平成15年度　研究開発成果報告書

「モバイル環境やセキュリティを考慮した名前解決方式と その検証環境の研究開発」

目　次

1 **研究開発課題の背景**.....3
2 **研究開発分野の現状**.....4
2-1 DNS を利用できない場合における名前解決.....4
2-2 DNS サーバの自動発見.....7
2-3 ファイアウォールで分断されたネットワーク環境における選択的名前解決8
2-4 DNS サーバの障害によるシステム全体の稼動効率への影響.....9
2-5 名前を公開しないことでセキュリティを考慮した運用.....10
2-6 DNS の標準化と BIND の実装状況.....11
2-6-1 BIND とは.....11
2-6-2 DNS の標準化状況.....12
2-6-3 IPv6 対応に関する BIND サーバの実装についての調査.....14
2-6-4 BIND サーバの性能に関する調査.....16
3 **研究開発の全体計画**.....22
3-1 研究開発課題の概要.....22
3-2 研究開発目標.....24
3-3 研究開発の年度別計画.....25
3-4 研究開発体制.....26
4 **研究開発の概要（平成１５年度まで）**.....27
4-1 研究開発実施計画.....27
4-1-1 研究開発の計画内容.....27
4-1-2 研究開発課題実施計画.....29
4-2 研究開発の実施内容.....30
5 **研究開発実施状況（平成１５年度）**.....31
5-1 ソフトウェアの試作.....32
5-1-1 複数の名前解決モジュールを組み込み、それらを選択的に利用可能な汎用名前解決エンジン.....33
5-1-2 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュール.....45
5-1-3 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュール.....54
5-2 統合動作検証.....59
5-2-1 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールと汎用名前解決エンジンの組み合わせ.....60

5-2-2 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールと汎用名前解決エンジンの組み合わせ 73
5-2-3 すべての名前解決モジュールと汎用名前解決エンジンの組み合わせ 79
5-2-4 Linux 版実装の動作検証..... 91
5-3 総括 100
5-3-1 本研究開発の成果について 100
5-3-2 本研究開発の成果がもたらす効果について 101
5-3-3 まとめ 102
参考資料、参考文献

（添付資料）
1 研究発表、講演、文献等一覧

1　研究開発課題の背景

現在インターネットを利用するほぼすべてのアプリケーションは DNS（Domain Name System）[1]による名前解決を利用している。通信相手の名前を解決して IP アドレスを取得する名前解決システムはインターネット上で通信相手を特定するために必要不可欠な機構であり、インターネット上でアプリケーションを利用するためには DNS による名前解決システムがプラットフォームに依存しない形で安定して稼動している必要がある。

通常の OS（Operating System）では DNS による名前解決を行うための共通のライブラリを提供している。また RFC2553[2]ではこのライブラリの基本的な API（Application Program Interface）を定めており、OS に依存しない汎化された API を利用できるようになっている。これらの API は通常近傍にある DNS cache サーバと通信し、名前と IP アドレスの相互変換を行う形になっている。このライブラリ群は通常リゾルバライブラリと呼ばれる。また DNS cache サーバとの通信は port 53 を利用して UDP あるいは TCP で行われる。これらのプロトコルは RFC1035、RFC2671[3]で定められている。

一方近年になり、特定のネットワーク上に固定的に接続された計算機環境だけではなく、異なるネットワーク間をノードが移動するモバイル環境も一般的になりつつある。特定の DNS cache サーバを固定的に設定しておけば十分であった固定環境と違い、モバイル環境においては移動先ネットワーク上で利用可能な DNS サーバを探索する必要があったり、DNS が利用できない環境で近隣ノードの名前解決を行ったりする必要が生じる。

あるいはセキュリティやプライバシーを考慮したネットワーク環境において、ファイアウォールで分断されたいずれのネットワークにおいても適切な名前解決を行えるような機能も必要となることがある。

本研究開発においては、一般のネットワークアプリケーションが名前解決を行う際に標準的に使用する DNS cache サーバとの通信機能を利用することで、様々なモバイル環境やセキュリティを考慮したネットワーク環境で利用可能な名前解決メカニズムをそれぞれの環境ごとに用意し、それらのメカニズムを統合的に処理することで環境に応じた適切な名前の解決を行うための汎用名前解決機構を試作開発して、各名前解決メカニズムの有用性を評価、検証することを目的とする。

---

[1] RFC1034・RFC1035・RFC1886

[2] RFC2553

[3] RFC2671

2　研究開発分野の現状

2-1　DNS を利用できない場合における名前解決

DNS（Domain Name System）を利用できない場合において、近隣のノードの名前の解決に利用できる方法として現在 IETF で議論されているプロトコルとしては以下の 2 つがある。

- Linklocal Multicast Name Resolution （LLMNR）[4]
- IPv6 Node Information Queries[5]

LLMNR は基本的に各ノードが自分自身の名前を答える DNS サーバになる。各ノードは問い合わせの際に、Link-scope multicast addresses を利用して問い合わせる。これにより、link 内のノードに対してのみ問い合わせを行うことが可能になる。問い合わせ／応答に利用されるパケットフォーマットは
図 1 のように DNS とほぼ共通である。

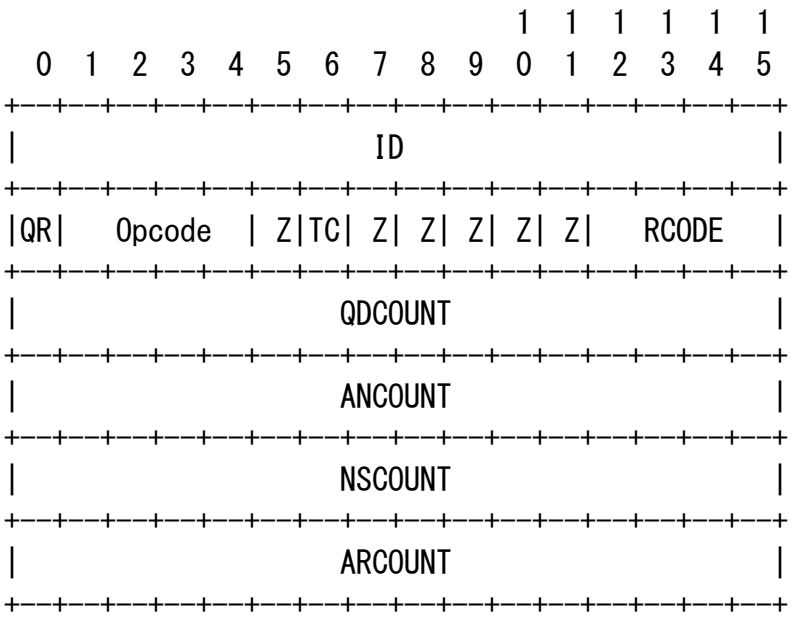
一方、IPv6 node information query は、ICMP を利用してノードの情報を取得するためのプロトコルである。このノードの情報には、ノードの持つアドレスや、ノード自身が定義した自己の名前（hostname）が含まれる。問い合わせ／応答に利用されるパケットフォーマットは、図 2 のようなフォーマットの ICMP パケットとなる。

LLMNR では、名前を解決したい対象となるノードで LLMNR を理解し、応答するためのプログラムが必要となる。

また、ICMP node information query もこのプロトコルを理解して応答するためのプログラムが必要となる。しかし現状では、node information query の方が仕様が固まっており、また実装も多いので、今回の研究では node information query を使った。

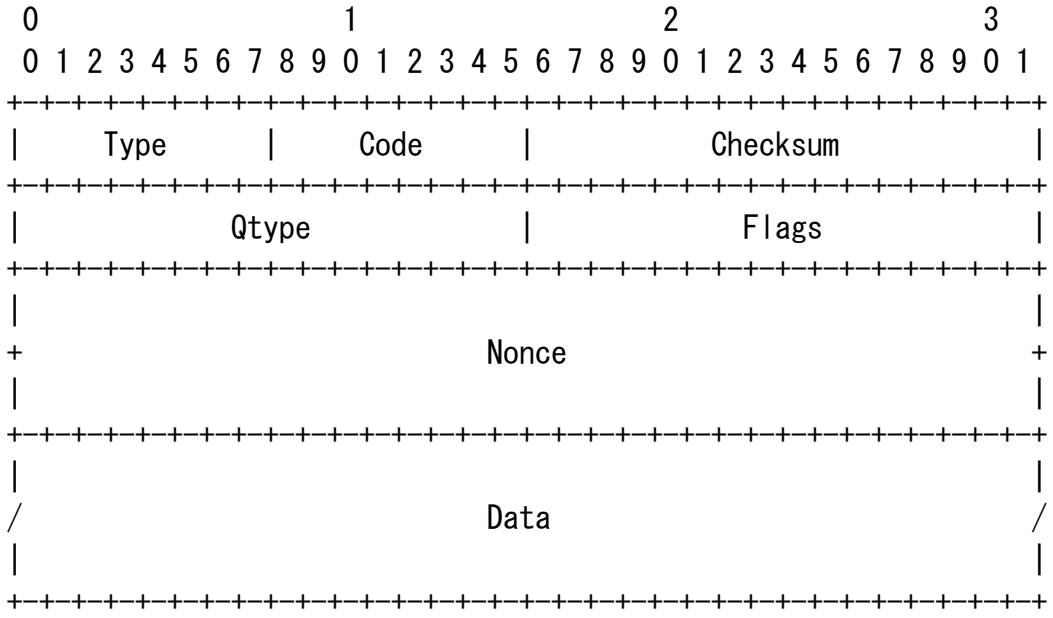
[4] draft-ietf-dnsext-mdns-30.txt

[5] draft-ietf-ipngwg-icmp-name-lookups-10.txt



ID	16bit の識別子
QR	0 ならば問い合わせ， 1 ならば応答
OPCODE	問い合わせの種類を表す。基本的に 0
TC	TrunCation bit: パケット長に対してデータ量が多く, すべてのデータがパケットに入らなかった場合は 1 となる
Z	予約ビット。基本的に 0
RCODE	応答の種類を表す。基本的に 0。
QDCOUNT	問い合わせ数。基本的に 1。
ANCOUNT	応答に含まれるレコード数。
NSCOUNT	Authority section に含まれる name server のレコード数
ARCOUNT	Additional records section に含まれるレコード数

図 1 LLMNR の問い合わせ／応答パケットフォーマット



Type　　パケットの種類を示す。 139 ならば問い合わせで 140 ならば応答である。

Code　　問い合わせ/応答に対するデータ部分に含まれる情報の種類を示す
Checksum　　ICMPv6 の checksum に従う。

Qtype　　問い合わせ/応答の種別を表す
Flags　　問い合わせ/応答に対する補助的な情報を示す。
Nonce　　64 ビットのランダムな値。なりすましを防ぎ，問い合わせと応答のマッチングをとるのに利用される。
Data　　問い合わせるデータ

図 2 IPv6 node information query の問い合わせ／応答パケットフォーマット

2-2　DNS サーバの自動発見

現在の DNS では、クライアントは通常近傍の DNS サーバを IP アドレスで指定することによって名前解決が行われる。この方法では、ユーザが何らかの形で近傍の DNS サーバを発見し、その IP アドレスを設定しなければならない。これはユーザが手設定することも可能であるが、現在の IPv4（Internet Protocol version 4）における主流は DHCP（Dynamic Host Configuration Protocol）クライアントが DHCP サーバから近傍の DNS サーバを通知してもらい、その値を設定するという方法である。この方法はユーザへの負担が少ないが、管理者は最終的に DNS サーバの設定だけでなく、DHCP サーバにその設定した DNS サーバのアドレスを静的に設定しなければならない。これは管理者の設定および保守コストを引き上げる結果となっている。加えて、IPv6（Internet Protocol version 6）においては、各ノードがルータからの RA（Router Advertisement） message を受信することによって、自立的に自分のアドレスを決定し、インタフェースに割り当てる機能がある。この機能は Stateless Address Autoconfiguration と呼ばれ、ルータを設定するだけでそのルータに接続する IPv6 ノードが通信可能となることから、IPv6 の利用の容易さを高める特徴でもある。よって、アドレス情報など様々な管理オブジェクトを持つ DHCP というプロトコルは IPv6 では広く利用されないことが考えられ、DHCP によって DNS サーバのアドレスをクライアントに通知するという方法は好ましくないと思われる。

これを解決するために、Well-known なアドレスを DNS サーバに割り当て、そのアドレスに対して問い合わせを行うことにより、自動的な DNS サーバの発見を行う。また、Well-known なアドレスについては anycast を利用することも考慮する。

Well-known なアドレスとしては、Well known site local unicast addresses for DNS resolver[6]で定義されているアドレスを利用することができる。

ただし Well-known site local unicast address を利用して DNS サーバを自動発見する本方式はその後標準化への正式採用が見送られ、現在は DHCPv6（Dynamic Host Configuration Protocol for IPv6）[7]を利用する方式[8]と RA を利用する方式[9]とを中心として標準化の検討が進められている。

---

[6] draft-ietf-ipv6-dns-discovery-07.txt

[7] RFC3646

[8] RFC3736

[9] draft-jeong-dnsop-ipv6-dns-discovery-01.txt

2-3　ファイアウォールで分断されたネットワーク環境における選択的名前解決

DNS では、名前空間の構造は木構造となっており、インターネットにおいてはこの構成木は一つだけであって、いかなるDNSサーバに名前を問い合わせても必ず同じ答えが得られるようになっている。例えば www.toshiba.co.jp という名前を持つ IP アドレスの値を、日本にある DNS サーバとアメリカにある DNS サーバのどちらに問い合わせても同じ値が返される。そのため、クライアントは任意の DNS サーバに名前を問い合わせることができる。

近年は網の安全性を確保するためにしばしばファイアウォールが導入される。このような環境においては、セキュリティ向上のためファイアウォール内部のノードの IP アドレスを公開しない場合が多い。これはすなわち前述の「構成木は一つ」という前提が失われていることを意味する。すなわち、ファイアウォール内部の DNS サーバに問い合わせた場合と、ファイアウォール外部で問い合わせた場合、応答の結果が違う場合が起こりうる。これは応答の内容が異なるというだけではなく、どちらかの応答は害となる可能性が大きい。例えばファイアウォール内部でしか公開されていない名前の IP アドレスをファイアウォール外部の DNS サーバに問い合わせた場合、内部の DNS サーバに訊けば IP アドレスが得られるにも関わらず、そのような名前を持つノードは存在しないという応答が返され通信できないという状態に陥る。ファイアウォールは大企業だけの利用にとどまらず、SOHO（Small-Office/Home-Office）、あるいは簡易なものであれば家庭網でも利用されることが考えられるので、この問題は深刻となる場合が大きい。

このため、移動するノードは、複数のDNSサーバに問い合わせることが可能になっている場合においては、ある問い合わせに対して適切な応答を選択しなければならない。

2-4　DNS サーバの障害によるシステム全体の稼動効率への影響

ネットワークを運用する上で、DNS はインフラストラクチャとして非常に重要な役割を担っている。ネットワークが依存している DNS サーバに障害が発生した場合、それがネットワーク上で稼動する各種アプリケーションの利用に実質的な影響を及ぼすことは少なくない。したがって通常ネットワークを構築する際には、DNS サーバの配置に冗長性を持たせることによって、いずれかの DNS サーバに障害が発生したとしてもネットワーク全体の運用を継続できるような構成が必須となる。

しかしながらアプリケーションを稼動させるクライアント上で利用される一般的な OS（Operating System）では、参照可能な DNS サーバの数はあまり多くなく、一台からせいぜい三台程度である。また複数の DNS サーバを参照可能な OS であっても、参照する際には優先順位を設定するようになっており、優先度の高い DNS サーバへの参照を先に行って、応答がない場合にのみ次の優先度が設定された DNS サーバへの参照を行うという振る舞いをするように実装されていることが多い。この挙動により、冗長性を持たせるためにネットワーク上に複数の DNS サーバを配置させていた場合であっても、優先度の高い特定の DNS サーバへの参照が集中することと、優先度の高い DNS サーバに障害が発生した場合に多くのアプリケーションにおいて当該 DNS サーバに対する参照のタイムアウト待ちが同時に発生してネットワークシステム全体の稼動効率が著しく下がることは避けられない。

2-5　名前を公開しないことでセキュリティを考慮した運用

昨今セキュリティ上の脅威の一種として、DoS（Denial of Service）攻撃や DDoS（Distributed Denial of Service）攻撃などが流行している。IPv4 環境においては実質的な IP アドレスの数が限られているため、全アドレスに対する網羅的な攻撃が少なからず認められたが、IPv6 環境においては利用可能な IP アドレスの数が実質上無限に近いため、網羅的な攻撃よりはピンポイント的な攻撃のほうが主流となることが予想される。この場合 DNS に名前が登録されているノードは攻撃対象となる可能性が高いため、実際にサービスを提供するなど名前を公開する必要性があるノード以外に関しては、DNS への登録を行わず名前を公開しないという運用が行われることが少なくないものと思われる。

IPv6 では IP アドレスが非常に長く、ネットワークを利用する際にユーザが直接 IP アドレスを指定するのは現実的とはいえない。そのため名前が公開されていないノードに対してアクセスをするために、ローカルなデータベースに名前と IP アドレスを登録する運用形態を想定することができる。またそのような運用形態の環境下においても、名前が公開されているノードに対しても並行してアクセスをする状況を考慮し、グローバルな DNS への問い合わせによる名前解決と透過的に利用可能であることも求められるだろう。

2-6　DNS の標準化と BIND の実装状況

2-6-1　BIND とは

BIND（Berkeley Internet Name Domain）は、DNS（とくにサーバ）の実装としてデファクト的な地位を占めるフリーソフトウェアである。BIND は米 ISC（Internet Software Consortium, 2004 年より Internet Systems Consortium に改称）で開発されている。

BIND には大きく 3 つのバージョンが存在する。現在の最新バージョンは BIND 9 である。BIND 9 は、これまでの BIND の開発・運用経験をもとに、設計段階から作り直した改訂バージョンである。BIND 9 における機能上の特徴として、以下が挙げられる。

- 次世代インターネットプロトコル IPv6 への対応
- DNS のセキュリティ拡張機能である DNSSEC への対応

2-6-2　DNS の標準化状況

IETF における DNS 関連技術の標準化において、中心となっているのは DNSSEC である。現在、DNS extension working group において、従来仕様の RFC2535[10]を抜本的に改訂する作業が進められている。この作業は 2003 年末の段階でほぼ完成に近い状態となっており、2004 年中には RFC として発行される可能性が高い。

IPv6 への対応という点では、2002 年 8 月に A6[11]リソースレコードおよびビットラベルを用いた表現方法が廃止となった（RFC3363[12]）ことで、プロトコル仕様については完成された状態となっている。

A6 は、ホスト名を IPv6 アドレスに対応させるための RR であり、AAAA RR の拡張という位置づけであった。

AAAA RR がホスト名から IPv6 アドレスそのものへの対応を与えるのとは異なり、A6 RR はプレフィクス名から対応する IPv6 アドレスの一部を与える。現在定義されているアドレス体系では、IPv6 アドレスは上位組織から下位組織への割り当てを繰り返す形で構成される。A6 によるアドレス解決の例を図 3 に示す。

