ネームサーバを用いた柔軟な負荷分散

下川 俊彦† 吉田 紀彦‡ 牛島 和夫†
† 九州大学大学院システム情報科学研究科 ‡ 長崎大学工学部

Flexible Load Balancing Mechanism using Name Server.

Toshihiko Shimokawa*, Norihiko Yoshida** and Kazuo Ushijima*
*Graduate School of Information Science and Electrical Engineering, Kyushu University.
**Department of Computer and Information Sciences, Nagasaki University.

概要

インターネット上で、各種サーバの負荷集中への対策として、サーバの複数化が広く利用されている。サーバを複数化した場合、どのようにして各ユーザが複数化したサーバの一つを選択するかが負荷分散の鍵となる。しかし既存のサーバ選択方法はユーザに過度の負担を強いたり、柔軟性に欠けるものである。そこで、本稿ではネームサーバを用いた柔軟なサーバ選択手法を提案する。我々は、ネームサーバに柔軟なサーバ評価機構を組み込むことによりこれを実現する。ネームサーバが名前解決要求に対して、様々な判断基準を用いてサーバ選択を行なうことにより柔軟性と適用性に優れた負荷分散を可能とする。

1 はじめに

近年インターネットでは WWW を始めとする様々なサービスが提供されるようになってきた。これに伴い、ユーザ数も爆発的に増加している。この結果、サーバへの負荷の集中という問題が表面化してきた。

これに対処するため、サーバを複数化することによる負荷の分散が試みられている。サーバの複数化に当ってはサービス自体がサーバ複数化を考慮しているものと、考慮していないものがある。

サービス自体がサーバの複数化を考慮している例として IRC(Internet Relay Chat) や NTP(Network Time Protocol) がある。これらは複数のサーバが基本的に同一の情報を提供できるよう設計されている。

サービス自体はサーバの複数化を考慮していない例としては WWW や FTP がある。これらではミラーサーバを設置し、複数のサーバで同一コンテンツを提供するように運用する。

サーバを複数化した場合、どのようにして各ユーザが複数化したサーバの一つを選択するかが負荷分散の鍵となる。しかし既存のサーバ選択方法はユーザに過度の負担を強いたり、柔軟性に欠けるものである。そこで、我々は柔軟性の高いサーバ選択手法を開発し、効率的な負荷分散を実現することを目指している。

本論文で提案する方式は、ネームサーバに様々なサーバ評価方式を組み込むことで、柔軟なサーバ選択を可能にする。ネームサーバは分散的に動作するために、負荷分散処理も分散的に実行可能となる。さらにネームサーバのプロキシサーバ的に働く質問処理プロセッサを導入することで、既存のネームサーバ網への変更も不要にする。

なお、本論文においてネームサーバとは、現在インターネットでの名前解決に広く利用されている DNS(Domain Name System) を指す。

本論文では 2 章で既存のサーバ選択モデルを概観する。3 章で本研究で提案するサーバ選択方式を説明し、4 章で本方式を評価する。5 章でまとめと今後の課題について述べる。

