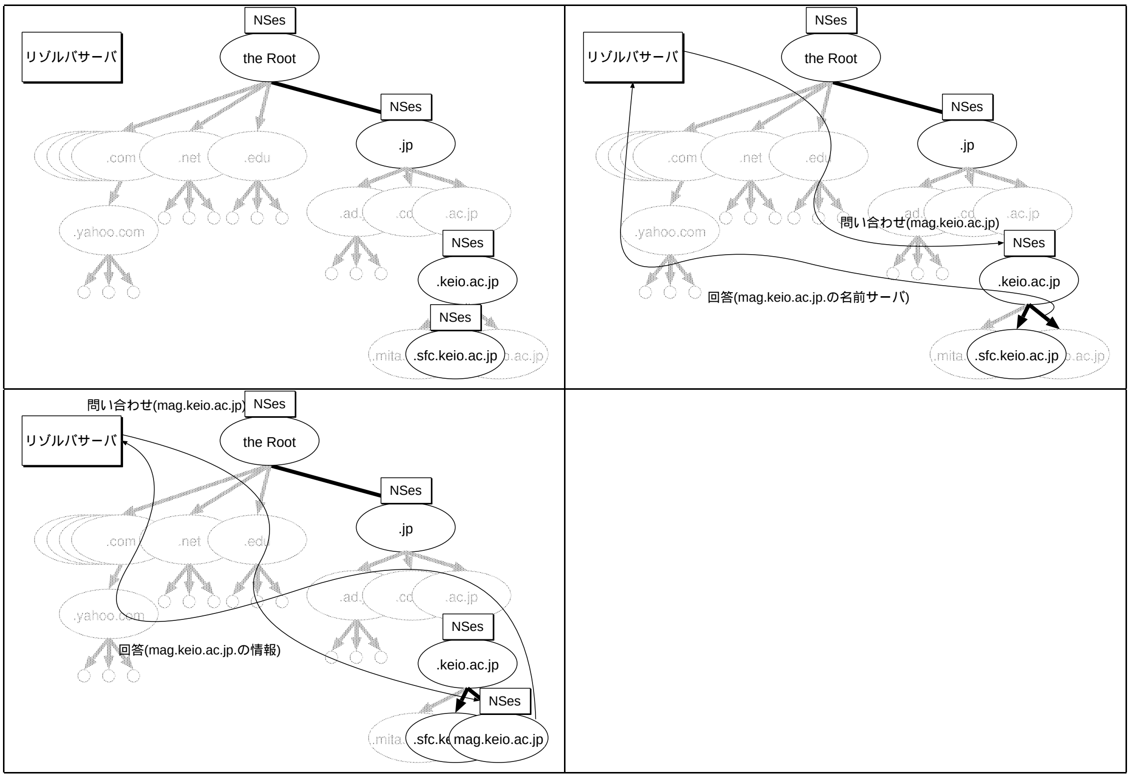

図 A.4: mag.keio.ac.jp ドメインの解決 (キャッシュ利用)

付 録B　関連ソフトウェアとその入手

関連ソフトウェアを次の URL にて公開している。

`http://www.sfc.wide.ad.jp/~doi/softs/index.html`

なお、本文第 6 章にて用いた、`dnsdump` について補足をする。`dnsdump` は、本研究の過程で作られた、DNS メッセージ解析プログラムである。これは、`libpcap` を利用して DNS に関するメッセージのみを読み、ASCII テキストでメッセージの内容を標準出力に出力するプログラムである。また、`tcpdump` 等で取得したネットワークダンプを読み、解析する機能を持つ。本研究では、ネットワークダンプの解析は全面的に `dnsdump` によった。

`$Id: main.tex,v 1.26 2000/01/15 16:15:09 doi Exp $`