リゾルバは、まずドメイン「example.org」の DNS サーバに対してこのホスト名の A6 RR を問い合わせる。その結果から、下位６４ビットと、その上位を識別するプレフィクス名「prefix1」が得られる。次に、同じ DNS サーバに対して、得られたプレフィクス名に対する A6 RR を問い合わせる。その結果から、48ビット目以降の16ビットの値と、上位のプレフィクス名「example.provider-X.net」が得られる。そこで、プロバイダ X の DNS サーバに対して、新しく得られたプレフィクス名「example.provider-X.net」の A6 RR を問い合わせ、その結果として、32ビット目以降48ビット目までの値と、さらに上位のプレフィクス名を得る。これを上位のプレフィクスがなくなるまで繰り返すと、最終的に完全なアドレス「2001:0501:4819:2000::1234」が得られる。

現在は、主に運用面での問題点を洗い出そうとする動きが中心である。ただし、DNS に関連したプロトコル仕様という意味では、IPv6 のホスト自動設定機能の中で DNS サーバアドレスをどのように設定するかという点が dnsop working group で議論されている。現在、DHCPv6 を用いる案、ルータ探索（router advertisement）機能を用いる案、既知（well-known）のアドレスを用いる案の 3 つが検討されており、2004 年夏までに結論を出す方向で議論が進められている。

---

[10] RFC2535
[11] RFC2874
[12] RFC3363

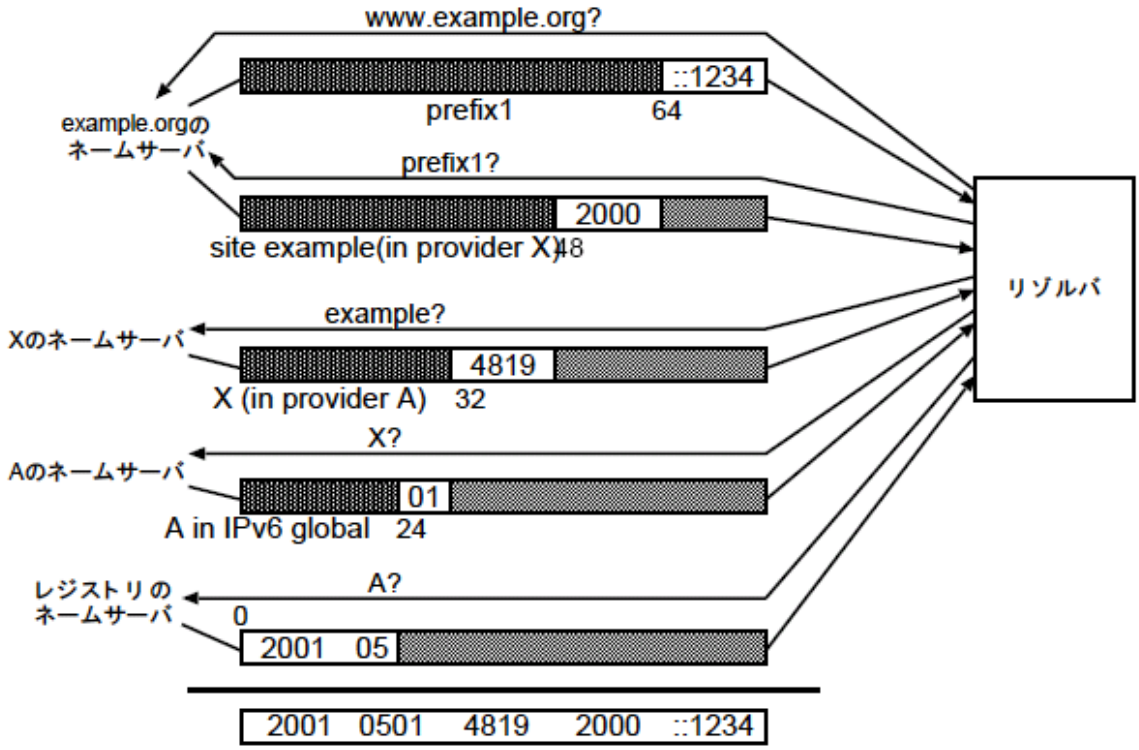


図 3 A6 RR 問い合わせによる名前解決の例

2-6-3　IPv6 対応に関する BIND サーバの実装についての調査

前述した標準化動向を踏まえ、BIND 9 における IPv6 対応の現状を調査した。その結果を以下に示す。なお、断りのない限り、以下の内容は BIND 9 の最新版である 9.3 に限定している。

(1) A6、ビットラベル廃止への対応

RFC3363 で A6、ビットラベルが廃止されたことを受けて、BIND9.3 ではビットラベルのサポートは全廃され、A6 を用いて名前を解決する機能も削除された。さらに、これに伴う実装面での整理により、3-5%の高速化が見込まれることがわかった。

(2) スコープ付きアドレスへの対応

IPv6 の特徴の一つに、リンクローカルアドレスに代表されるスコープ付きアドレスの概念がある。BIND におけるスコープ付きアドレスへの対応状況は以下の通りである。

- BIND に付属するテストツールの dig において、IPv6 の scoped address（実質的にはリンクローカルアドレス）をトランスポートとして利用できるようになった。
- ソースアドレスや他のサーバのアドレス、アクセス制御等で、IPv6 のスコープ付きアドレスを指定ができるような拡張が導入され、受信したパケットの処理も、その設定に対応した動作を取るように変更された。

(3) ip6.arpa ドメインへの対応

BIND 9.2 および 9.3 に対して、dig を IPv6 逆引きの新しいドメイン ip6.arpa に対応させるコードが追加された。

(4) その他の IPv6 対応の動向

- 待ち受け用のアドレスを IPv6 でも個別に指定できるようにする拡張機能が実装された（9.2 までは"all or nothing"でしか指定できなかった）。同時に、アクセス制御に利用する"localnets"および"localhost"という特殊な keyword が IPv6 にも適用

されるような改善も追加された。

- 上記の拡張に加え、API（Application Program Interface）仕様に最近追加された IPV6_V6ONLY socket option が利用できる場合にのみ wildcard bind するような機能が追加された。これより、IPv4 packet を意図しないうちに AF_INET6 socket で受信してしまうことによるアクセス制御漏れを防げるようになる。
- 9.2 までは、サーバからの問い合わせ IPv6 アドレス、ポートを指定する機能を利用すると、そのアドレス、ポート宛に送られた問い合わせを受け付けなくなる、という問題があった。この原因として、wildcard bind の問い合わせ待ち受けポートを指定したアドレス用のポートが奪っている状態となっていることが判明した。その対応として、問い合わせ用に指定されたポートが待ち受けポートを奪う形になっている場合には、問い合わせ用のポートも待ち受け用と併用するという修正が導入された。

2-6-4　BIND サーバの性能に関する調査

BIND 9 のサーバにおいて従来から指摘されている欠点に、性能面の劣化が挙げられる。すなわち、旧バージョンである BIND 8 に比べて多機能化された分、応答性能や起動性能が下がり、このために、BIND 9 への移行が遅れているという点がしばしば指摘されてきた。

IPv6 対応や DNSSEC などの機能の多くは、BIND 9 でのみ実現されており、性能上の問題によって BIND 9 への移行が遅れることは、これらの新機能の普及への障害ともなる。

そこで、BIND 9 の性能に関する問題を調査し、可能な場合にはその対応策を具体的に考察した。その結果を以下に述べる。

(1) 内部データベース検索の遅延

プロファイラによる性能検証の結果、BIND9 サーバの性能上のボトルネックの一つに、DNS のデータを内部データベースから検索する処理があることがわかった。これは、DNS の検索が一種の最長一致検索であり、またゾーン境界における権威の委譲などの概念があることによって処理が複雑となっていることが一因であり、本来ある程度のオーバヘッドが生じることは避けられない。実際、現在の BIND 9 の実装では木構造とハッシュを利用したデータベース実装で、アルゴリズム的には性能に十分配慮されており、ここに大きな改善を施す余地はあまりない。

このオーバヘッドを BIND9 サーバ実装のブロック図を用いて説明したものが図 4 である。"www.toshiba.com"という名前への問い合わせに対し、問い合わせ受け付け部（query processing routine）がそれを受理し、BIND9 内部のデータベース管理ルーチン（database management routine）に内部的に問い合わせる。データベース管理ルーチンは内部データベースから最善のデータを検索する。この例では toshiba.com ゾーンへの委譲が起きているため、NS レコードおよび付随する additional section 用 RR が応答データとなる。図 4 では、この部分が主要なオーバヘッドであることを太線を用いることで示した。

一方、DNS の応答メッセージの特性から、高価なデータベース検索を回避できる場面がいくつかある。それを利用した高速化手法を検討し、実装して性能を測定した。その内容は、具体的には以下の 2 つに分類される。

- additional section のキャッシュ
  問い合わせに対する直接の答え（answer section または authority section に入る）がわかっている場合、実際には additional section の値もほとんど既知である場合

が多い。そこで、answer section または authority section の RR に対応する additional section 用の RR をキャッシュしておくことで、これに対するデータベース検索を省略するという最適化が考えられる。

図 4 と対比させる形でこの最適化を表現した図が図 5 である。この図は、NS RR に対応する additional section 用 RR をキャッシュしている場面を示している。

図 6 では、キャッシュ後に同様の問い合わせが発生した場合の処理を示した。この場合、すでにあるキャッシュをもとに additional section の RR を内部データベースから検索するため、図 4 に比べるとこのときのオーバヘッドが軽微となっている。

この手法による性能向上の度合いは、当然ながら必要とする additional section のリソースレコードの数に依存する（多いほど効果的なはずである）。試作版における性能評価の結果から、ルートゾーンなど、13 の additional section RR を必要とする状況では、現在の実装に比べて 70% 処理能力が向上し、BIND 8 と比べても 80%（20%減）の性能が得られることを確認した。また、additional section の RR が 3 個の場合でも、現状の実装に比べて 26%向上、対 BIND 8 比で 64%の性能が得られた。

- NS（当該ゾーンの authority を持つ DNS サーバを指定する RR），SOA（当該ゾーンの authority 情報を表す RR）のキャッシュ
  NS および SOA リソースレコードは、各 DNS ゾーンについて高々一つ存在し、そのゾーンに対する問い合わせへの様々な場面で利用される。従来の BIND9 のコードは、必要に応じて NS や SOA RR をデータベースから検索していたが、これらの検索結果をキャッシュして直接利用することによって性能を向上させられる可能性がある。データベースのインタフェースを一部拡張することでこの機能を試作した結果、数%程度の性能向上の効果を確認した。

(2) マルチプロセッサ環境における性能

BIND 9 サーバはマルチプロセッサ環境を意識して thread を用いて実装されているが、実際には OS の thread サポートが貧弱だったり、ロックによるオーバヘッドが高すぎるなどして、マルチプロセッサ環境ではかえって性能が落ちる場合が多いことがわかっている。これを改善させることを性能面での次の主要な課題に位置づけて、そのための基本的な調査を行った。

Alpha プロセッサ 4 台を持つ Digital UNIX Tru64 5.1 をターゲット環境として、性能上のボトルネックを洗い出し、その改善方法を検討した。

初期段階での単純なテストの結果から、BIND 9 のコード自体に含まれる thread のオーバヘッドよりも、単一のソケットに対する複数の thread からの入出力の競合の方が大きな障害となっているらしいことが判明した。

そこで、純粋に入出力のオーバヘッドだけを測れるように簡潔化したマルチスレッドのテストコードを書き、複数のスレッドが同時に入出力を試みるモデルと、入出力専用のスレッドによってこの部分の処理を一本化するモデル（この場合は実際の処理をするスレッドと入出力スレッドの間で同期を取る必要があり、それが別種のオーバヘッドとなる）との間での性能を比較してみた。その結果としては、後者の方が全体としての性能は若干よくなることがわかったが、実際のデータ処理をするスレッドと入出力スレッドとの間の通信のタイミングが難しく、それほど大きな向上にはつながらなかった。

これらの結果は、OS によって大きく変わる可能性もあるため、今後は、まず複数の OS における状況の詳細な調査と、それらに共通する改善方法を検討する必要があるであろう。

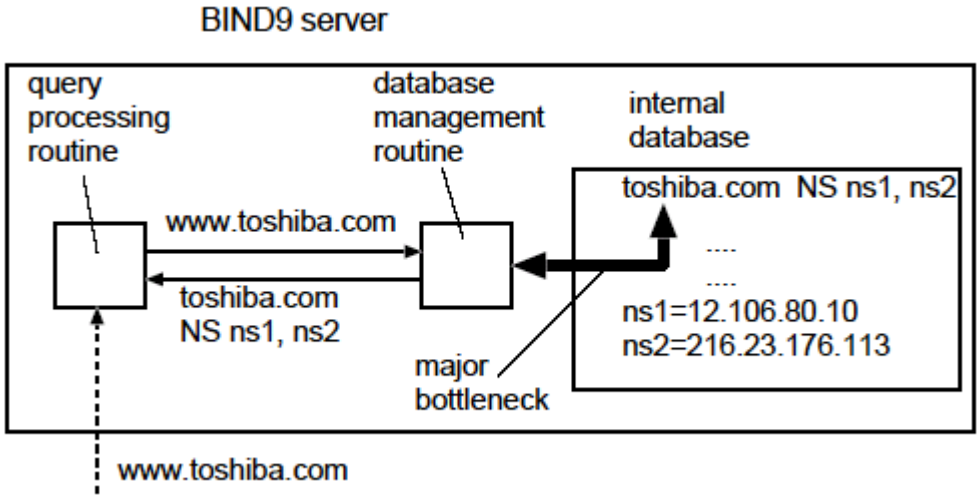


図 4 BIND9 内部データベース処理のオーバヘッド

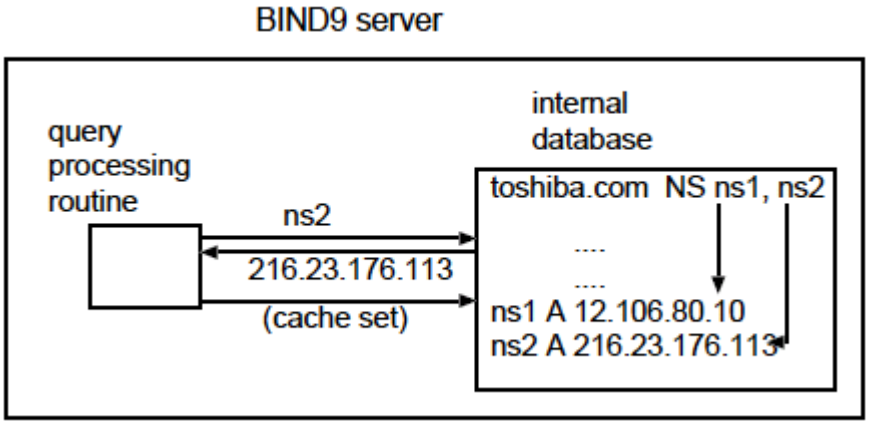


図 5　additional section のキャッシュによる最適化

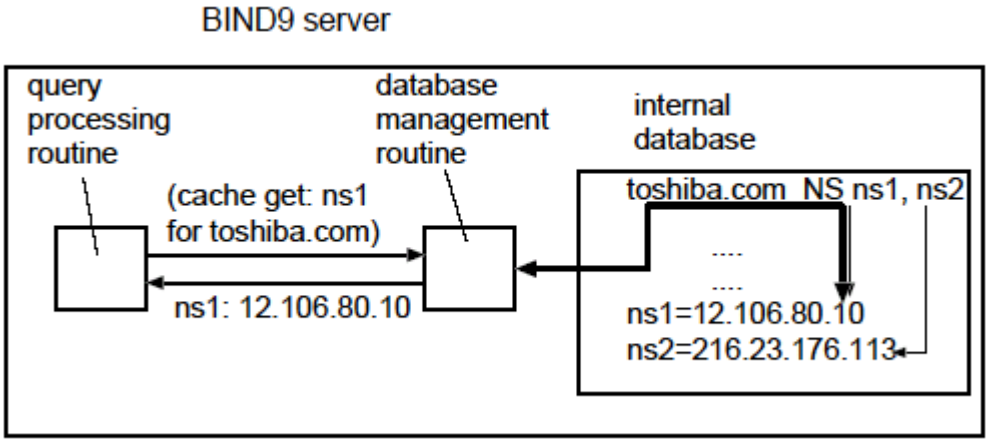


図 6 キャッシュ後の問い合わせ処理

3　研究開発の全体計画

3-1　研究開発課題の概要

IPv6（Internet Protocol version 6）システムを利用するために必須となる名前解決システムを実現するため、様々な名前解決プロトコルを組み合わせてもユーザが意識することなくそれらを利用可能な汎用名前解決エンジンの研究開発を行う。またこのエンジン上のモジュールとして、ユーザが様々な場所へ移動してIPv6システムを利用可能な名前解決システムの研究開発、およびプライバシーやセキュリティを考慮した名前解決システムの研究開発を行う。さらに、これらモジュールを汎用名前解決エンジン上で選択的に動作させることにより、その有効性を確認する。
具体的には、以下の研究開発を行う。

（ア）　汎用名前解決エンジン
様々な名前解決メカニズムを統一的に扱い、既存の IP アプリケーションに対してそれらが解決した名前情報を透過的に利用可能な形で通知できるインタフェースを持つ汎用名前解決エンジンを、特定のプラットフォームに依存しない形で実装し、機能検証する。

（イ）　モバイルサポートのための名前解決システム
移動ノードが通信を行う場合に必要な名前解決への要件として、以下の項目について検討する。さらに、汎用名前解決エンジン上のモジュールとして実装し、機能検証する。

（イ―1）DNS（Domain Name System）が利用できない環境での（近隣ノードの）名前解決方式

（イ―2）移動場所に依存しない DNS サーバの自動発見方式

（イ―3）障害発生時等を考慮した DNS サーバの適応的選択方式

（ウ）　セキュリティやプライバシーを考慮した名前解決システム
名前解決におけるセキュリティおよびプライバシーからの観点の課題への要件として、以下の項目について検討する。さらに、汎用名前解決エンジン上のモジュールとして実装し、機能検証する。

（ウ―1）ファイアウォールで分断されたネットワーク環境における選択的名前解決方式

（ウ―2）名前を一般に公開しない機器に対する名前解決方式

そして、(ア)で開発した汎用名前解決エンジン上で、(イ)および(ウ)で開発した名前解決モジュール群を統合して動作させ、その有効性を確認する。

3-2　研究開発目標

平成16年3月末時点で、次の研究開発項目が完了することを目標とする。

(1) 複数の名前解決モジュールを組み込み、それらを選択的に利用可能な汎用名前解決エンジンの実現(3-1 (ア)に対応)

(2) DNS が利用できない環境でも近隣のノードの名前を解決可能な名前解決モジュールの作成(3-1 (イ―1)に対応)

(3) 近隣の DNS サーバを動的に発見可能な名前解決モジュールの作成(3-1 (イ―2)に対応)

(4) 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とするモジュールの作成(3-1 (イ―3)に対応)

(5) ファイアウォール等で分断されたネットワーク環境において、適切な応答を選択可能な名前解決モジュールの作成(3-1 (ウ―1)に対応)