2 サーバ選択モデル

本章では、まず望ましいサーバ選択モデルについて述べる。次に、既存のサーバ選択モデルを分類し、それぞれを概説する。最後に、現在利用されているモデルについて述べる。

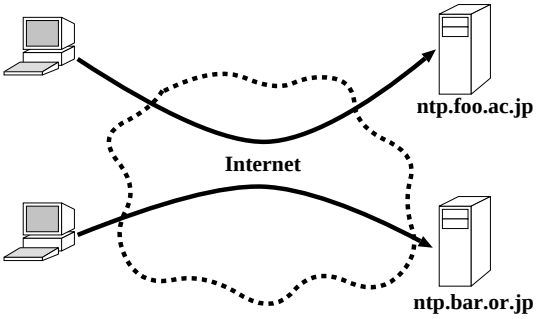

図 1: クライアント側選択モデル

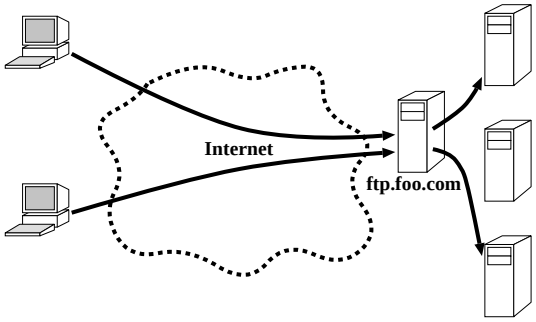

図 2: サーバ選択モデル

2.1 望ましいモデル

サーバの複数化を用いるサービスには様々な種類がある。よって、できるだけサービスに依存しない広い適用性を持つものが望ましい。また、サーバ群の一部だけでなく、サービス全体としての負荷分散に対応できるほうが良い。

サーバ選択システムが新たなボトルネックになっては、サーバ複数化が無意味となる。よって、ボトルネックが存在せず、規模適応性があるものが良い。

サーバ選択に当っては、複数化されたサーバを評価するかしないのか、評価しない場合どのようにして選択するのか、評価する場合その評価基準は何か、と言った点を柔軟に変更できるべきである。

2.2 既存のモデルの分類

既存のモデルを、サーバ・クライアント間のどこでサーバを選択するかを基準に以下の 3 つに分類した。

1. クライアント側選択モデル
2. サーバ側選択モデル
3. 中間システム選択モデル

2.3 クライアント側選択モデル

クライアント、またはクライアントを起動するユーザが判断し、複数化したサーバから任意の一台を選択するモデルである (図 1)。

このモデルは、クライアント側が負荷分散に必要な知識を持っていることを仮定する。一般的にはこの仮定は期待できない。

クライアント側に必要な知識を持たせるものとして、Smart Clients[1] がある。この方法ではクライアントとして Java applet を用い、サーバ側の状態を通知することでクライアント側での負荷分散を実現している。この方法はクライアント側に Java の機能が必要であり、適用性に制限がある。また、既存のクライアントを利用することができないという欠点もある。

ユーザが判断する場合、ユーザは本来不必要な判断を強いられることになる。この方法は実現が容易であるが、採用するべきではないモデルと考える。

2.4 サーバ側選択モデル

要求を受け付けたサーバ (以下、このサーバを主たるサーバと呼ぶ) で判断するモデルである (図 2)。状況に応じて、主たるサーバが、複数化された他のサーバに要求を転送する。

このモデルは、主たるサーバの判断で様々な状況に対応することが可能となり、柔軟性は高い。ただし、主たるサーバがボトルネックとなる可能性が高く、規模適応性に欠ける。

SWEB [2] では、主たるサーバを複数置くことで、ボトルネックとなる可能性の軽減を計っている。しかし、複数化されたサーバ間で負荷状態情報の交換のオーバヘッドか生じる。また、要求を転送されたサーバとクライアント間で新たにコネクションを確立する必要もあり、このオーバーヘッドも大きい。

2.5 中間システム選択モデル

クライアントとサーバの中間にあるシステムで選択するモデルである。ネットワーク中には中間システムとして多様なものが存在する。ネットワークの階層に沿って、さらに以下のモデルに分類できる。

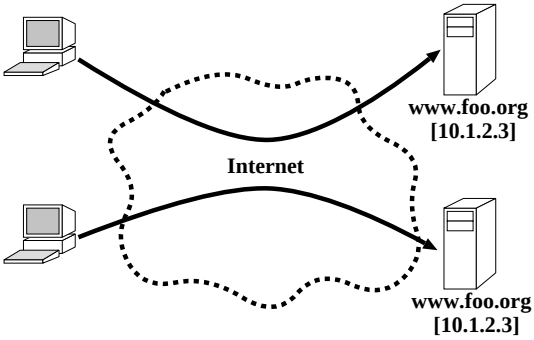

図 3: ネットワーク選択モデル

2.5.1 ネットワーク選択モデル

ネットワーク自体が選択するモデルである (図 3)。

[3] では、複数化したサーバ群に同一 IP アドレスを割り当てておき、選択をネットワークの経路制御にまかせた例が報告されている。IP Version 6 で導入された anycast も、このモデルの実装例の一つと考えることができる。

非常に特殊な例を除くと、全てのアプリケーションはネットワークの経路制御の影響を受ける。つまり、このモデルは最も適用性が広いという利点を持つ。しかし、このモデルを実現するための技術が未成熟という欠点がある。また、サーバ評価基準の柔軟性に欠ける。

2.5.2 アプリケーションゲートウェイ選択モデル

サーバとクライアントが直接通信するのではなく、中間的なシステムを介して通信する場合が増えてきている。例えば、ファイアウォールに伴うプロキシサーバや通信量の削減を目的としたキャッシュサーバを用いる場合である。以下、これらの中間的なシステムのことを、アプリケーションゲートウェイと呼ぶ。

このアプリケーションゲートウェイでサーバを選択するモデルである (図 4)。このモデルでは、アプリケーションゲートウェイが持つサーバ評価基準により柔軟性が決まる。多様なサーバ選択基準を持ったアプリケーションゲートウェイを用いた場合、高い柔軟性を持つ。このモデルは、各サービスをアプリケーションゲートウェイに対応させる必要があり、適用性に問題がある。

2.5.3 メタサーバ選択モデル

サーバへアクセスする前に利用されるメタなサービスが存在する。例えば、ホスト名の名前解決を行う DNS や、サービスの発見を行う Service Location Protocol[4] である。また URL Resolver[5] のように、負荷分散を目的としたメタサービスもある。

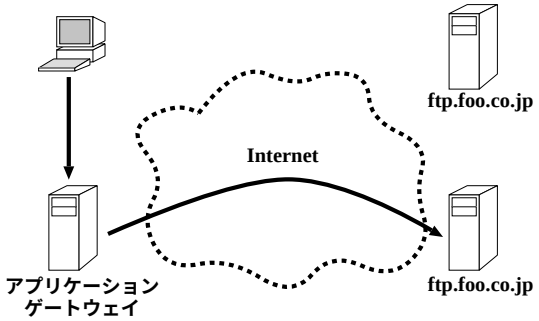

図 4: アプリケーションゲートウェイ選択モデル

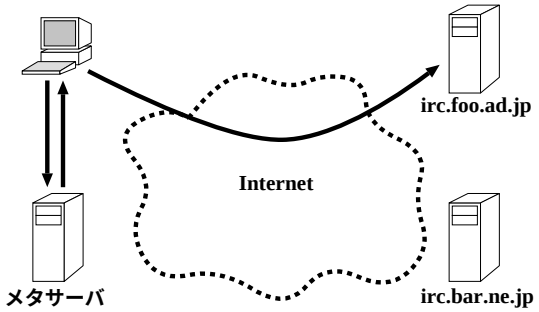

図 5: メタサーバ選択モデル

これらのメタサービスのサーバでサーバを選択するモデルである (図 5)。このモデルの適用性および柔軟性は利用するメタサービスに左右される。

メタサービスとして DNS のように、ほぼ全てのアプリケーションが利用するものを用いた場合、広い適用性が得られる。一方 SLP のように必ずしも普及が進んでいないものを用いると、適用性に制限が出る。

2.6 現在利用されているモデル

現在広く用いられているモデルは、クライアント側選択モデルとメタサービスとして DNS を用いたメタサーバ選択モデルである。

以下のような方法で、クライアント側選択モデルを実現している。サーバを複数化しそれらに異なるホスト名を付ける。その全てをアナウンスし、ユーザは各自の判断でこれらの中から任意の一台を選択し利用する。

この方法は実現が容易であるという利点がある。しかし、2.3 節でも述べたように、この方法はユーザに本来不要なはずの判断を強いており、望ましくないと考えている。

メタサービスとして DNS を用いたメタサーバ選択

モデルは、適用範囲が広いという特徴を持つ。インターネット上のアプリケーションのほとんどは、他のホストと通信する前に、ネームサーバ (DNS) を用いてホスト名を IP アドレスへ変換 (名前解決) する。さらに、ネームサーバは分散的に動作するために、ボトルネックになりにくいという利点もある。