(6) 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールの作成(3-1 (ウ―2)に対応)

(7) 各名前解決モジュールを組み込んだ汎用名前解決エンジンの統合動作検証の完了

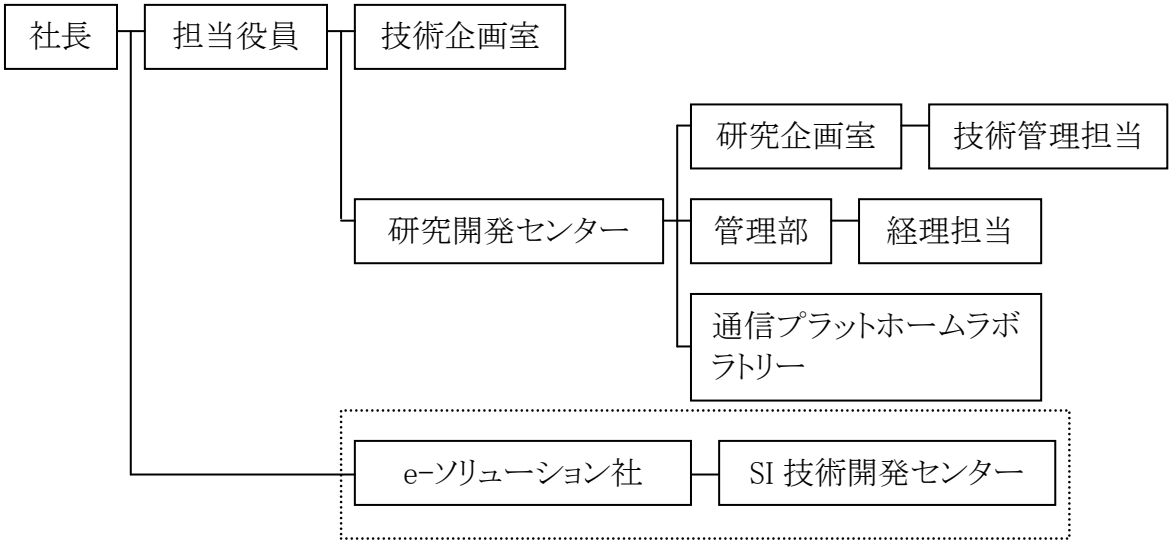
3-3　研究開発の年度別計画

（金額は非公表）

研究開発項目	13年度	14年度	15年度
【研究開発課題名】 モバイル環境セキュリティを考慮した名前解決方式とその検証環境の研究開発 【サブテーマ】 （ア）　汎用名前解決エンジン （イ）　モバイルサポートのための名前解決システム （ウ）　セキュリティやプライバシーを考慮した名前解決システム	調査・仕様検討	機能試作・評価	機能試作・評価・統合動作検証
間接経費			
合　計			

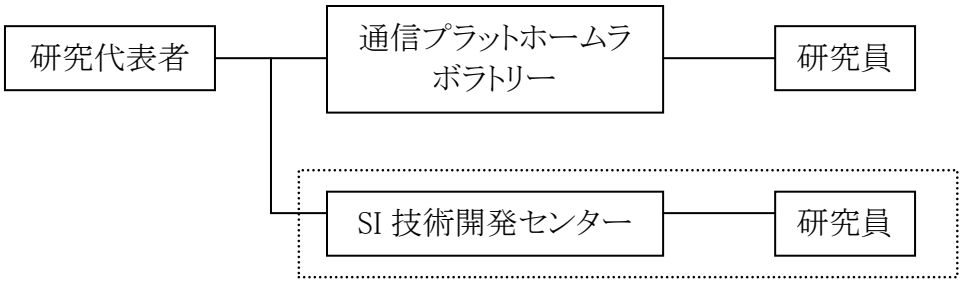
3-4　研究開発体制

(1)　研究開発管理体制



※平成15年9月末まで

(2)　研究開発実施体制



※平成15年9月末まで

4　研究開発の概要（平成１５年度まで）

4-1　研究開発実施計画

4-1-1　研究開発の計画内容

IPv6（Internet Protocol version 6）システムを利用するために必須となる名前解決システムを実現するため、さまざまな名前解決プロトコルを組み合わせてもユーザが意識すること無くそれらを利用可能な汎用名前解決エンジンの研究開発を行う。またこのエンジン上のモジュールとして、ユーザが様々な場所へ移動してIPv6システムを利用可能な名前解決システムの研究開発、及び、プライバシーやセキュリティを考慮した名前解決システムの研究開発を行う。さらに、これらモジュールを汎用名前解決エンジン上で選択的に動作させることにより、その有効性を確認する。

平成13年度は、既存の名前解決の機能と関連標準化（IETF）の動向を調査し、“モバイルサポートのための名前解決システム”と“セキュリティやプライバシーを考慮した名前解決システム”の要求仕様をまとめる。また、現在使用されている名前解決システムについての調査を行い、前述の要求仕様を実現する上での課題を明らかにし、“汎用名前解決エンジン”への要求仕様をまとめる。

平成14年度は、“モバイルサポートのための名前解決システム”と“セキュリティやプライバシーを考慮した名前解決システム”のために必要な共通基盤要素技術の試作、および様々なモバイルサポート・セキュリティ要件に対応するための名前解決モジュールの試作を行い、両者を組み合わせた単体動作検証を完了する。具体的には、以下の5項目の研究開発を行う。

(1) 様々な名前解決システムのための共通基盤要素技術である、名前解決モジュールを組み込み可能な汎用名前解決エンジンの試作完了

(2) DNS（Domain Name System）が利用できない環境でも近隣のノードの名前を解決可能な名前解決モジュールの試作完了

(3) 近隣のDNSサーバを動的に発見可能な名前解決モジュールの試作完了

(4) ファイアウォール環境等で名前空間の構造が異なる場合において、複数のサーバから異なる応答を受けた場合においても適切な応答を選択可能な名前解決モジュールの試作完了

(5) 個別の名前解決モジュールと汎用名前解決エンジンの組み合わせによる

単体動作検証の完了

平成15年度は、平成14年度に開発した“汎用名前解決エンジン”、“モバイルサポートのための名前解決システム”、“セキュリティやプライバシーを考慮した名前解決システム”の各モジュールに加え、企業ネットワークやホームネットワークなどの実運用環境を想定した、様々な環境に適応するために必要な様々なモバイルサポート・セキュリティ要件に対応するための名前解決モジュールの試作を行う。さらにこれまで2年間に開発した名前解決モジュールを全て組み込んだ汎用名前解決エンジンの統合動作検証、評価を行い、さらに実環境への適用に必要となるマルチプラットフォーム化対応を行う。具体的には、以下の4項目の研究開発を行う。

(1) 様々な名前解決システムのための共通基盤要素技術として、複数の名前解決モジュールを同時に組み込み、それらを選択的に利用可能な汎用名前解決エンジンの試作完了

(2) 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールの試作完了（モバイルサポートのための名前解決システムの継続）

(3) 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールの試作完了（セキュリティやプライバシーを考慮した名前解決システムの継続）

(4) 試作が完了したすべての名前解決モジュールと汎用名前解決エンジンの組み合わせによる統合動作検証の完了およびマルチプラットフォーム化対応

4-1-2　研究開発課題実施計画

（金額は非公表）

研究開発項目	第 1 四半期	第 2 四半期	第 3 四半期	第 4 四半期	計
【研究開発課題名】 モバイル環境セキュリティを考慮した名前解決方式とその検証環境の研究開発 【サブテーマ】 (ア)　モバイルサポートのための名前解決システム	基本設計 →	コーディング →	単体検査 →		
(イ)　セキュリティやプライバシーを考慮した名前解決システム	基本設計(5) →	コーディング →	総合試験 →		
(ウ)　マルチプラットフォーム化、検証、適用評価	基本設計 →	→	コーディング →	総合試験 →	
間接経費					
合　計					

4-2　研究開発の実施内容

本研究開発では、次にあげる各ソフトウェアの試作および統合動作検証を行うものとする。

□　試作:

- 統合動作検証のため、複数の名前解決モジュールを組み込み、それらを選択的に利用可能な汎用名前解決エンジン
- 近隣のDNSサーバ障害発生時に適応的にDNSサーバを選択することによって、効率的な名前解決を可能とする名前解決モジュール
- 複数のローカルデータベースを利用した名前解決がグローバルなDNSシステムと透過的に利用可能な名前解決モジュール

□　統合動作検証:

- 各名前解決モジュールを組み込んだ汎用名前解決エンジン

5　研究開発実施状況（平成１５年度）

本研究開発では、次にあげる各ソフトウェアの試作および統合動作検証を行った。

□　試作:

- 複数の名前解決モジュールを組み込み、それらを選択的に利用可能な汎用名前解決エンジン
- 近隣の DNS（Domain Name System）サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュール
- 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュール

□　統合動作検証:

- 各名前解決モジュールを組み込んだ汎用名前解決エンジン

5-1　ソフトウェアの試作

複数の名前解決メカニズムを統一的に扱い、既存の IP アプリケーションに対してそれらが解決した名前情報を透過的に利用可能な形で通知できるインタフェースを持つ汎用名前解決エンジン(5-1-1　)を、特定のプラットフォームに依存しない形で実装した。

移動ノードが通信を行う場合に必要な名前解決の機能として、近隣の DNS サーバに障害が発生したときに適応的に DNS サーバを選択することで効率的な名前解決を可能とする機能(5-1-2　)を、前述の汎用名前解決エンジン上のモジュールとして実装した。

名前解決におけるセキュリティおよびプライバシーの保護を行うために必要な名前解決の機能として、複数のローカルデータベースを利用してグローバルなDNSシステムとの間で透過的に名前解決を行う機能(5-1-3　)を、前述の汎用名前解決エンジン上のモジュールとして実装した。

5-1-1　複数の名前解決モジュールを組み込み、それらを選択的に利用可能な汎用名前解決エンジン

(1) 試作開発の内容

リゾルバライブラリからの要求を受け、各名前解決モジュールに要求を伝え、その複数の結果から適切に応答を選び出しリゾルバライブラリに返す汎用名前解決エンジンを実装する。

具体的には、以下の項目について実装する。

- 同時接続の実現

  複数の異なる名前解決方式を実装したモジュールと試作した汎用名前解決エンジンが同時に通信可能とする。

- 動的管理の実現

  各名前解決方式モジュールの動的な状態管理を行えるようにし、汎用名前解決エンジンに対してモジュールが動的に接続、切断可能とする。

- モジュールポリシーの実現

  各モジュールの持つポリシーをモジュール単位で管理し、アプリケーションからの名前解決に対して適切なモジュールを選択し、その中のモジュールに対して問い合わせを要求可能とする。

- 応答選択の実現

  複数のモジュールを接続している場合、一つの応答に対して非同期に各モジュールからの応答が得られる。これらの応答を管理し、適切な応答を選択してアプリケーションに送信可能とする。

- 統合的タイムアウト処理の実現

  複数のモジュールを接続している場合、ある一つの問い合わせに対してそれぞれのモジュールが独立のタイムアウトを持つか、あるいはまったくタイムアウトしないモジュールもありうる。これらに対して、汎用名前解決システム自身が独立にタイムアウト時間を持ち、システム

内で予測できる名前解決の応答時間を得ることを可能とする。

(2) 背景

現在インターネットを利用するほぼすべてのアプリケーションは DNS による名前解決を利用している。通信相手の名前を解決して IP アドレスを取得する名前解決システムはインターネット上で通信相手を特定するために必要不可欠な機構であり、インターネット上でアプリケーションを利用するためには DNS による名前解決システムがプラットフォームに依存しない形で安定して稼動している必要がある。

通常の OS(Operating System)では DNS による名前解決を行うための共通のライブラリを提供している。また RFC2553 ではこのライブラリの基本的な API (Application Program Interface)を定めており、OS に依存しない汎化された API を利用できるようになっている。これらの API は通常近傍にある DNS cache サーバと通信し、名前と IP アドレスの相互変換を行う形になっている。このライブラリ群は通常リゾルバライブラリと呼ばれる。また DNS cache サーバとの通信は port 53 を利用して UDP あるいは TCP で行われる。これらのプロトコルは RFC1035、RFC2671 で定められている。

一方近年になり、特定のネットワーク上に固定的に接続された計算機環境だけではなく、異なるネットワーク間をノードが移動するモバイル環境も一般的になりつつある。特定の DNS cache サーバを固定的に設定しておけば十分であった固定環境と違い、モバイル環境においては移動先ネットワーク上で利用可能な DNS サーバを探索する必要があったり、DNS が利用できない環境で近隣ノードの名前解決を行ったりする必要が生じる。

あるいはセキュリティやプライバシーを考慮したネットワーク環境において、ファイアウォールで分断されたいずれのネットワークにおいても適切な名前解決を行えるような機能も必要となることがある。

(3) 本項目における課題

一般のネットワークアプリケーションが名前解決を行う際に標準的に使用する DNS cache サーバとの通信機能を利用することで、さまざまなモバイル環境やセキュリティを考慮したネットワーク環境で利用可能な名前解決メカニズムをそれぞれの環境ごとに用意し、それらのメカニズムを統合的に処理することで環境に応じた適切な名前の解決を行うための汎用名前解決機構の中心となる汎用名前解決エンジンを開発する。

(4) 実装の方針

汎用名前解決システムの概要を図 7 に示す。

ある名前解決を行うにあたり、その応答を引き出す概念上のエンティティをネームソースと呼ぶ。ネームソースは名前解決を行う各プロトコルに対して定義される。例えばある一つの DNS cache server は DNS 上の一つのネームソースであり、ある一つのリンクに対してある特定のプロトコルでブロードキャストを行って名前解決を行うような場合では、そのリンクは利用した特定のプロトコル上において一つのネームソースとなる。また、ノードが独自に管理するデータベースもネームソースにあたる。

汎用名前解決システムは、現在インターネットを利用するほぼすべてのアプリケーションが名前解決のために利用する DNS の仕組みを利用するものとする。具体的には、アプリケーションが使用する基本 API ライブラリ(リゾルバライブラリ)が DNS cache server に対して行う DNS 問い合わせのための通信を利用し、アプリケーションに対して DNS cache server として振舞うことで名前解決機能を提供することとなる。

各ノード上では DNS への問い合わせパケットを理解するデーモンプログラムを起動しておく。このデーモンプログラムは port 53 のパケットを受信し、名前要求の内容を理解し、適切な名前解決を行い応答するプログラムであり、これが汎用名前解決システムとなる。ノードでは、近傍の DNS cache server として自己のノード、すなわち IPv6(Internet Protocol version 6)アドレスでは::1、IPv4(Internet Protocol version 4)アドレスでは 127.0.0.1 を指定する。

各アプリケーションは従来どおりの動作を行う。すなわち、名前の解決のためにリゾルバライブラリを呼ぶ。リゾルバライブラリは指定された DNS cache server、すなわち自己に対して DNS の問い合わせパケットを送信する。このパケットは汎用名前解決システムによって受信される。汎用名前解決システムは、様々な名前解決機構に準じたモジュールを内包している。これら複数のモジュールに問い合わせることにより、適切なネームソースからの応答を選択し、リゾルバライブラリに対して応答する。リゾルバライブラリはこの応答をアプリケーションに伝える。

この結果として、汎用名前解決システムではアプリケーションや OS が提供するリゾルバライブラリにまったく手を加える必要がなく、複数の名前解決機構を利用できる。また、モジュールを追加することによって、新しい名前解決のためのメカニズムが提案された場合においてもモジュールの追加だけで各アプリケーションは新しい名前解決メカニズムを利用でき、効率的に実環境において各名前解決メカニズムの検証を行うことができる。

汎用名前解決システムは、主に汎用名前解決エンジンとモジュールに分けられる。汎用名前解決エンジンは、リゾルバライブラリからの要求を受け、モジュールに各要求を伝え、その複数の結果から適切に応答を選び出しリゾルバライブラリに返すのが主な役割である。図 8 に汎用名前解決システムの概略を示す。

汎用名前解決エンジンは、次の方針に基づいて設計を行った。汎用名前解決エンジンの概略を、図 9 に示す。

- 容易に実装可能となるようにモジュラリティの高い構成を持つこと。
- 既存のアプリケーションなどにインパクトを与えずに導入することが可能であること。
- 多くのプラットフォーム上で稼動させられるように移植性が高いこと。
- 複数の名前解決方式を同時に利用可能であること。
- 名前解決に対して得られた複数の結果に対し、適応的に応答を選択する機能を持つこと。