3 ネームサーバを用いた柔軟なサーバ選択

前章で述べたように、ネームサーバを用いたサーバ選択は利点が多い。しかし、現在広く利用されている実装である BIND(Berkeley Internet Name Domain) のサーバ選択には限界がある。ます複数化されたサーバの評価を行わない。そして選択方法は単純なラウンドロビン選択である。つまり、サーバ選択に柔軟性が無い。

この改善としては重み付けラウンドロビン方式 [6] がある。この方法では、サーバ毎に処理能力に応じた数の IP アドレスを割り当てている。こうすることで、ラウンドロビンでも擬似的にサーバの処理能力を評価できるようにしている。また、既存の BIND への変更も不要である。しかし、この方法ではクライアント・サーバ間のネットワークの状況などに対応することはできない。

Cluster DNS[7] では、新たな負荷分散機構をネームサーバに組み込んでいる。ただし、本手法では Cluster DNS をサーバ提供者が動かすことを仮定している。つまり、サービス提供者が Cluster DNS を提供しない場合、ユーザは複数化されたサーバを使い分けるのは困難である。

現在のインターネットでは、多くのサービス提供者が複数化したサーバを用意している。しかし、残念ながらそれらを適切に振り分ける機能を提供していない場合も多い。そこで、本研究では、サービス提供者が振り分け機能を提供しない複数化されたサーバ群についても、最適なサーバの選択を可能とするシステムの実現を目指す。

本研究でも Cluster DNS と類似のアプローチを用いる。筆者らはネームサーバに様々なサーバ評価方式を組み込めるようにすることで、柔軟なサーバ選択を実現する手法を提案している [8][9]。本章でこの手法について説明する。

3.1 様々な評価基準

サーバを評価する基準としては、様々なものがある。例えば、クライアント・サーバ間のネットワークの状態や、サーバの負荷状態である。本論文では、特にクライアント・サーバ間のネットワークの状態に着目してサーバ評価を行う方法について述べる。

評価を行うために、クライアント・サーバ間のネットワークを監視する。以下、監視を行う主体を監視主体と呼ぶ。監視主体はネームサーバから分離していても良い。

3.1.1 能動的状態監視

監視主体から能動的にネットワークの状態を監視する。具体的には以下のような方法がある。

- プローブによる監視
- サービスを利用した監視

プローブ (例えば ICMP ECHO REQUEST パケット) を送信し、複数化されたサーバ群とクライアントの距離 (例えばラウンドトリップタイム) を計測する。この結果を評価に用いる。

あらかじめ、利用しようとしているサービスの種類が分かっている場合、プローブとして、そのサービス自体を用いることもできる。ここではこの方法をサービスを利用した監視と呼ぶ。例えば WWW サーバに対して、なんらかのコンテンツの転送を要求しその時間を計測する方法である。

しかしネームサーバが、名前解決要求を出したクライアントが利用しようとしているサービスの種類をあらかじめ知る一般的な方法は存在しない。そこで、RFC2219[10] で提案されているサービス名に対応したホスト名の別名定義を仮定して推測することを考えている。つまり、‘www’ と言う文字列で始まるホストへのアクセスは WWW サービスへのアクセス、‘ftp’ という文字列で始まるホストへのアクセスは FTP サービスへのアクセスと推測する。この点に関しては 4.3.1 節で考察する。

能動的状態監視は、ネットワークに無駄なトラフィックを流すという欠点がある。特に、実際のサービス要求を行う方法は、サーバに対する負荷も含め、影響は小さくない。一方 ICMP ECHO REQUEST パケットをプローブとして用いる方法は負荷の問題は比較的小さい。しかし、昨今は ICMP ECHO REQUEST のように、本来のサービスに不要なパケットはフィルタリングされてしまい、サーバまで届かないことも多い。

この場合、正しく状態を監視できない。この点、実サービスを用いた場合、フィルタリングされる可能性は非常に小さい[1]ので問題がない。

3.1.2 受動的状態監視

受動的に監視するために、ネットワークへのオーバーヘッドがほとんど無いと言う利点を持つ。ただし、能動的監視と比べると、必要な情報を得にくいという欠点がある。

受動的状態監視の対象としては以下のようなものがある。

- サービスパケット
- 経路情報

ここで、サービスパケットとは、クライアントがサーバと通信するパケットのことである。これは、実際のサービス利用の状態を監視することができるので、高い精度が期待できる。しかし、監視主体がネットワーク上を流れるサービスパケットを監視するのは容易ではないという問題点がある。

動的経路制御に用いる経路情報には、経路を評価するための情報が含まれる。この情報を流用してクライアント・サーバ間のネットワークを評価する。こうすることで、ネットワーク的な距離がもっとも近い可能性が高い、もしくは、ネットワークの構成的にそのサーバの方が他のサーバよりも望ましい可能性が高いサーバを選択できる。ただし、経路情報からは、途中のネットワークの状況を知ることはできない。また、経路制御情報を得られない場合にはこの方法を用いることはできない。

3.2 選択候補

本方式は、サービス提供者がサーバ振り分け機能を提供しない場合への適用性を目標の一つとしている。クライアント側でサーバ振り分けを行う場合、その選択の候補とするサーバ群、より正確にはその IP アドレス群をどうやって求めるか、という問題がある。

サーバを複数化する場合、そのサーバ群のホスト名と IP アドレスをどうアナウンスするかで、以下の 4 つに分類できる。

1. 単一ホスト名・単一 IP アドレス
2. 単一ホスト名・複数 IP アドレス
3. 複数ホスト名・単一 IP アドレス
4. 複数ホスト名・複数 IP アドレス

1. はサーバ側以外には単一 IP アドレスしか見えない。よって、サーバ側選択モデル以外では選択の余地は無い。

2. は、ネームサーバに問い合わせを出すことで、全ての IP アドレス群を得られることが多い。このアドレス群を選択候補として利用する。ただし、ネームサーバが複数の IP アドレスを一つづつしか返答しない場合、自動的に全ての IP アドレス群を得ることは困難である。

3. は、バーチャルサーバサービス等で利用されることはあるが、負荷分散を目的としたサーバの複数化に使われることは少ない。使われたとしても、1. 同様サーバ側選択モデル以外では選択の余地は無い。次に述べる 4. と同等の方法で選択することも可能である。しかし、実際に選択されるホストは単一であるので、選択の効果は無い。

4. は、クライアント側選択モデルで利用されることが多い。この場合でも 2. 同様これらの IP アドレス群を選択候補とする。ただし、2. と違い、 IP アドレス群を一括して取得する方法がない。よって、自動的にこれら全てを選択候補とすることは不可能である。しかし人手を介することによって、全てを選択候補とすることを可能とする。この点については次節で詳細を述べる。

3.3 複数 ホスト名・複数 IP アドレスへの対応

前節で述べたように同一ホスト名に対して複数の IP アドレスをネームサーバデータベースに登録してあれば、自動的にそれらを選択候補とする。しかし、2.6 節で述べたように、現在のインターネットでは、単一のサービスに対して複数のホスト名を用意し、クライアント側選択モデルによる負荷分散を利用している場合が少くない。そこで、人手を介してこれらを選択候補とする方法を述べる。

まず、同一サービスを提供する複数ホストのリストを作成する。前述のように現在はこれを自動化する方法はない。よって、手動で作成する必要がある。このリスト上のホスト群がもつ全ての IP アドレス群を選択候補とする。