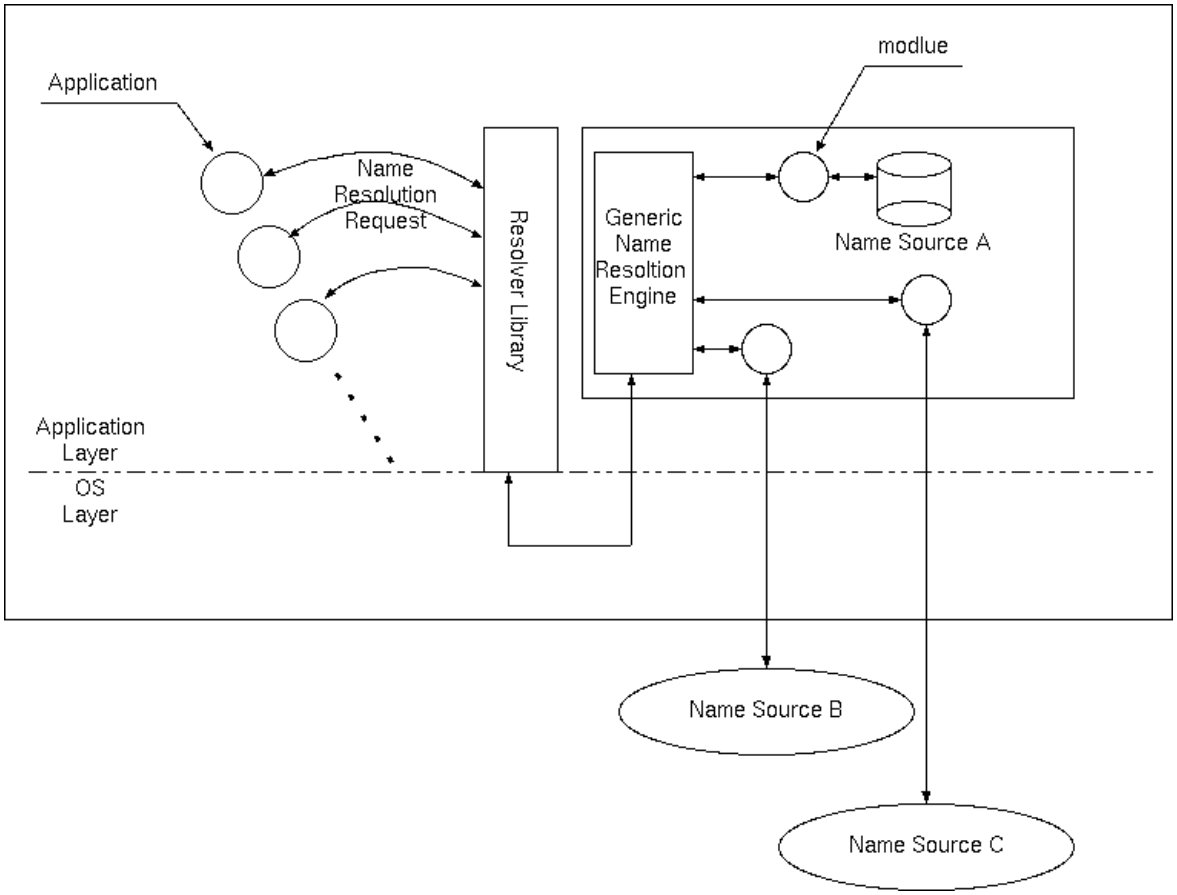


図 7 汎用名前解決システムの概要

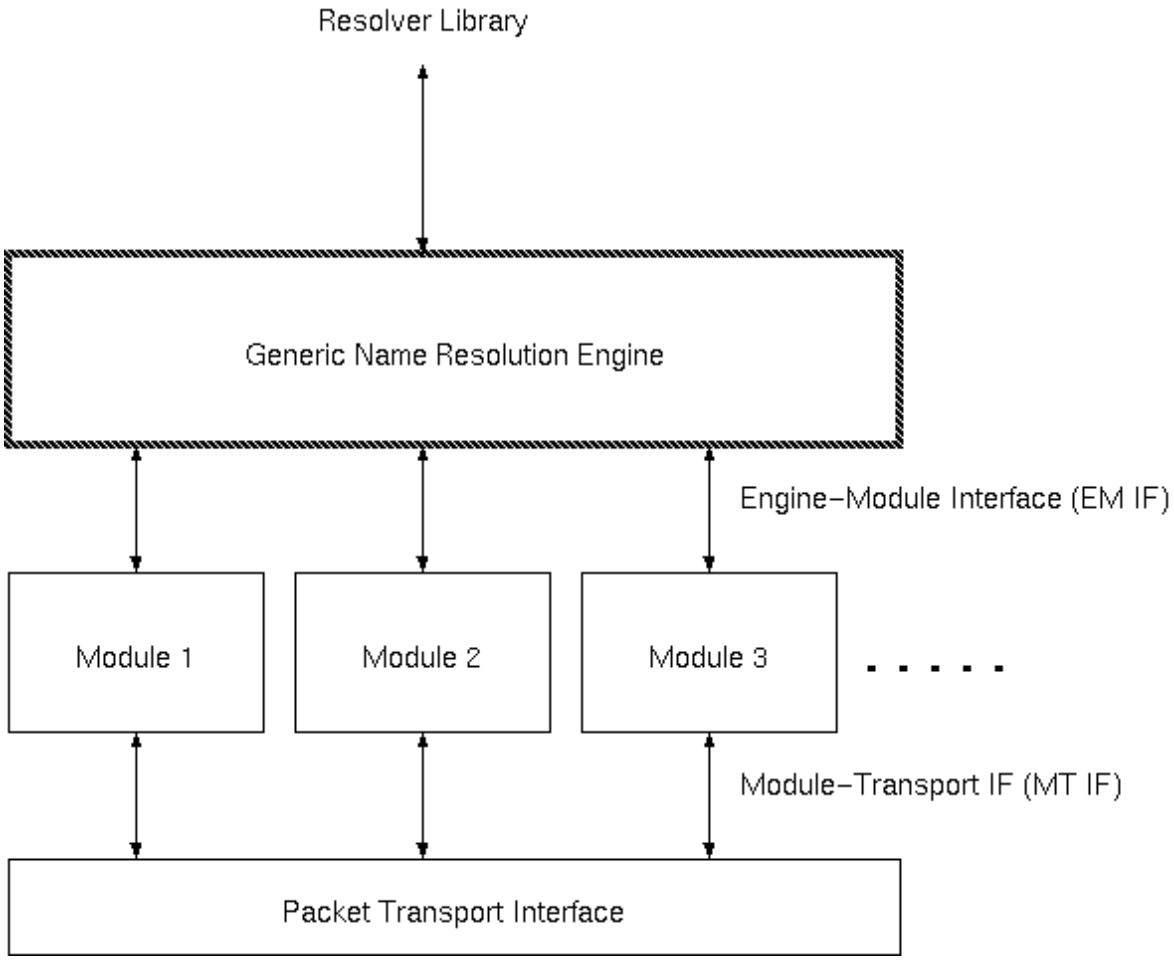


図 8 汎用名前解決システムの概略

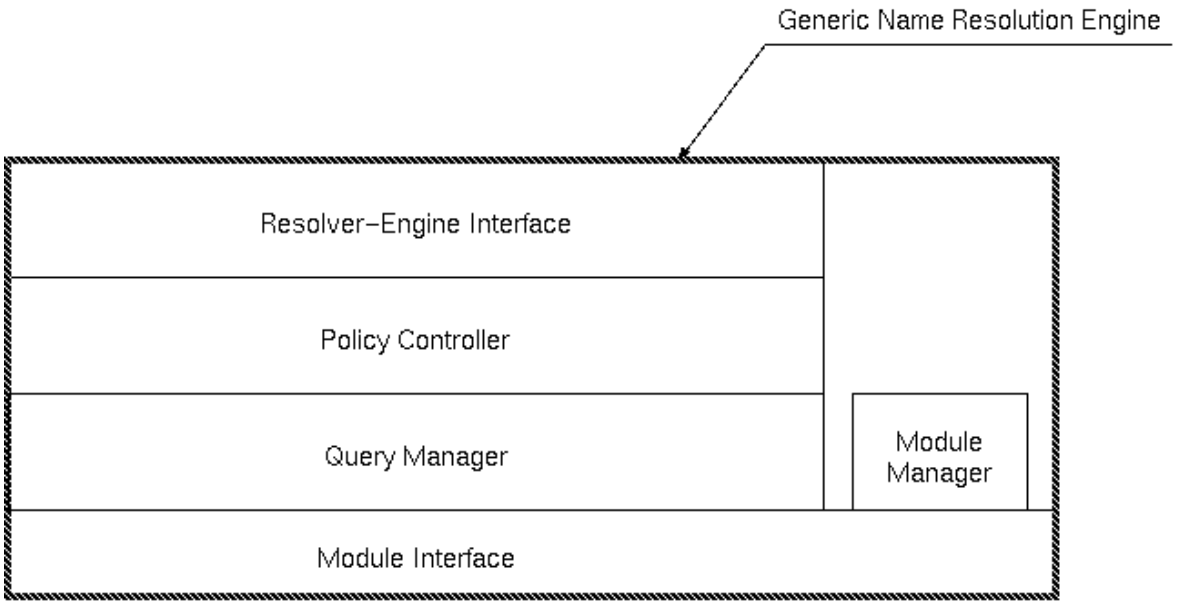


図 9 汎用名前解決エンジン概要

(5) 試作開発の状況

リゾルバライブラリからの要求を受け、各名前解決モジュールに要求を伝え、その複数の結果から適切に応答を選び出しリゾルバライブラリに返す汎用名前解決エンジンの試作開発を完了した。

(6) 本項目における成果

以下の 5 つのサブユニットからなる汎用名前解決エンジン「mng」を実装した。

- Resolver-Engine Interface
  リゾルバライブラリからの要求パケットを受け付け、内部形式に展開する。また、モジュールからの応答をリゾルバライブラリに伝える。

- Policy Controller
  要求された名前に対して複数の名前解決モジュールに問い合わせた場合に得られる複数個の応答から、どのような応答を優先するかを管理する。

- Query Manager
  非同期に行われる複数の名前解決に関し、それぞれの状態を管理する。

- Module Manager
  現在どのような名前解決モジュールが接続されているかを管理する。

- Module Interface
  汎用名前解決エンジンと各名前解決モジュールとのインタフェースを抽象化する。

汎用名前解決エンジン「mng」は、次の手順で起動する。

```
mng
```

ただしここでは、mng コマンドがコマンド実行パス中のディレクトリにインストールされているものとする。

汎用名前解決エンジン「mng」は、各名前解決モジュールより先に起動しておく必要がある。

また OS 上でアプリケーションが名前解決をするために、自ノード上で稼動している汎用名前解決エンジンに対して DNS 参照の問い合わせを行うように設定をする。たとえば UNIX 系 OS であれば、参照するネームサーバを登録する設定ファイルである/etc/resolv.conf ファイルに、次のような設定を行う。

```
nameserver ::1
```

この設定により、アプリケーションからの名前解決は自ノード上で稼動している汎用名前解決エンジン「mng」に対して名前の参照を行うことで実行されることになる。

(7) 実装内容の詳細

汎用名前解決エンジンを構成する各サブユニットについて説明する。

Resolver-Engine Interface

リゾルバライブラリからの要求パケットを受け付け、内部形式に展開する。また、モジュールからの応答をリゾルバライブラリへと伝える。リゾルバライブラリに対する反応も Resolver-Engine Interface の役割である。これにはまず各モジュールの処理とは独立に、最大のタイムアウト時間を決め、各モジュールの反応を待たずとも自立的に名前解決の失敗をリゾルバに通知するという役割がある。
汎用名前解決機構検証システム内の内部表現形式は、圧縮を利用しない DNS パケットフォーマットを利用する。

Policy Controller

要求された名前に対して、複数のモジュールに問い合わせた場合、当然複数個の応答が返ってくることが考えられる。どのモジュールを利用するか、どのような応答を優先するか、どのモジュールからの応答を優先するかを管理するのが Policy Controller の役割である。Policy は基本的に以下の要素によって管理される。以後この要素を Policy Selector と呼ぶ。

- Protocol
  名前の解決に使用するプロトコル。例えば DNS。

- Type
  問い合わせが解決を要求している型。RFC1035、RFC1886 等の QTYPE

に準ずる。

- 名前
  問い合わせが解決を要求している名前。

各モジュールはそれぞれが解決可能な Policy Selector の要素を Policy Controller に伝える。Policy Controller は、Resolver-Engine Interface から受けた要求を元に問い合わせ可能なモジュールを選択し、それらのモジュールに対して重み付けを行ったうえで各モジュールに問い合わせ要求を渡す。各モジュールに問い合わせ要求を渡す作業は実際には Query Manager を通して行われる。

Query Manager
名前の解決は非同期に行われる。例えば 3 つのアプリケーション A・B・C が同時に問い合わせを発行した場合には、これらは見かけ上同時に処理される。そのため、現在どのような応答をモジュールに対して待っているのかといった管理が必要となる。この処理を行うのが Query Manager である。現在どのような問い合わせが解決中であるか、また各問い合わせに対して現在どのモジュールから返ってきているかを管理する。

Module Manager
現在どのようなモジュールが接続されているかを識別し管理する。モジュールは接続がモジュール情報をエンジンに対して宣言する。Module Manager はこの情報を管理し、また接続情報も保持する。

Module Interface
各モジュールは汎用名前解決エンジンとは別に動作するプログラムであるので、エンジンとモジュールを結ぶインタフェースが必要となる。Module Interface はこのインタフェースの抽象化を行う層である。
今回試作したプロトタイプでは、各モジュールはローカルアドレスに対する TCP 接続によってエンジンとリンクする。Module Interface はモジュールからの Connection・Disconnection を扱う。また、Query Manager から、あるいはモジュールからのデータの Serialize・Packetize を行う。

名前解決モジュールは基本的にプロトコルごとに用意される。プロトコル単位でモジュールの内容は大きく異なる。基本的なモジュールフレームワークは図 10 に示すような構造を持ち、以下の要素から成り立つ。

Module Interface
エンジン側の Module Interface の stub となる層。また、プロトコル依存である Module Core で利用されるデータ構造に変換する。

Module Core
プロトコルごとの処理が記述される。

Per-Name-Source Connector
汎用名前解決機構検証システムにおいては、一つのプロトコル上で概念上複数のネームソースを持つことができる。各モジュールはどのネームソースからの応答かを名前解決エンジンに伝える必要がある。すなわち、応答はモジュール単位ではなくネームソース単位で管理される。

汎用名前解決エンジンとモジュールの間は、ローカルアドレスを使用した TCP コネクションで接続される。汎用名前解決エンジンは事前に定められた port を passive open し、各モジュールはその port に対して接続をする。

モジュールは接続後、以下の情報をモジュール情報として宣言する。

- プロトコル識別番号
  そのモジュールがどのようなプロトコルを利用するかを示す番号。プロトコル識別番号は別途定義される。

- モジュール名
  そのモジュール自身の名前。7 ビット ASCII で表記される。

- モジュールバージョン番号
  そのモジュールのバージョン番号。

- タイプセレクタ
  そのモジュールが解決できる Query Type。512 ビットのビットマスクで表現され、ビットが立っている部分が解決可能な Query Type である。RFC1035 的には 256 ビットしか使用しない。上位 256 ビットはローカルな拡張のために利用される。

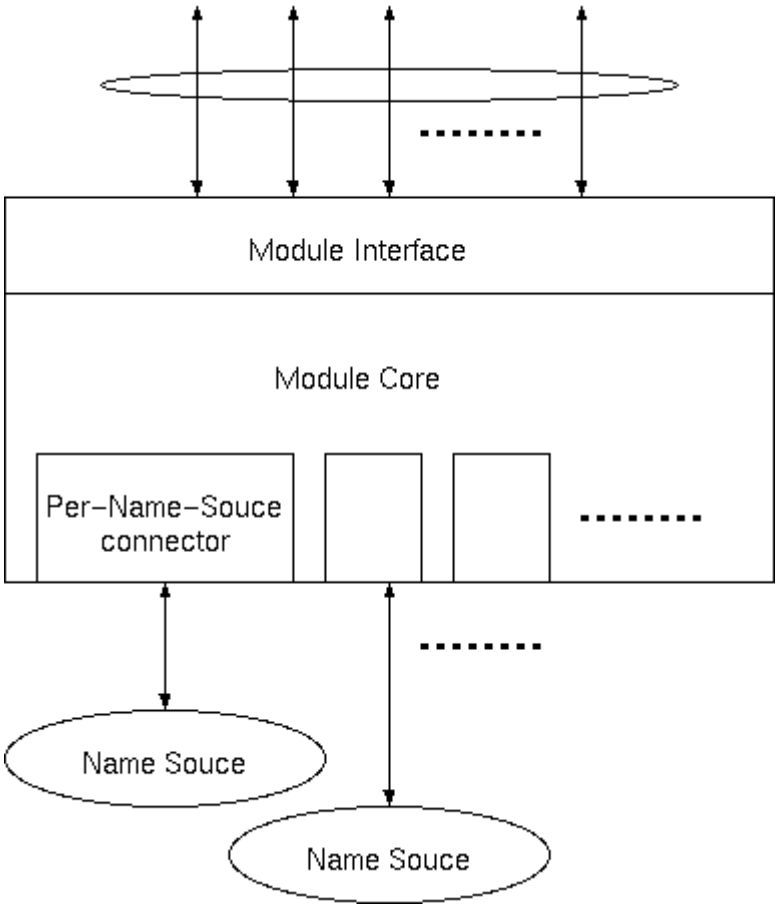


図 10 モジュールフレームワーク

5-1-2　近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュール

(1) 試作開発の内容

ノードの名前を解決するための DNS サーバが近隣に複数存在することを認識した状態において、それぞれの DNS サーバへの到達性や問い合わせに対する応答の有用性を動的に管理することで、いずれかの DNS サーバに障害が発生した場合でも、タイムアウト待ちなどによって運用効率を低下させることなく継続的な名前解決を可能とする名前解決モジュールを実装する。

具体的には、以下の項目について実装する。

- DNS サーバ単位の到達性管理の実現

  現在使用できると指定された DNS サーバに対して、それぞれのサーバに対して独立に現在の到達性を管理することを可能とする。

- DNS サーバの到達性の状態遷移把握の実現

  上記項目で把握された到達性管理情報に対して、現在は到達できないサーバ、あるいは到達できなくなったサーバに対しても、到達性の変化を確認することにより、再度到達可能となったサーバを再度利用可能とする。

(2) 背景

ネットワークを運用する上で、DNS はインフラストラクチャとして非常に重要な役割を担っている。ネットワークが依存している DNS サーバに障害が発生した場合、それがネットワーク上で稼動する各種アプリケーションの利用に実質的な影響を及ぼすことは少なくない。したがって通常ネットワークを構築する際には、DNS サーバの配置に冗長性を持たせることによって、いずれかの DNS サーバに障害が発生したとしてもネットワーク全体の運用を継続できるような構成が必須となる。

しかしながらアプリケーションを稼動させるクライアント上で利用される一般的な OS では、参照可能な DNS サーバの数はあまり多くなく、一台からせいぜい三台程度である。また複数の DNS サーバを参照可能な OS であっても、参照

する際には優先順位を設定するようになっており、優先度の高いDNSサーバへの参照を先に行って、応答がない場合にのみ次の優先度が設定されたDNS サーバへの参照を行うという振る舞いをするように実装されていることが多い。この挙動により、冗長性を持たせるためにネットワーク上に複数の DNS サーバを配置させていた場合であっても、優先度の高い特定の DNS サーバへの参照が集中することと、優先度の高い DNS サーバに障害が発生した場合に多くのアプリケーションにおいて当該 DNS サーバに対する参照のタイムアウト待ちが同時に発生してネットワークシステム全体の稼動効率が著しく下がることは避けられない。

通常の DNS サーバ参照時と、本名前解決モジュール使用時における DNS サーバ参照時とで、DNS サーバに障害が発生した場合の挙動の違いを、図 11 および図 12 に示す。

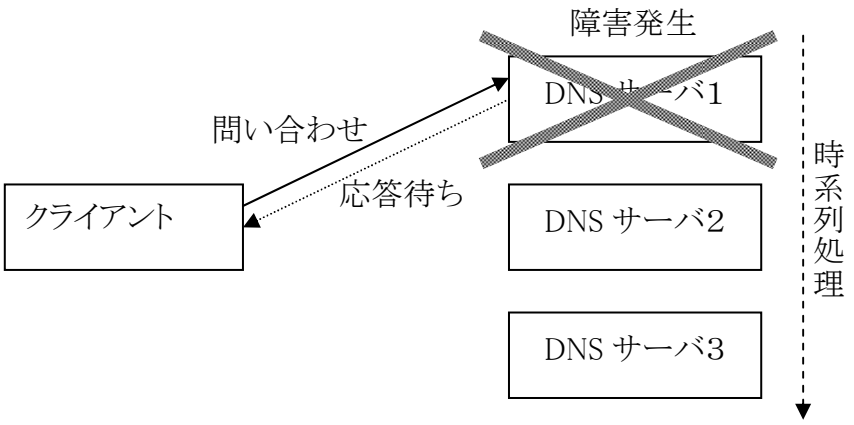


図 11 通常の DNS 参照における DNS サーバ障害の影響

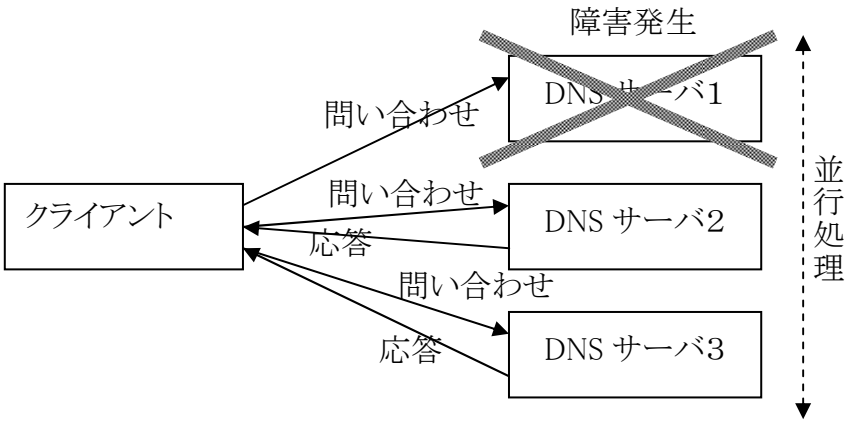


図 12 本名前解決モジュール使用時における DNS サーバ障害の影響

(3) 本項目における課題

近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールの基本機能は、複数 DNS サーバの応答検地である。また、一定時間応答のなかった DNS サーバに対して到達性が無いとみなして query を投げない、とする機能もある。

これにより、近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって効率的な名前解決を可能とし、また、ファイアウォール等で分断されたネットワーク環境においても適切な応答を選択可能となる。

複数の DNS サーバを登録し、到達可能でかつ問い合わせに対して有効な応答を返す DNS サーバからの応答を効率的に採用する構造を持ち、またすべての DNS サーバからの応答の質を随時監視することで到達可能性を動的に管理する機能を持つ名前解決モジュールを開発する。

(4) 実装の方針

近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールの構成を図 13 に示す。

本名前解決モジュールにおいては、名前解決を行うための参照先として複数の DNS サーバを登録することが可能であるものとする。本名前解決モジュールでは登録された DNS サーバごとに接続状態を管理するためのエンティティを用意し、各々のエンティティは他のエンティティとは独立に、割り当てられている DNS サーバの接続状態を管理することが可能であるものとする。

それぞれの DNS サーバの接続状態は、名前解決を行う際に参照のための問い合わせを送信し、それに対する当該 DNS サーバからの応答の有無や遅延を監視することによって動的に把握するものとする。

名前解決を行う際に、到達可能性があると判断されている状態の DNS サーバすべてに対し、並行して同時に問い合わせを送信する。これにより特定の DNS サーバの状態が到達可能から到達不可能に変化した場合のタイムアウト待ちによるネットワーク運用効率の低下を最小限に抑える効果を期待する。

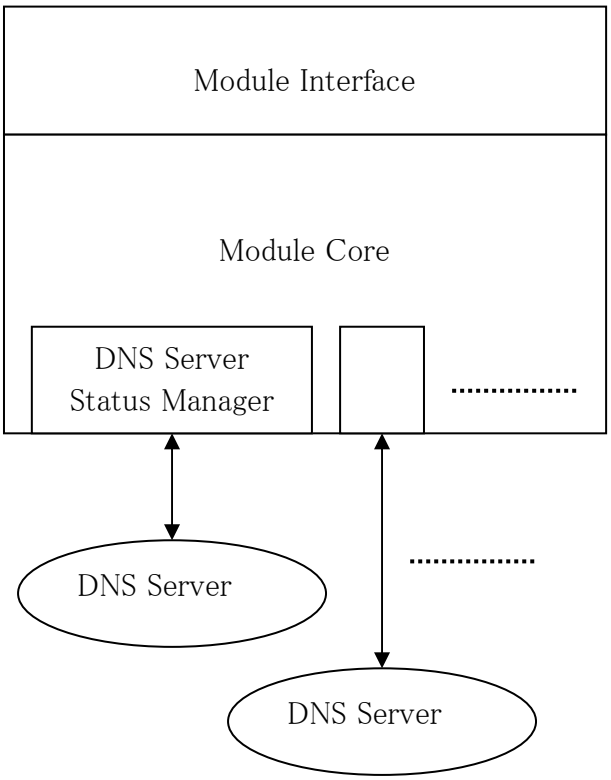


図 13 本名前解決モジュールの構成

(5) 試作開発の状況

ノードの名前を解決するための DNS サーバが近隣に複数存在することを認識した状態において、それぞれの DNS サーバへの到達性や問い合わせに対する応答の有用性を動的に管理することで、いずれかの DNS サーバに障害が発生した場合でも、タイムアウト待ちなどによって運用効率を低下させることなく継続的な名前解決を可能とする名前解決モジュールの試作開発を完了した。

(6) 本項目における成果

複数の DNS サーバを登録し、到達可能でかつ問い合わせに対して有効な応答を返す DNS サーバからの応答を効率的に採用する構造を持ち、またすべての DNS サーバからの応答の質を随時監視することで到達可能性を動的に管理する機能を持つ名前解決モジュール「dnsmodule」を開発した。

名前解決モジュール「dnsmodule」は次の手順で起動する。

dnsmodule -t <DNS サーバ 1> -t <DNS サーバ 2> ・・

名前解決モジュール「dnsmodule」の起動に先立ち、汎用名前解決エンジン「mng」を起動しておく必要がある。

(7) 実装内容の詳細

本名前解決モジュールでは、登録されている DNS サーバに対して以下の管理方法を用いることで、それぞれの DNS サーバへの到達可能性を監視することで、接続状態を管理するものとする。

- DNS サーバを複数指定し、それぞれの DNS サーバに対して接続状態を次のいずれかとして管理する。

REACHABLE 問い合わせに対する応答があった。

UNREACHABLE　問い合わせに対する応答が一定期間なかった。

STALE　一定期間以上 **UNREACHABLE** 状態が続いた。

- 初期状態としては、すべての DNS サーバが **REACHABLE** 状態であるものとみなす。
- 問い合わせは、**REACHABLE** 状態または **STALE** 状態である DNS サーバすべてに対して送信する。**STALE** 状態である DNS サーバは、問い合わせを送信すると同時に **UNREACHABLE** 状態とする。
- 問い合わせに対する応答があった DNS サーバは、すべて **REACHABLE** 状態とする。
- **REACHABLE** 状態であり、かつ問い合わせに対する応答が一定期間なかった DNS サーバは **UNREACHABLE** 状態とする。
- 一定期間以上 **UNREACHABLE** 状態であった DNS サーバは **STALE** 状態とする。

本名前解決モジュールにおける各 DNS サーバに対する管理状態の遷移を図 14 に示す。

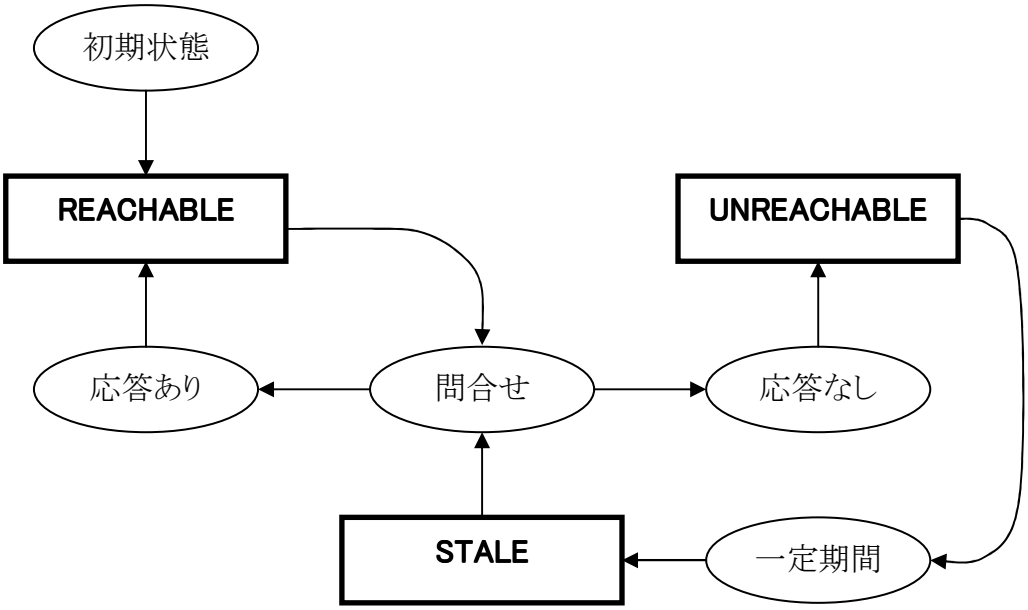


図 14 DNS サーバの管理状態遷移

5-1-3　複数のローカルデータベースを利用した名前解決がグローバルなDNS システムと透過的に利用可能な名前解決モジュール

(1) 試作開発の内容

セキュリティ上の観点から名前を外部に対して非公開とするために DNS に登録しない状態で運用するケースを想定し、ローカルなデータベースにのみ登録された名前を解決することが可能であると同時に、グローバルな DNS システムとも透過的に利用が可能となる名前解決モジュールを実装する。

具体的には、以下の項目について実装する。

- 複数データベースの利用可能性の実現

  異なる名前空間の集合に対しては異なるデータベースファイルが利用できるよう、複数のデータベースファイルを独立に扱うことを可能とする。

- 複数問い合わせ形の利用可能性の実現

  同一のデータベースファイル内において、複数の問い合わせ方を扱えることを可能とする。

(2) 背景

昨今セキュリティ上の脅威の一種として、DoS（Denial of Service）攻撃やDDoS（Distributed Denial of Service）攻撃などが流行している。IPv4 環境においては実質的な IP アドレスの数が限られているため、全アドレスに対する網羅的な攻撃が少なからず認められたが、IPv6 環境においては利用可能なIP アドレスの数が実質上無限に近いため、網羅的な攻撃よりはピンポイント的な攻撃のほうが主流となることが予想される。この場合 DNS に名前が登録されているノードは攻撃対象となる可能性が高いため、実際にサービスを提供するなど名前を公開する必要性があるノード以外に関しては、DNS への登録を行わず名前を公開しないという運用が行われることが少なくないものと思われる。

IPv6 では IP アドレスが非常に長く、ネットワークを利用する際にユーザが直接IP アドレスを指定するのは現実的とはいえない。そのため名前が公開されて

いないノードに対してアクセスをするために、ローカルなデータベースに名前と IP アドレスを登録する運用形態を想定することができる。またそのような運用形態の環境下においても、名前が公開されているノードに対しても並行してアクセスをする状況を考慮し、グローバルな DNS への問い合わせによる名前解決と透過的に利用可能であることも求められるだろう。

(3) 本項目における課題

セキュリティ上の脅威としてのピンポイント的な DoS 攻撃などを回避する目的で、通信相手となるノードの名前情報を DNS に登録しない状態で運用をするネットワーク環境において、ローカルなデータベースに登録した当該名前情報をその他の DNS 登録済みの名前情報と透過的に参照して利用することが可能な名前解決モジュールを開発する。

(4) 実装の方針

複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールの構成を図 15 に示す。

本名前解決モジュールにおいては、名前解決を行うための参照先として複数のローカルデータベースを登録することが可能であるものとする。ローカルデータベースのデータベースエンジンに関しては設計上基本的に制限を設けないが、本名前解決モジュールにおいては実装例として対応関係を記述したテキストファイルによるマッピングテーブルを使用するものとする。

登録されたローカルデータベースの情報は、名前解決モジュール起動時にテキストファイルから読み込み、メモリ上に蓄える形式で管理するものとする。

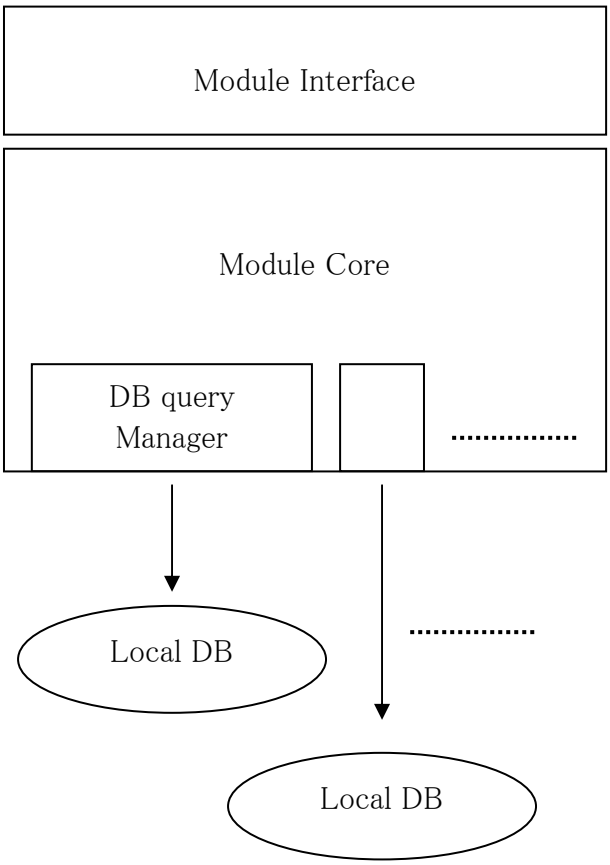


図 15 本名前解決モジュールの構成

(5) 試作開発の状況

セキュリティ上の観点から名前を外部に対して非公開とするために DNS に登録しない状態で運用するケースを想定し、ローカルなデータベースにのみ登録された名前を解決することが可能であると同時に、グローバルな DNS システムとも透過的に利用が可能となる名前解決モジュールの試作開発を完了した。

(6) 本項目における成果

セキュリティ上の脅威としてのピンポイント的な DoS 攻撃などを回避する目的で、通信相手となるノードの名前情報を DNS に登録しない状態で運用をするネットワーク環境において、ローカルなデータベースに登録した当該名前情報をその他の DNS 登録済みの名前情報と透過的に参照して利用することが可能な名前解決モジュール「localdbmodule」を開発した。

名前解決モジュール「localdbmodule」は次の手順で起動する。

localdbmodule <テキストファイル 1> <テキストファイル 2> ・・

名前解決モジュール「localdbmodule」の起動に先立ち、汎用名前解決エンジン「mng」を起動しておく必要がある。

(7) 実装内容の詳細

本名前解決モジュールでは、以下の手順によりローカルなデータベースにのみ登録された名前を解決することを可能とする。

- DNS ゾーンファイルに準じた形式で記述可能なテキストファイル形式のローカルデータベースを指定し、起動時に読み込む。ローカルデータベースは複数指定することが可能である。
- 問い合わせに対し、読み込んだローカルデータベースを検索して該当する名前情報を応答として採用する。

本名前解決モジュールを、他の DNS サーバへの問い合わせを行う名前解決

モジュールと併用することにより、ローカルなデータベースにのみ登録された名前とグローバルなDNSに登録された名前とを透過的に参照して解決することが可能となる。

ローカルデータベースとして読み込むテキストファイルは、次の形式で記述するものとする。

```
; <コメント>
<FQDN>                                IN A        <IPv4 アドレス>
<FQDN>                                IN AAAA     <IPv6 アドレス>
<IP アドレス（逆順）>.in-addr.arpa.  IN PTR      <FQDN>
```

5-2　統合動作検証

まず試作開発した汎用名前解決エンジンに、近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールと、複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールとをそれぞれ単独で組み合わせ、各名前解決モジュールが適切な挙動をとることを確認するための単体動作検証を行った。

次に平成14年度と平成15年度に試作した汎用名前解決エンジンと名前解決モジュール群に対し、汎用名前解決エンジン上ですべての名前解決モジュールを統合して動作させ、その有効性を確認した。

また本研究開発では、基本的に FreeBSD-4.7R＋KAME-SNAP030127 という OS 環境上での試作開発を進めてきたが、IPv6 の標準仕様にしたがった標準的な機能のみを利用しているため、必ずしも上記 OS に依存した実装というわけではない。本研究開発の目的から考えても、本名前解決システムがプラットフォームに依存しない形で実装できることが望ましいと考えられる。そこで本来の試作開発の主旨とは若干ずれるが、本研究開発期間中に試作開発を進めてきた名前解決システムを Linux OS 環境上に移植し、簡単な動作検証も行った。

5-2-1　近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールと汎用名前解決エンジンの組み合わせ

(1) 単体動作検証の内容

5-1-1　で実装した汎用名前解決エンジン上に、5-1-2　で実装した近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールを組み合わせ、名前解決機能の動作検証を行う。

(2) 背景

ネットワークを運用する上で、DNS はインフラストラクチャとして非常に重要な役割を担っている。ネットワークが依存している DNS サーバに障害が発生した場合、それがネットワーク上で稼動する各種アプリケーションの利用に実質的な影響を及ぼすことは少なくない。したがって通常ネットワークを構築する際には、DNS サーバの配置に冗長性を持たせることによって、いずれかの DNS サーバに障害が発生したとしてもネットワーク全体の運用を継続できるような構成が必須となる。

しかしながらアプリケーションを稼動させるクライアント上で利用される一般的な OS では、参照可能な DNS サーバの数はあまり多くなく、一台からせいぜい三台程度である。また複数の DNS サーバを参照可能な OS であっても、参照する際には優先順位を設定するようになっており、優先度の高い DNS サーバへの参照を先に行って、応答がない場合にのみ次の優先度が設定された DNS サーバへの参照を行うという振る舞いをするように実装されていることが多い。この挙動により、冗長性を持たせるためにネットワーク上に複数の DNS サーバを配置させていた場合であっても、優先度の高い特定の DNS サーバへの参照が集中することと、優先度の高い DNS サーバに障害が発生した場合に多くのアプリケーションにおいて当該 DNS サーバに対する参照のタイムアウト待ちが同時に発生してネットワークシステム全体の稼動効率が著しく下がることは避けられない。

(3) 本項目における課題

近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによっ

て、効率的な名前解決を可能とする名前解決モジュールの基本機能は、複数 DNS サーバの応答検地である。また、一定時間応答のなかった DNS サーバに対して UNREACHABLE とみなして query を投げない、とする機能もある。

これにより、近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって効率的な名前解決を可能とし、また、ファイアウォール等で分断されたネットワーク環境においても適切な応答を選択可能となる。

複数の DNS サーバを登録し、到達可能でかつ問い合わせに対して有効な応答を返す DNS サーバからの応答を効率的に採用する構造を持ち、またすべての DNS サーバからの応答の質を随時監視することで到達可能性を動的に管理する機能を持つ名前解決モジュールの動作を評価、検証する。

(4) 検証作業の方針

本検証作業においては、本名前解決モジュールに登録されている DNS サーバがすべて稼動して到達可能である場合、稼動して到達可能である DNS サーバと到達不可能である DNS サーバが混在している場合、すべて到達不可能である場合のそれぞれの状況下において名前解決を試み、到達可能な DNS サーバが最低1台稼動していれば運用効率を下げずに名前の解決が可能であることを検証するものとする。

また NI Queries による名前解決モジュール（nimodule）と組み合わせた動作確認・評価も行う。汎用名前解決エンジンに本名前解決モジュールと NI Queries による名前解決モジュールを組み合わせることにより、前述の機能に加えて DNS サーバが利用できない環境でも近隣のノードの名前を解決可能となり、すべての DNS サーバに障害が発生している状況でも運用効率を下げずに名前の解決を可能とする環境が構築可能であることを検証するものとする。

(5) 単体動作検証の状況

5-1-2 で試作開発を行った近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールを 5-1-1 で実装した汎用名前解決エンジンと接続した環境で単体動作検証を行い、正常に動作していることを確認した。

(6) 本項目における成果

近隣のDNSサーバ障害発生時に適応的にDNSサーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールを汎用名前解決エンジンと組み合わせて使用した状態で名前解決試験を実施し、以下の結果を得た。

- 近隣のDNSサーバ障害発生時に適応的にDNSサーバを選択することによって効率的な名前解決を可能とする名前解決モジュールと汎用名前解決エンジンとの組み合わせ

  →単体動作試験環境において名前解決に成功した。

- 近隣のDNSサーバ障害発生時に適応的にDNSサーバを選択することによって効率的な名前解決を可能とする名前解決モジュールと、NI Queries による名前解決モジュールとによる汎用名前解決エンジンとの組み合わせ

  →単体動作試験環境において名前解決に成功し、query 送出についての最適化ができる可能性があることが確認できた。

(7) 検証作業の詳細

本動作検証は、第一段階として図 16 に示す単体動作検証用ネットワーク環境上で行った。

単体動作検証用ネットワーク(1－A)の機器構成を次に示す。

- cl1
  FreeBSD-4.7R＋KAME-SNAP030127
- ns1・ns2
  Red Hat Linux 3.2.2-5
- ns1・ns2
  BIND-9.2.2

ここで、汎用名前解決エンジンと近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって、効率的な名前解決を可能とする名前解決モジュールを cl1 にインストールし、次の動作環境の下で動作検証を行った。

- ns1・ns2 のネームサーバ設定

  - cl1.tao-mng.org に対する AAAA レコードとして、2001:218:4eb:1::2 を設定。

- cl1 上の汎用名前解決エンジンの起動

  - ./mng
  - ./dnsmodule -t 2001:218:4eb:1::53 -t 2001:218:4eb:1::54

動作検証は、以下の手順で行った。

- cl1 から、cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ（ns1 または ns2）に問い合わせを行った結果である。

- ns1 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ（ns2）に問い合わせを行った結果である。

- ns2 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ（ns1）に問い合わせを行った結果である。

- ns1 および ns2 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスが取得できなかった。これは dnsmodule によって問い合わせた DNS サーバからの応答がいずれからもなかったためである。

次に第二段階として図 17 に示す単体動作検証用ネットワーク環境上で動作検証を行った。

単体動作検証用ネットワーク(1－B)の機器構成を次に示す。

・ cl1・cl2・cl3
  FreeBSD-4.7R＋KAME-SNAP030127
・ ns1・ns2・ns3
  Red Hat Linux 3.2.2-5
・ rt
  Linux
・ ns1・ns2・ns3
  BIND-9.2.2

ここで、汎用名前解決エンジンと各名前解決モジュールを cl1・cl2 にインストールし、次の動作環境の下で動作検証を行った。

- ns1・ns2 のネームサーバ設定
  - cl1.tao-mng.org に対する AAAA レコードとして、2001:218:4eb:1::2 を設定。
- ns3 のネームサーバ設定
  - cl3.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl2 上の汎用名前解決エンジンの起動
  - ./mng
  - ./dnsmodule -t 2001:218:4eb:2::53 -t 2001:218:4eb:2::54 -t 2001:218:4eb:1::53
  - ./nimodule

動作検証は、以下の手順で行った。

・ ns1・ns2・ns3 が到達可能な状態で、tao-mng.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ(ns1 または ns2)に問い合

わせを行った結果である。

- 上記状態から ns1 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得したが、1回目は名前解決に12秒、2回目は6秒かかり、3回目はすぐに終了した。これは dnsmodule によってDNS サーバ(ns1)に問い合わせを行った結果であるが、1回目は ns2・ns3 の状態が **REACHABLE** であったのに対し、2回目は **UNREACHABLE** になったために名前解決にかかる時間が短くなったためである。また3回目がすぐに終了したのは ns2・ns3 とも状態が **UNREACHABLE** であったために query 自体が送出されなかったことによる。

- 上記状態から ns2 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得したが、1回目は名前解決に11 秒、2回目は6秒かかり、3回目はすぐに終了した。これは dnsmodule によってDNS サーバ(ns2)に問い合わせを行った結果で、また上記と同様の理由により時間がかかったものである。ns2・ns3 は状態が **STALE** であったためにどちらにも query が送出され、ns2 からの応答があったことで ns2 の状態が **REACHABLE** になった。このとき **REACHABLE** であった ns1 は、応答がなくなったことによって **UNREACHABLE**、**STALE** と状態が変化した。

- 上記状態から ns3 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして fe80:210:dcff:fe1f:a0db(リンクローカルアドレス)を取得した。これは nimodule によって近隣ノードとしての名前解決を行った結果である。ns1・ns3 は状態が **STALE** であったためにどちらにも query が送出され、ns3 からの応答があったことで ns3 の状態が **REACHABLE** になった。このとき **REACHABLE** であった ns2 は、応答がなくなったことによって **UNREACHABLE**、**STALE** と状態が変化した。

- ns1・ns2・ns3 が到達可能な状態で、tao-mng.org 上の cl2 から cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは dnsmodule によってDNS サーバ(ns3)に問い合わせを行った結果である。

・ 上記状態から ns1 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスが取得できなかった。これは dnsmodule による名前解決も nimodule による名前解決もできなかった結果である。1回目は名前解決不可の結果が出るまでに23秒かかったが、2回目以降はすぐ終了した。これは名前解決ができない DNS サーバの状態を1回目で認識したためである。

・ 上記状態から ns2 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスが取得できなかった。これは dnsmodule による名前解決も nimodule による名前解決もできなかった結果である。1回目は名前解決不可の結果が出るまでに23秒かかったが、2回目以降はすぐ終了した。これは名前解決ができない DNS サーバの状態を1回目で認識したためである。

・ 上記状態から ns3 のみが到達可能な状態に変更し、tao-mng.org 上の cl2 から cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは dnsmodule によってDNS サーバ(ns3)に問い合わせを行った結果である。1回目は名前解決に12秒かかったが、2回目以降はすぐ終了した。これは1回目の query で ns3 の状態を REACHABLE と認識したためである。

・ ns1・ns2・ns3 が到達可能な状態で、external.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によってDNS サーバ(ns1 または ns2)に問い合わせを行った結果である。

・ 上記状態から ns1 のみが到達可能な状態に変更し、external.org 上の

cl2 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得したが、1回目は名前解決に12秒、2回目は6秒かかり、3回目はすぐに終了した。これは dnsmodule によってDNSサーバ(ns1)に問い合わせを行った結果であるが、1回目は ns2・ns3 の状態が REACHABLE であったのに対し、2回目は UNREACHABLE になったために名前解決にかかる時間が短くなったためである。また3回目がすぐに終了したのは ns2・ns3 とも状態が UNREACHABLE であったために query 自体が送出されなかったことによる。

- 上記状態から ns2 のみが到達可能な状態に変更し、external.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得したが、1回目は名前解決に11 秒、2回目は5秒かかり、3回目はすぐに終了した。これは dnsmodule によってDNSサーバ(ns2)に問い合わせを行った結果で、また上記と同様の理由により時間がかかったものである。ns2・ns3 は状態が STALE であったためにどちらにも query が送出され、ns2 からの応答があったことで ns2 の状態が REACHABLE になった。このとき REACHABLE であった ns1 は、応答がなくなったことによって UNREACHABLE、STALE と状態が変化した。

- 上記状態から ns3 のみが到達可能な状態に変更し、external.org 上の cl2 から cl1.tao-mng.org に対する名前解決を行った。

  →cl1.tao-mng.org に対するアドレスが取得できなかった。これは dnsmodule による名前解決も nimodule による名前解決もできなかった結果である。1回目は名前解決不可の結果が出るまでに24秒かかったが、2回目以降はすぐ終了した。これは名前解決ができない DNS サーバの状態を1回目で認識したためである。

- ns1・ns2・ns3 が到達可能な状態で、external.org 上の cl2 から cl3.external.org に対する名前解決を行った。

  →cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは dnsmodule によってDNSサーバ(ns3)に問い合わせを行った結果である。

- 上記状態から ns1 のみが到達可能な状態に変更し、external.org 上の

cl2 から cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスとして fe80:210:dcff:fe2c:9c46（リンクローカルアドレス）を取得した。これは nimodule によって近隣ノードとしての名前解決を行った結果である。1回目は名前解決に18秒かかったが、2回目以降はすぐ終了した。これは名前解決ができない DNS サーバの状態を1回目で認識したためである。

- 上記状態から ns2 のみが到達可能な状態に変更し、external.org 上の cl2 から cl3.external.org に対する名前解決を行った。

  →cl3.external.org に対するアドレスとして fe80:210:dcff:fe2c:9c46（リンクローカルアドレス）を取得した。これは nimodule によって近隣ノードとしての名前解決を行った結果である。1回目は名前解決に18秒かかったが、2回目以降はすぐ終了した。これは名前解決ができない DNS サーバの状態を1回目で認識したためである。

- 上記状態から ns3 のみが到達可能な状態に変更し、external.org 上の cl2 から cl3.external.org に対する名前解決を行った。

  →cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは dnsmodule によって DNS サーバ（ns3）に問い合わせを行った結果である。1回目は名前解決に12秒かかったが、2回目以降はすぐ終了した。これは1回目の query で ns3 の状態を **REACHABLE** と認識したためである。

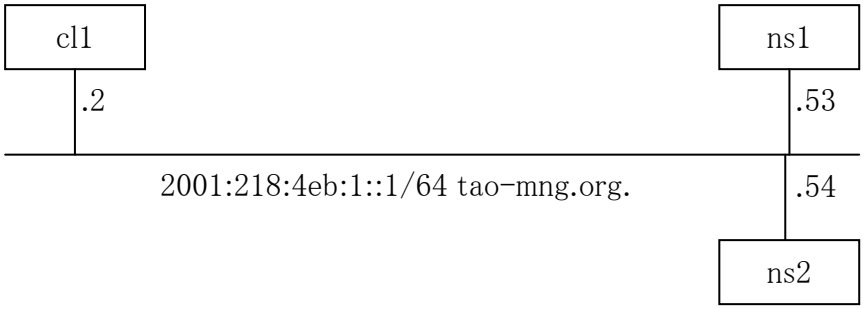


図 16 単体動作検証用ネットワーク環境(1－A)

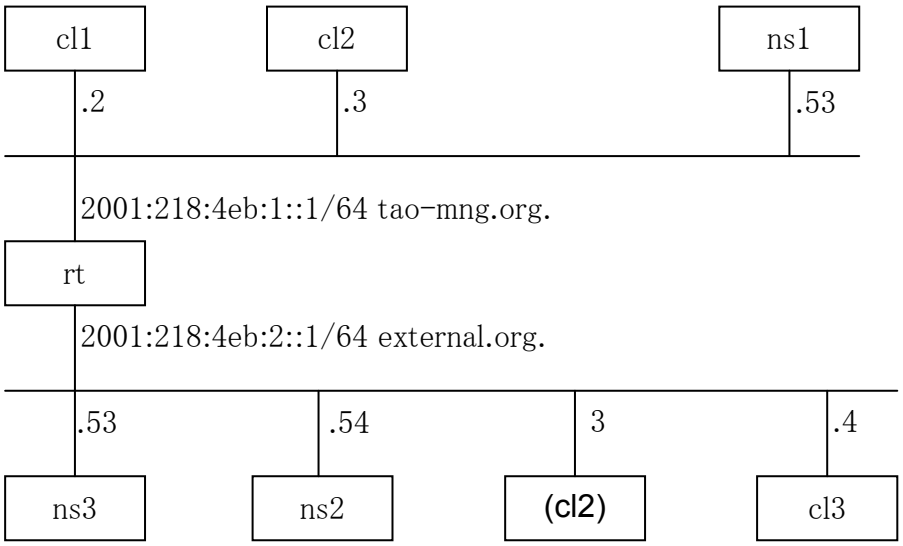


図 17 単体動作検証用ネットワーク環境(1－B)

(8) 検証作業の具体例

以下、検証結果の具体例を示す。ここでは第二段階において、cl2 から cl1.tao-mng.org または cl3.external.org に対する名前解決を行った場合を例にあげる。なお log の status は、それぞれ

1: **REACHABLE**
2: **UNREACHABLE**
3: **STALE**

を表している。

```
cl2# date ; host cl1.tao-mng.org ; date
Mon Sep 22 16:01:51 JST 2003
cl1.tao-mng.org has address 192.168.1.2
cl1.tao-mng.org has address 2001:218:4eb:1::2
Mon Sep 22 16:01:51 JST 2003
cl2# grep status log
cl2#
```

上記は、DNS サーバから cl1.tao-mng.org に対するアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl2# date ; host cl1.tao-mng.org ; date
Mon Sep 22 16:03:44 JST 2003
cl1.tao-mng.org has address 2001:218:4eb:1::2
Mon Sep 22 16:03:56 JST 2003
cl2# date ; host cl1.tao-mng.org ; date
Mon Sep 22 16:03:59 JST 2003
cl1.tao-mng.org has address 2001:218:4eb:1::2
Mon Sep 22 16:04:05 JST 2003
cl2# date ; host cl1.tao-mng.org ; date
Mon Sep 22 16:04:53 JST 2003
cl1.tao-mng.org has address 2001:218:4eb:1::2
Mon Sep 22 16:04:53 JST 2003
cl2# grep status log
Sep 22 16:04:05: dsSetStatus: 2001:218:4eb:2::54 status changed 1 to 2
Sep 22 16:04:05: dsSetStatus: 2001:218:4eb:2::53 status changed 1 to 2
Sep 22 16:04:16: dsSetStatus: 2001:218:4eb:2::54 status changed 2 to 3
```

```
Sep 22 16:04:16: dsSetStatus: 2001:218:4eb:2::53 status changed 2 to 3
```

上記は DNS サーバを一部停止させた状態の実行結果で、1回目の実行に12秒、2回目の実行に6秒かかり、3回目はすぐに終了していることが確認できる。

```
cl2# date ; host cl3.external.org ; date
Mon Sep 22 16:19:54 JST 2003
Host not found, try again.
Mon Sep 22 16:20:17 JST 2003
cl2# date ; host cl3.external.org ; date
Mon Sep 22 16:20:18 JST 2003
Host not found, try again.
Mon Sep 22 16:20:18 JST 2003
cl2# date ; host cl3.external.org ; date
Mon Sep 22 16:20:20 JST 2003
Host not found, try again.
Mon Sep 22 16:20:20 JST 2003
cl2# grep status log
Sep 22 16:20:17: dsSetStatus: 2001:218:4eb:1::53 status changed 1 to 2
Sep 22 16:20:17: dsSetStatus: 2001:218:4eb:2::54 status changed 1 to 2
Sep 22 16:20:17: dsSetStatus: 2001:218:4eb:2::53 status changed 1 to 2
Sep 22 16:20:28: dsSetStatus: 2001:218:4eb:1::53 status changed 2 to 3
Sep 22 16:20:28: dsSetStatus: 2001:218:4eb:2::54 status changed 2 to 3
Sep 22 16:20:28: dsSetStatus: 2001:218:4eb:2::53 status changed 2 to 3
Sep 22 16:21:24: dsSetStatus: 2001:218:4eb:1::53 status changed 3 to 1
```

上記は名前解決ができなかった状態の実行結果で、1回目の実行には23秒かかっているが、2回目以降はすぐに終了していることが確認できる。

5-2-2　複数のローカルデータベースを利用した名前解決がグローバルなDNS システムと透過的に利用可能な名前解決モジュールと汎用名前解決エンジンの組み合わせ

(1) 単体動作検証の内容

5-1-1 で実装した汎用名前解決エンジン上に、5-1-3 で実装した複数のローカルデータベースを利用した名前解決がグローバルなDNSシステムと透過的に利用可能な名前解決モジュールを組み合わせ、名前解決機能の動作検証を行う。

(2) 背景

昨今セキュリティ上の脅威の一種として、DoS（Denial of Service）攻撃やDDoS（Distributed Denial of Service）攻撃などが流行している。IPv4 環境においては実質的な IP アドレスの数が限られているため、全アドレスに対する網羅的な攻撃が少なからず認められたが、IPv6 環境においては利用可能なIPアドレスの数が実質上無限に近いため、網羅的な攻撃よりはピンポイント的な攻撃のほうが主流となることが予想される。この場合 DNS に名前が登録されているノードは攻撃対象となる可能性が高いため、実際にサービスを提供するなど名前を公開する必要性があるノード以外に関しては、DNS への登録を行わず名前を公開しないという運用が行われることが少なくないものと思われる。

IPv6 ではIPアドレスが非常に長く、ネットワークを利用する際にユーザが直接 IP アドレスを指定するのは現実的とはいえない。そのため名前が公開されていないノードに対してアクセスをするために、ローカルなデータベースに名前と IP アドレスを登録する運用形態を想定することができる。またそのような運用形態の環境下においても、名前が公開されているノードに対しても並行してアクセスをする状況を考慮し、グローバルな DNS への問い合わせによる名前解決と透過的に利用可能であることも求められるだろう。

(3) 本項目における課題

セキュリティ上の脅威としてのピンポイント的な DoS 攻撃などを回避する目的で、通信相手となるノードの名前情報を DNS に登録しない状態で運用をするネットワーク環境において、ローカルなデータベースに登録した当該名前情

報をその他の DNS 登録済みの名前情報と透過的に参照して利用することが可能な名前解決モジュールの動作を評価、検証する。

(4) 検証作業の方針

本検証作業においては、本名前解決モジュールに登録されているローカルデータベース上のエントリに対する名前解決と、通常の DNS サーバに対する名前解決とが、同一の環境下において同時並行的にかつ透過的に可能であることを検証するものとする。

(5) 単体動作検証の状況

5-1-3 で試作開発を行った複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールを 5-1-1 で実装した汎用名前解決エンジンと接続した環境で単体動作検証を行い、正常に動作していることを確認した。

(6) 本項目における成果

複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールを汎用名前解決エンジンと組み合わせて使用した状態で名前解決試験を実施し、以下の結果を得た。

- 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールと汎用名前解決エンジンとの組み合わせ

  →単体動作試験環境において名前解決に成功した。

(7) 検証作業の詳細

本動作検証は、図 18 に示す単体動作検証用ネットワーク環境上で行った。

単体動作検証用ネットワーク(2)の機器構成を次に示す。

・ cl2・cl3
　FreeBSD-4.7R＋KAME-SNAP030127
・ ns3
　Red Hat Linux 3.2.2-5
・ ns3
　BIND-9.2.2

ここで、汎用名前解決エンジンと複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールを cl2 にインストールし、次の動作環境の下で動作検証を行った。

- ns3 のネームサーバ設定
  - cl3.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl2 のローカルデータベース(localdb-test.txt)設定
  - cl3-local.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl2 上の汎用名前解決エンジンの起動
  - ./mng
  - ./nullmodule -t 2001:218:4eb:2::53
  - ./localdbmodule localdb-test.txt

動作検証は、以下の手順で行った。

・ cl2 から、cl3-local.external.org に対する名前解決を行った。

→cl3-local.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは localdbmodule によってローカルデータベース localdb-test.txt に登録された情報によって名前解決を行った結果である。

・ cl2 から、cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得し

た。これはnullmoduleによってDNSサーバ(ns3)に問い合わせを行った結果である。

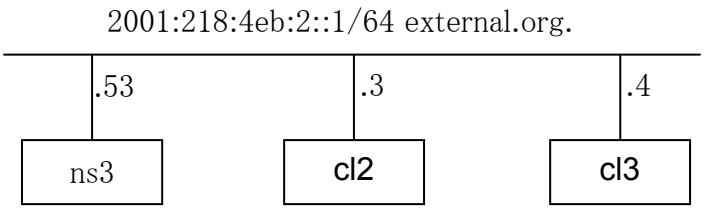


図 18 単体動作検証用ネットワーク環境(2)

(8) 検証作業の具体例

以下、検証結果の具体例を示す。ここでは cl2 から cl3-local.external.org または cl3.external.org に対する名前解決を行った場合を例にあげる。

```
cl2# time ping6 -c 2 cl3-local.external.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:2::3 --> 2001:218:4eb:2::4
16 bytes from 2001:218:4eb:2::4, icmp_seq=0 hlim=64 time=0.625 ms
16 bytes from 2001:218:4eb:2::4, icmp_seq=1 hlim=64 time=0.288 ms

--- cl3-local.external.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.288/0.457/0.625/0.168 ms
0.000u 0.001s 0:01.01 0.0%          0+0k 0+0io 0pf+0w
```

上記は、ローカルデータベースから cl3-local.external.org に対するアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl2# time ping6 -c 2 cl3.external.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:2::3 --> 2001:218:4eb:2::4
16 bytes from 2001:218:4eb:2::4, icmp_seq=0 hlim=64 time=0.568 ms
16 bytes from 2001:218:4eb:2::4, icmp_seq=1 hlim=64 time=0.181 ms

--- cl3.external.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.181/0.374/0.568/0.194 ms
0.000u 0.001s 0:01.00 0.0%          0+0k 0+0io 0pf+0w
```

上記は、DNS サーバから cl3.external.org に対するアドレスを取得した場合の実行結果である。実行はすぐに終了している。

5-2-3　すべての名前解決モジュールと汎用名前解決エンジンの組み合わせ

(1) 統合動作検証の内容

5-1-1　で実装した汎用名前解決エンジン上に、平成14年度の研究開発において実装した三種類の名前解決モジュール、5-1-2　で実装した名前解決モジュール、5-1-3　で実装した名前解決モジュールをすべて統合し、それぞれの名前解決モジュールを利用した名前解決機能の動作検証を行う。

(2) 背景

現在インターネットを利用するほぼすべてのアプリケーションは DNS による名前解決を利用している。通信相手の名前を解決して IP アドレスを取得する名前解決システムはインターネット上で通信相手を特定するために必要不可欠な機構であり、インターネット上でアプリケーションを利用するためには DNS による名前解決システムがプラットフォームに依存しない形で安定して稼動している必要がある。

通常の OS では DNS による名前解決を行うための共通のライブラリを提供している。また RFC2553 ではこのライブラリの基本的な API を定めており、OS に依存しない汎化された API を利用できるようになっている。これらの API は通常近傍にある DNS cache サーバと通信し、名前と IP アドレスの相互変換を行う形になっている。このライブラリ群は通常リゾルバライブラリと呼ばれる。また DNS cache サーバとの通信は port 53 を利用して UDP あるいは TCP で行われる。これらのプロトコルは RFC1035、RFC2671 で定められている。

一方近年になり、特定のネットワーク上に固定的に接続された計算機環境だけではなく、異なるネットワーク間をノードが移動するモバイル環境も一般的になりつつある。特定の DNS cache サーバを固定的に設定しておけば十分であった固定環境と違い、モバイル環境においては移動先ネットワーク上で利用可能な DNS サーバを探索する必要があったり、DNS が利用できない環境で近隣ノードの名前解決を行ったりする必要が生じる。

あるいはセキュリティやプライバシーを考慮したネットワーク環境において、ファイアウォールで分断されたいずれのネットワークにおいても適切な名前解決を行えるような機能も必要となることがある。

(3) 本項目における課題

一般のネットワークアプリケーションが名前解決を行う際に標準的に使用するDNS cache サーバとの通信機能を利用することで、さまざまなモバイル環境やセキュリティを考慮したネットワーク環境で利用可能な名前解決メカニズムをそれぞれの環境ごとに用意し、それらのメカニズムを統合的に処理することで環境に応じた適切な名前の解決を行うための汎用名前解決機構の試作開発成果物を対象として、統合的な動作の検証を行うことで各名前解決メカニズムの有用性を評価、検証する。

(4) 検証作業の方針

本検証作業においては、単一の汎用名前解決エンジンに対して複数の名前解決モジュールを組み合わせた環境を用意し、その環境下で個々の名前解決モジュールがそれぞれの機能を適切に提供することが可能であるか、および複数の名前解決モジュールの機能を同時に利用しようとしたときに適切な応答が選択されるかを検証し、それらの結果として汎用名前解決システムの運用環境下における有用性を確認することを目的とする。

(5) 統合動作検証の状況

平成14年度の研究開発において試作開発を行った三種類の名前解決モジュール、5-1-2 で試作開発を行った名前解決モジュール、5-1-3 で試作開発を行った各名前解決モジュールをすべて 5-1-1 で実装した汎用名前解決エンジンと接続した環境で統合動作検証を行い、正常に動作していること、および機能の有効性を確認した。

(6) 本項目における成果

各名前解決モジュールを汎用名前解決エンジンと組み合わせて使用した状態で名前解決試験を実施し、以下の結果を得た。

- DNS が利用できない環境でも近隣のノードの名前を解決可能な名前解決モジュール

  →統合動作試験環境において名前解決に成功した。

- 近隣の DNS サーバを動的に発見可能な名前解決モジュール

  →統合動作試験環境において名前解決に成功した。

- 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって効率的な名前解決を可能とする名前解決モジュール

  →統合動作試験環境において名前解決に成功した。

- 複数のサーバから異なる応答を受けた場合に適切な応答を選択可能な名前解決モジュール

  →統合動作試験環境において名前解決に成功した。

- 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュール

  →統合動作試験環境において名前解決に成功した。

- 各名前解決モジュールにそれぞれ優先度を指定して起動することによる汎用名前解決エンジンの名前解決モジュール選択機能

  →統合動作試験環境において、優先度に従った名前解決に成功した。

(7) 検証作業の詳細

本動作検証は、図 19 に示す統合動作検証用ネットワーク環境上で行った。ここで、cl2 は tao-mng.org から external.org へ移動するものとする。

統合動作検証用ネットワークの機器構成を次に示す。

- cl1・cl2・cl3
  FreeBSD-4.7R＋KAME-SNAP030127
- ns1・ns2・ns3
  Red Hat Linux 3.2.2-5
- rt
  Linux
- ns1・ns2・ns3
  BIND-9.2.2

ここで、汎用名前解決エンジンと各名前解決モジュールを cl1・cl2 にインストールし、次の動作環境の下で動作検証を行った。

- ns1・ns2 のネームサーバ設定
  - cl1.tao-mng.org に対する AAAA レコードとして、2001:218:4eb:1::2 を設定。
- ns3 のネームサーバ設定
  - cl3.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl2 のローカルデータベース(localdb-test.txt)設定
  - cl3-local.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl1・cl2 上の汎用名前解決エンジンの起動
  - ./mng
  - ./nimodule
  - ./nullmodule -t fec0:0:0:ffff:1
  - ./nullmodule -t fec0:0:0:ffff:2
  - ./nullmodule -t 2001:218:4eb:2::53(cl1 上でのみ起動)
  - ./dnsmodule -t 2001:218:4eb:1::53 -t 2001:218:4eb:1::54
  - ./localdbmodule localdb-test.txt(cl2 上でのみ起動)

動作検証は、以下の手順で行った。

・ DNS が利用できない環境でも近隣のノードの名前を解決可能な名前解決モジュールの動作検証

cl1 から、cl2.tao-mng.org に対する名前解決を行った。

→cl2.tao-mng.org に対するアドレスとして、fe80::210:dcff:fe2c:dff5(リンクローカルアドレス)を取得した。これは nimodule によって近隣ノードとしての名前解決を行った結果である。

・ 近隣の DNS サーバを動的に発見可能な名前解決モジュールの動作検証

tao-mng.org 側に接続した cl2 から、cl1.tao-mng.org および cl3.external.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして、2001:218:4eb:1::2 を取得し、cl3.external.org に対するアドレスは取得できなかった。これは nullmodule によって近隣の DNS サーバ(ns1 または ns2)を発見した結果である。

external.org 側に接続した cl2 から、cl1.tao-mng.org および cl3.external.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスは取得できず、cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは nullmodule によって近隣の DNS サーバ(ns3)を発見した結果である。

・ 近隣の DNS サーバ障害発生時に適応的に DNS サーバを選択することによって効率的な名前解決を可能とする名前解決モジュールの動作検証

cl1 から、cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ(ns1 または ns2)に問い合わせを行った結果である。

ns1 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ(ns2)に問い合わせを行った結果である。

ns2 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして 2001:218:4eb:1::2 を取得した。これは dnsmodule によって DNS サーバ(ns1)に問い合わせを行った結果である。

ns1 および ns2 の tao-mng.org への接続を切断した状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして fe80::210:dcff:fe1f:a0db(リ

ンクローカルアドレス)を取得した。これは dnsmodule によって問い合わせた DNS サーバからの応答がいずれからもなかったため、代わりに nimodule によって近隣ノードとしての名前解決を行った結果である。

・ 複数のサーバから異なる応答を受けた場合に適切な応答を選択可能な名前解決モジュールの動作検証

cl1 から、cl3.external.org に対する名前解決を行った。

→cl3.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。ns1 および ns2 への問い合わせに対しては cl3.external.org に対するアドレスは存在しないという応答が返されるが、ns3 への問い合わせに対しては適切なアドレスが返されるため、その応答を選択した結果である。

・ 複数のローカルデータベースを利用した名前解決がグローバルな DNS システムと透過的に利用可能な名前解決モジュールの動作検証

cl2 から、cl3-local.external.org に対する名前解決を行った。

→cl3-local.external.org に対するアドレスとして 2001:218:4eb:2::4 を取得した。これは localdbmodule によってローカルデータベース localdb-test.txt に登録された情報によって名前解決を行った結果である。

・ 各名前解決モジュールにそれぞれ優先度を指定して起動することによる汎用名前解決エンジンの名前解決モジュール選択機能の動作検証

cl1.tao-mng.org に対する AAAA レコードを、それぞれ次のように設定しなおした。

- ns1 のネームサーバ・・2001:218:4eb:10::2
- ns2 のネームサーバ・・2001:218:4eb:11::2
- cl1 のローカルデータベース・・2001:218:4eb:12::2

そののち cl1 上の汎用名前解決エンジンを次のように起動しなおした。ここで各名前解決モジュールの起動時に、-P の後に指定している数値は優先度を表す。数値が大きいほど当該名前解決モジュールの優先度が高いものとする。

- ./mng
- ./nimodule -P 2
- ./nullmodule -P 1 -t 2001:218:4eb:1::53
- ./dnsmodule -P 1 -t 2001:218:4eb:1::54
- ./localdbmodule -P 1 localdb-test.txt

cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして、fe80::210:dcff:fe1f:a0db（リンクローカルアドレス）を取得した。これは他の名前解決モジュールよりも優先度の高い nimodule によって近隣ノードとしての名前解決を行った結果である。

nimodule の優先度を 1 に下げ、nullmodule の優先度を 2 に上げた状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして、2001:218:4eb:10::2 を取得した。これは他の名前解決モジュールよりも優先度の高い nullmodule によって、ns1 に対する問い合わせの応答による名前解決を行った結果である。

nullmodule の優先度を 1 に下げ、dnsmodule の優先度を 2 に上げた状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして、2001:218:4eb:11::2 を取得した。これは他の名前解決モジュールよりも優先度の高い dnsmodule によって、ns2 に対する問い合わせの応答による名前解決を行った結果である。

dnsmodule の優先度を 1 に下げ、localdbmodule の優先度を 2 に上げた状態で、cl1 から cl1.tao-mng.org に対する名前解決を行った。

→cl1.tao-mng.org に対するアドレスとして、2001:218:4eb:12::2 を取得した。これは他の名前解決モジュールよりも優先度の高い localmodule によって、ns2 に対する問い合わせの応答による名前解決を行った結果である。

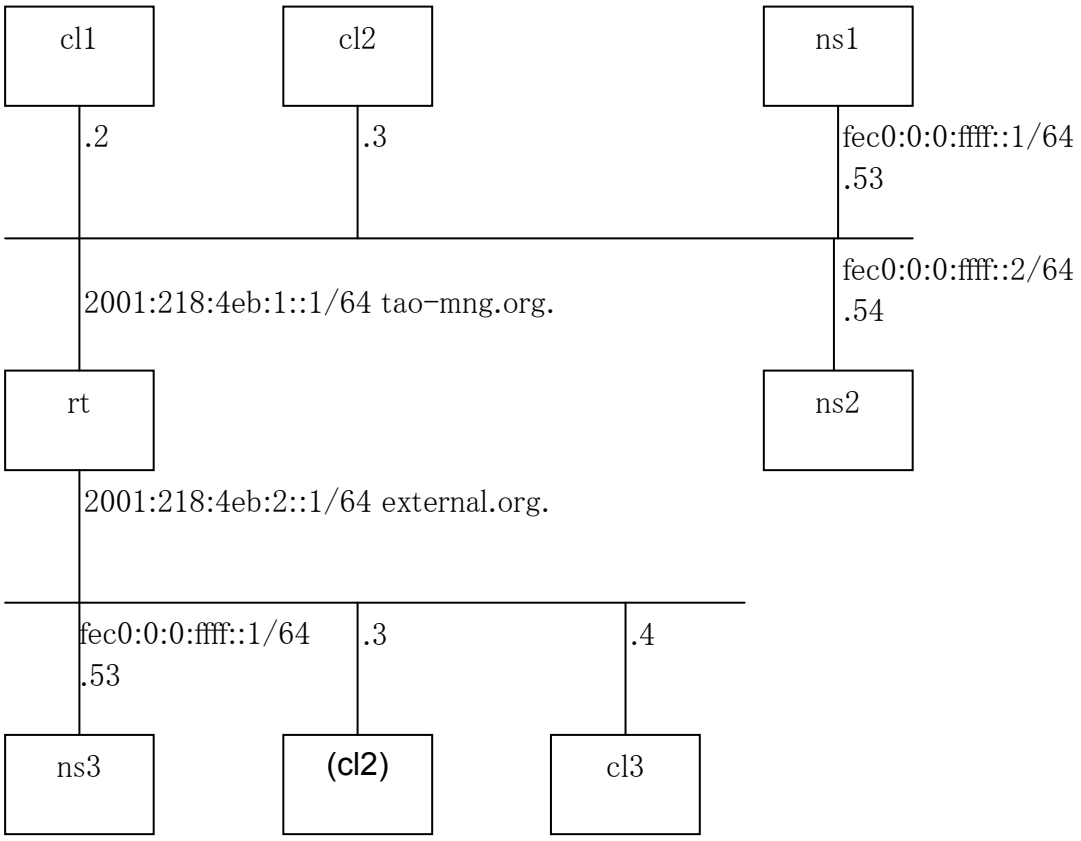


図 19 統合動作検証用ネットワーク環境

(8) 検証作業の具体例

以下、検証結果の具体例を示す。

```
cl1# time ping6 -c 2 cl2.tao-mng.org.
PING6(56=40+8+8 bytes) fe80::210:dcff:fe1f:a0db%rl0 --> fe80::210:dcff:fe2c:dff5
16 bytes from fe80::210:dcff:fe2c:dff5%rl0, icmp_seq=0 hlim=64 time=0.439 ms
16 bytes from fe80::210:dcff:fe2c:dff5%rl0, icmp_seq=1 hlim=64 time=0.282 ms

--- cl2.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.282/0.360/0.439/0.078 ms
0.000u 0.001s 0:01.00 0.0%  0+0k 0+0io 0pf+0w
```

上記は、NI query を利用して cl2.tao-mng.org のリンクローカルアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl2# time ping6 -c 2 cl1.tao-mng.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:1::3 --> 2001:218:4eb:1::2
16 bytes from 2001:218:4eb:1::2, icmp_seq=0 hlim=64 time=0.305 ms
16 bytes from 2001:218:4eb:1::2, icmp_seq=1 hlim=64 time=0.316 ms

--- cl1.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.305/0.310/0.316/0.005 ms
0.000u 0.001s 0:01.01 0.0%  0+0k 0+0io 0pf+0w
```

上記は、tao-mng.org 側に接続した cl2 が近隣の DNS サーバを発見し、cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl2# time ping6 -c 2 cl3.external.org.
ping6: No address associated with hostname
0.000u 0.001s 0:11.18 0.0%  0+0k 0+0io 0pf+0w
```

上記は、tao-mng.org 側に接続した cl2 が発見した近隣の DNS サーバでは、cl3.external.org のドレスが取得できなった場合の実行結果である。

```
cl2# time ping6 -c 2 cl1.tao-mng.org.
ping6: No address associated with hostname
0.000u 0.001s 0:11.90 0.0%          0+0k 0+0io 0pf+0w
```

上記は、external.org 側に接続した cl2 が発見した近隣の DNS サーバでは、cl1.tao-mng.org のドレスが取得できなった場合の実行結果である。

```
cl2# time ping6 -c 2 cl3.external.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:2::3 --> 2001:218:4eb:2::4
16 bytes from 2001:218:4eb:2::4, icmp_seq=0 hlim=64 time=0.399 ms
16 bytes from 2001:218:4eb:2::4, icmp_seq=1 hlim=64 time=0.171 ms

--- cl3.external.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.171/0.285/0.399/0.114 ms
0.000u 0.001s 0:01.01 0.0%          0+0k 0+0io 0pf+0w
```

上記は、external.org 側に接続した cl2 が近隣の DNS サーバを発見し、cl3.external.org のアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl1# time ping6 -c 2 cl1.tao-mng.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:1::2 --> 2001:218:4eb:1::2
16 bytes from 2001:218:4eb:1::2, icmp_seq=0 hlim=64 time=0.223 ms
16 bytes from 2001:218:4eb:1::2, icmp_seq=1 hlim=64 time=0.166 ms

--- cl1.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.166/0.195/0.223/0.029 ms
0.000u 0.001s 0:01.00 0.0%          0+0k 0+0io 0pf+0w
```

上記は ns1 と ns2 が両方とも参照可能な状態において、cl1 から DNS サーバに問い合わせることによって cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl1# time ping6 -c 2 cl1.tao-mng.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:1::2 --> 2001:218:4eb:1::2
16 bytes from 2001:218:4eb:1::2, icmp_seq=0 hlim=64 time=0.366 ms
16 bytes from 2001:218:4eb:1::2, icmp_seq=1 hlim=64 time=0.176 ms

--- cl1.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.176/0.271/0.366/0.095 ms
0.000u 0.001s 0:01.01 0.0%          0+0k 0+0io 0pf+0w
```

上記は ns1 のみが参照可能な状態において、cl1 から DNS サーバに問い合わせることによって cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl1# time ping6 -c 2 cl1.tao-mng.org.
PING6(56=40+8+8 bytes) 2001:218:4eb:1::2 --> 2001:218:4eb:1::2
16 bytes from 2001:218:4eb:1::2, icmp_seq=0 hlim=64 time=0.441 ms
16 bytes from 2001:218:4eb:1::2, icmp_seq=1 hlim=64 time=0.166 ms

--- cl1.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.166/0.303/0.441/0.138 ms
0.000u 0.001s 0:01.01 0.0%          0+0k 0+0io 0pf+0w
```

上記は ns2 のみが参照可能な状態において、cl1 から DNS サーバに問い合わせることによって cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl1# time ping6 -c 2 cl1.tao-mng.org.
PING6(56=40+8+8 bytes) fe80::210:dcff:fe1f:a0db%rl0 -->
fe80::210:dcff:fe1f:a0db
16 bytes from fe80::210:dcff:fe1f:a0db%rl0, icmp_seq=0 hlim=64 time=0.263 ms
16 bytes from fe80::210:dcff:fe1f:a0db%rl0, icmp_seq=1 hlim=64 time=0.169 ms

--- cl1.tao-mng.org ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 0.169/0.216/0.263/0.047 ms
0.000u 0.001s 0:12.37 0.0%          0+0k 0+0io 0pf+0w
```

上記は ns1 と ns2 が両方とも参照不可能な状態において、cl1 から NI query によって cl1.tao-mng.org のリンクローカルアドレスを取得した場合の実行結果である。実行には12秒かかっている。

5-2-4　Linux 版実装の動作検証

(1) 動作検証の内容

Linux 版実装である汎用名前解決エンジンと名前解決モジュールを組み合わせ、一通りの機能が動作することを確認する。また FreeBSD 版実装と挙動の差などがある場合、それが何に由来するものか確認し、マルチプラットフォーム化に際して何が必要であるかを考察する。

(2) 動作検証の準備

Linux 版実装の動作検証は、図 20 に示す動作検証用ネットワーク環境上で行う。ここで、cl2 は tao-mng.org から external.org へ移動するものとする。

動作検証用ネットワークの機器構成を次に示す。

- cl1・cl3
  FreeBSD-4.7R＋KAME-SNAP030127
- cl2
  Redhat Linux 9 Kernel 2.4.20+USAGI20030516
- ns1・ns2
  FreeBSD-4.7R＋KAME-SNAP030127
- ns3
  FreeBSD-4.6.2R
- rt
  CISCO 1605R
- ns1・ns2・ns3
  BIND-9.2.2

ここで、汎用名前解決エンジンと各名前解決モジュールを cl2 にインストールし、次の動作環境を設定した。

- ns1・ns2 のネームサーバ設定
  - cl1.tao-mng.org に対する AAAA レコードとして、2001:218:4eb:1::2 を設定。

- ns3 のネームサーバ設定
  - cl3.external.org に対する AAAA レコードとして、2001:218:4eb:2::4 を設定。
- cl2 上の汎用名前解決エンジンの起動
  - ./mng
  - ./nullmodule -t 2001:218:4eb:1::53
  - ./nullmodule -t 2001:218:4eb:1::54
  - ./nullmodule -t 2001:218:4eb:2::53

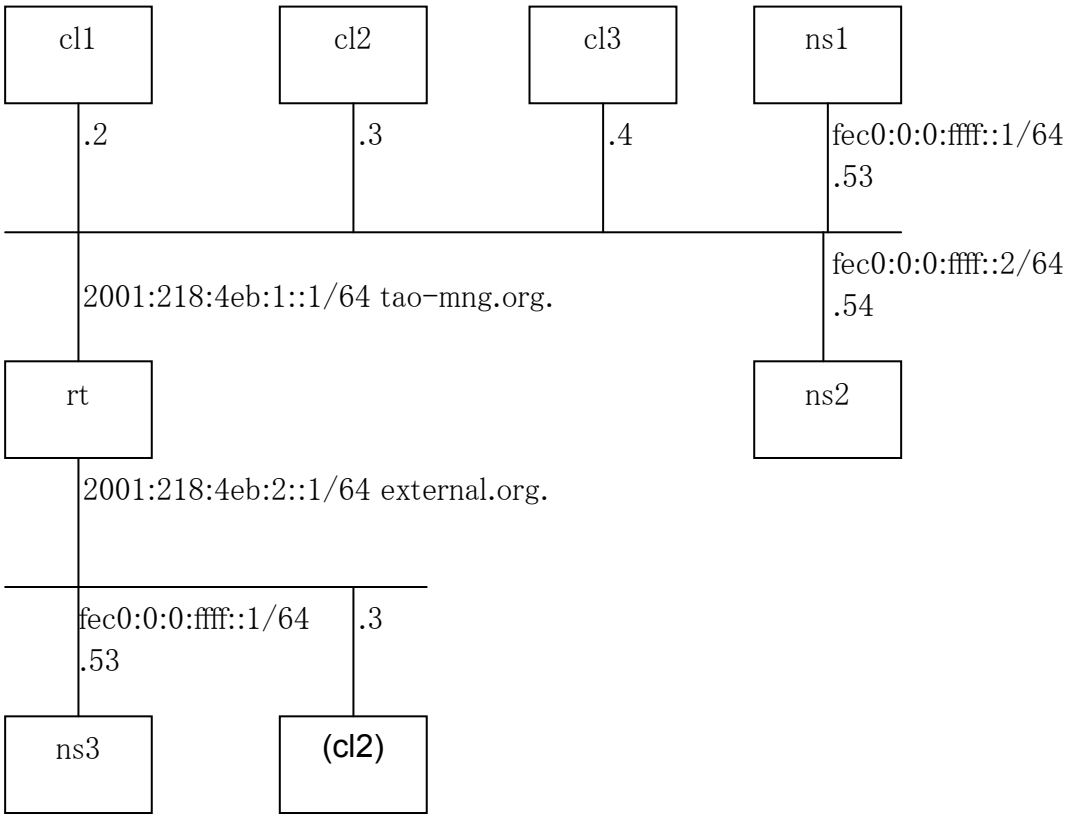


図 20 Linux 版実装動作検証用ネットワーク環境

(3) 動作検証の詳細

動作検証は、以下の手順で行った。

・ 近隣ノードへの名前解決

通常の名前解決が行えることを試験した。

→正常成功。

・ 名前構成木の異なる名前をもつノードへの名前解決

名前構成木の異なる ns3.external.org.の名前を解決する機能を試験した。

→正常成功。

・ プライマリネームサーバを切断したときのフォールバック

ns1 を停止させ、ns2 にフォールバックする機能を試験した。

→汎用名前解決エンジンを使用しない状況ではフォールバックに 6 秒かかっていたが、1 秒に短縮することができた。

・ ネームサーバが知らないノードへの名前解決

ここでは、ns1 を再び正常動作させている。
また cl2 上で、nimodule を起動させた。

■ ./nimodule -i eth0

FreeBSD 版とは異なり、-i オプションにてデフォルトインターフェイスを指定する。

→接続に成功した。
また、汎用名前解決エンジンの使用では A, AAAA レコードが返ってこないことを確認してからクライアントに名前解決の結果を返却するため、今回のように NI Queries 以外にもネームサーバに問い合わせるような状況においては、名前解決まで 22 秒余りかかっている。
FreeBSD 版では解決するまでに 16 秒余りかかっていたので、FreeBSD 版よりは、やや時間がかかっていると言える。

・ well-known site local unicast address による名前解決

機能試験用クライアントには cl2 を用い、アドレスを 2001:218:4eb:1::3 とした。
汎用名前解決エンジンおよび名前解決モジュールをすべて停止させ、次の手順で再起動を行った。

- ./mng
- ./nullmodule -t fec0:0:0:ffff:1
- ./nullmodule -t fec0:0:0:ffff:2

→well known site local unicast address を用いて正常に名前解決ができている。

・ プライマリネームサーバを切断したときのフォールバック試験

fec0:0:0:ffff::1 である ns1 が到達不可能なとき、fec0:0:0:ffff::2 にフォールバックする機能を試験した。

→正常に名前解決できている。

・ cl2 を external.org.側に接続したとき

cl2 に fec0:0:0:ffff::3 を設定した。
fec0:0:0:ffff::1 宛にパケットが到達することを確認する。Linux ではデフォルトインターフェイスを指定することが出来ないため、ping レベルでインターフェイスを指定する。
ただし、今回用いた cl2 はネットワークインターフェイスを一つしか持っていないため、インターフェイスを指定する必要は無かった。また、デフォルトゲートウェイは RA で広告されているものとする。

→fec0:0:0:ffff::/64 経由で名前解決ができ、ping が通ることを確認。

(4) 検証作業の具体例

以下、検証結果の具体例を示す。

```
cl2# time ping6 -nc 2 cl1.tao-mng.org
PING cl1.tao-mng.org(2001:218:4eb:1::2) 56 data bytes
64 bytes from 2001:218:4eb:1::2: icmp_seq=1 ttl=64 time=1.35 ms
```

```
64 bytes from 2001:218:4eb:1::2: icmp_seq=2 ttl=64 time=0.410 ms

--- cl1.tao-mng.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.410/0.883/1.356/0.473 ms

real   0m1.030s
user  0m0.000s
sys   0m0.010s
```

上記は近隣ノードへの名前解決として、NI query を利用して cl1.tao-mng.org のリンクローカルアドレスを取得した場合の実行結果である。実行はすぐに終了している。

```
cl2# time ping6 -nc 2 ns3.external.org
PING ns3.external.org(2001:218:4eb:2::53) 56 data bytes
64 bytes from 2001:218:4eb:2::53: icmp_seq=1 ttl=63 time=4.01 ms
64 bytes from 2001:218:4eb:2::53: icmp_seq=2 ttl=63 time=2.36 ms

--- ns3.external.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 2.369/3.190/4.011/0.821 ms

real   0m1.023s
user  0m0.000s
sys   0m0.000s
```

上記は名前構成木の異なる名前を持つノードへの名前解決として、tao-mng.org 側の cl2 から external.org 側の ns3.external.org のアドレスを取得した場合の実行結果である。実行は1秒ほどで終了している。

```
cl2# time ping6 -nc 2 cl1.tao-mng.org
PING cl1.tao-mng.org(2001:218:4eb:1::2) 56 data bytes
64 bytes from 2001:218:4eb:1::2: icmp_seq=1 ttl=64 time=9.49 ms
64 bytes from 2001:218:4eb:1::2: icmp_seq=2 ttl=64 time=0.404 ms

--- cl1.tao-mng.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1013ms
rtt min/avg/max/mdev = 0.404/4.948/9.493/4.545 ms
```

```
real   0m1.018s
user   0m0.000s
sys    0m0.000s
```

上記はプライマリネームサーバを切断したときのフォールバックとして、ns1 を停止させた場合の状況で cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行は1秒ほどで終了している。

```
cl2# time ping6 -nc 2 -I eth0 cl3.tao-mng.org
PING  cl3.tao-mng.org(fe80::207:95ff:fe15:bc7a)  from  fe80::2e0:18ff:febf:d04f
eth0: 56 data bytes
64 bytes from fe80::207:95ff:fe15:bc7a: icmp_seq=1 ttl=64 time=0.460 ms
64 bytes from fe80::207:95ff:fe15:bc7a: icmp_seq=2 ttl=64 time=0.530 ms

--- cl3.tao-mng.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.460/0.495/0.530/0.035 ms

real   0m22.644s
user   0m0.000s
sys    0m0.000s
```

上記はネームサーバが知らないノードへの名前解決として、DNS に登録されていない cl3.tao-mng.org のアドレスを取得した場合の実行結果である。結果は NI query により取得したリンクローカルアドレスとなっている。また実行には22秒かかっている。

```
cl2# time ping6 -nc 2 cl1.tao-mng.org
PING cl1.tao-mng.org(2001:218:4eb:1::2) 56 data bytes
64 bytes from 2001:218:4eb:1::2: icmp_seq=1 ttl=64 time=2.44 ms
64 bytes from 2001:218:4eb:1::2: icmp_seq=2 ttl=64 time=0.405 ms

--- cl1.tao-mng.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1006ms
rtt min/avg/max/mdev = 0.405/1.423/2.442/1.019 ms

real   0m1.025s
user   0m0.000s
sys    0m0.000s
```

上記は well-known site local unicast address による名前解決として、近隣の DNS サーバから cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行は1秒ほどで終了している。

```
cl2# time ping6 -nc 2 cl1.tao-mng.org
PING cl1.tao-mng.org(2001:218:4eb:1::2) 56 data bytes
64 bytes from 2001:218:4eb:1::2: icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from 2001:218:4eb:1::2: icmp_seq=2 ttl=64 time=0.514 ms

--- cl1.tao-mng.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 0.514/0.877/1.240/0.363 ms

real  0m1.027s
user  0m0.000s
sys   0m0.010s
```

上記はプライマリネームサーバを切断したときのフォールバック試験として、近隣の DNS サーバのうち ns1 を停止させ ns2 にフォールバックする状態で cl1.tao-mng.org のアドレスを取得した場合の実行結果である。実行は1秒ほどで終了している。

```
cl2# time ping6 -nc 2 ns3.external.org
PING ns3.external.org(2001:218:4eb:2::53) 56 data bytes
64 bytes from 2001:218:4eb:2::53: icmp_seq=1 ttl=64 time=1.24 ms
64 bytes from 2001:218:4eb:2::53: icmp_seq=2 ttl=64 time=0.278 ms

--- ns3.external.org ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1013ms
rtt min/avg/max/mdev = 0.278/0.763/1.249/0.486 ms

real  0m1.012s
user  0m0.000s
sys   0m0.000s
```

上記は cl2 を external.org 側に接続したときとして、近隣の DNS サーバである ns3 から ns3.external.org のアドレスを取得した場合の実行結果である。実行は1秒ほどで終了している。

(5) 動作検証のまとめ

今回の Linux 移植について、基本的な性能は移植できた。

Linux 移植版と FreeBSD 版の大きな違いは

- FreeBSD ではデフォルトインターフェイスを指定できるが、Linux では出来ない

ことである。そのため、いくつかの実験項目でインターフェイスを明示的に指定する必要がある。

また、性能について若干の速度の違いが見られる。これも OS のネットワークの作りに依存するものと思われる。

5-3　総括

5-3-1　本研究開発の成果について

本委託業務で行った研究開発は、当初予定されていた平成15年度の研究開発内容をすべて実施することができ、成果物の有効性が確認できたという意味で成功したものと考えられる。

本研究開発は、3 に記した研究開発全体計画の中の平成15年度の研究開発に相当する。本研究開発が成功裏に終了したことにより、研究開発の全体計画がすべて完了したことになる。

5-3-2　本研究開発の成果がもたらす効果について

近年になって名前解決のための新しい技術やプロトコルが提案されているにもかかわらず、あまり実際に使用されていない。それは実装がしにくいことに一つ原因がある。これらの状況を鑑み、本研究開発ではまず様々な名前解決方式を試作、検証できるための汎用名前解決エンジンを開発する。これは容易に従来のアプリケーションと組み合わせて利用可能であり、プラットフォームに依存しない。さらに、この名前解決エンジンの有用性を検証するために上記問題を解決する名前解決方式の実装をエンジンに組み込んで評価、検証する。

本研究開発の成果をベースとして実施される予定の研究開発全体計画の成果物を利用することにより、アプリケーションや OS が提供するリゾルバライブラリにまったく手を加える必要なしに、複数の名前解決機構を利用できる。また、モジュールを追加することによって、新しい名前解決のためのメカニズムが提案された場合においてもモジュールの追加だけで各アプリケーションは新しい名前解決メカニズムを利用でき、効率的に実環境において各名前解決メカニズムの検証を行うことができる。

5-3-3　まとめ

本研究開発では、様々な名前解決メカニズムを統一的に扱う仕組みである汎用名前解決エンジンと、そのエンジン上で利用可能なモバイルサポートのための名前解決モジュールおよびセキュリティやプライバシーを考慮した名前解決モジュールの試作を行い、各名前解決モジュールと汎用名前解決エンジンとを組み合わせた統合動作検証を行った。

本研究開発が成功裏に終了したことで、研究開発全体計画を予定通りに遂行させることができたといえる。

参考資料、参考文献

1

P. Mockapetris,
"DOMAIN NAMES - CONCEPTS AND FACILITIES",
Request for Comments,
rfc1034.txt,
November 1987.

P. Mockapetris,
"DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATION",
Request for Comments,
rfc1035.txt,
November 1987.

S. Thomson, C. Huitema,
"DNS Extensions to support IP version 6",
Request for Comments,
rfc1886.txt,
December 1995.

2

R. Gilligan, S. Thomson, J. Bound, W. Stevens,
"Basic Socket Interface Extensions for IPv6",
Request for Comments,
rfc2553.txt,
March 1999.[13]

3

P. Vixie,
"Extension Mechanisms for DNS (EDNS0)",
Request for Comments,
rfc2671.txt,
August 1999.

4

L. Esibov, et al.,
"Linklocal Multicast Name Resolution (LLMNR)",
Internet draft (work in progress),
draft-ietf-dnsext-mdns-30.txt,

[13] RFC3493 に改定された。

Mar 2004.

5
M. Crawford,
"IPv6 Node Information Queries",
Internet draft (work in progress),
draft-ietf-ipngwg-icmp-name-lookups-10.txt,
June 2003.

6
Alain Durand, Jun-ichiro itojun Hagino, Dave Thaler,
"Well known site local unicast addresses to communicate with recursive DNS servers",
Internet draft (work in progress),
draft-ietf-ipv6-dns-discovery-07.txt,
October 2002.

7
R. Droms,
"DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)",
Request for Comments,
Rfc3646.txt,
December 2003.

8
R. Droms,
"Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6",
Request for Comments,
rfc3736.txt,
April 2004.

9
J. Jeong,
"IPv6 DNS Discovery based on Router Advertisement",
Internet draft (work in progress),
draft-jeong-dnsop-ipv6-dns-discovery-01.txt,
February 2004.

10
D. Eastlake,
"Domain Name System Security Extensions",
Request for Comments,
rfc2535.txt,
March 1999.

11
M. Crawford, C. Huitema,
"DNS Extensions to Support IPv6 Address Aggregation and Renumbering",
Request for Comments,
rfc2874.txt,
July 2000.

12
R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain,
"Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)",
Request for Comments,
rfc3363.txt,
August 2002.

13
R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens,
"Basic Socket Interface Extensions for IPv6",
Request for Comments,
rfc3493.txt,
February 2003.

(添付資料)

1. 研究発表、講演、文献一覧

研究発表は以下の 1 件を実施

情報処理学会　第 3 回分散システム／インターネット運用技術研究会
開催日:2002 年 10 月 18 日(金):岡山大学)
題名:現在の名前解決システムの課題と汎用名前解決エンジンの提案
著者:山田　竜也、 嶋田　雄二郎、島津　伸行、倉富　修、岡　光秋、岡本　利夫、
栄　光宏
予稿:情報処理学会　分散システム／インターネット運用技術研究報告
No.27-5(2002.10.18), pp.25-30(査読なし)