上述のリスト上の全てのホストに対する問い合わせに対する選択候補として、ここで得られた IP アドレス群を利用する。つまり、あるホスト名に関する問い

[1]特定ホストからのリクエストしか受け付けない場合に、監視主体からのパケットがフィルタリングされる可能性がある。

合わせに対して、実際にネームサーバには登録されていない IP アドレスを返す可能性がある。

具体的な例を示す。irc.foo.ad.jp. と irc.bar.ne.jp. と言う 2 台の IRC サーバがあり、それぞれ 133.5.0.1 と 192.50.13.250 と言うアドレスを持つとする。また、この 2 台は相互に接続され、共通の IRC 空間をサービスしているとする。この場合に、irc.foo.ad.jp. および irc.bar.ne.jp. のいずれかの IP アドレスへの問い合わせに対して、133.5.0.1 と 192.50.13.250 の両方を選択候補として、サーバ選択を行い、応答を返す。つまり、irc.foo.ad.jp. と言うホストの名前解決要求に対して、本来 irc.foo.ad.jp. のアドレスではない、192.50.13.250 と言うアドレスを返すことがある。

以上のように人手を介することで、複数ホスト・複数 IP アドレスを用いたサーバ複数化に対応する。しかし、人手を介する必要があるため、この方法は適用範囲に限界がある。

3.4 質問処理プロセッサの分離

既存のネームサーバ網への影響を最小限に抑えるために、プロキシサーバ的に動作する質問処理プロセッサを導入する。既存のネームサーバは、ドメインネーム空間を管理するデータベース部分と、クライアントからの要求に答える質問処理プロセッサが一体となっている (図 6)。クライアントからのネームサーバへの問い合わせに従ってネームサーバデータベースを検索し、その結果を返す。

この両者を分離する (図 7)。質問処理プロセッサは、クライアントからのネームサーバへの問い合わせには答えるものの、単にネームサーバデータベースを検索するだけではない。問い合わせによっては、様々な選択基準を用いて選択候補の中から最も適切な候補を選択する。ネームサーバデータベースの検索先として従来のネームサーバを利用する。このため、既存のネームサーバには手を加える必要がない。

クライアントは利用するネームサーバを変更するだけで良い。DHCP や PPP を用いている場合、ネームサーバは自動で設定されることが多い。このため、利用者はネームサーバの変更を全く意識しないで良い場合が多いと考える。

4 評価と考察

本章ではプロトタイプシステムを用いて、本研究で提案するサーバ選択手法を評価し考察を加える。本論文では、評価として我々の提案方式の有効性を示すための予備調査の結果を示す。

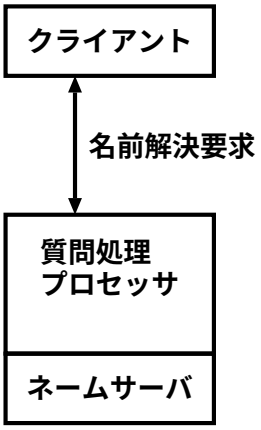

図 6: 従来のモデル

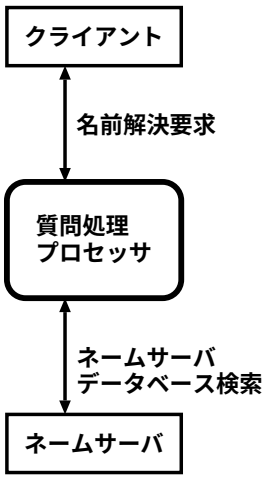

図 7: 本研究によるモデル

4.1 プロトタイプシステムの実装

現在、プロトタイプシステム “tenbin” [2]を実装している。

tenbin は、名前解決要求受け付け部、評価基準決定部、評価基準データベース、評価実行部からなる。

クライアントが発行した名前解決要求は、名前解決要求受け付け部が解釈をおこなう。tenbin が対応していない要求形式の場合、あらかじめ決めておいた既存のネームサーバに要求を転送する。

それ以外の要求は、評価基準決定部において評価基準を決定する。解決要求されたホスト名をキーにして、評価基準データベースを検索する。評価基準データベースにホスト名を与えると、利用する評価基準が得られる。現在は、サーバ選択機能を利用する全対象ホストを、あらかじめ評価基準データベースに登録しておく必要がある。

各評価基準は、それぞれ評価実行部を持ち、ここが評価及び選択処理を行う。各評価実行部は必要に応じて監視主体と通信し判断に必要な情報を得る。決定した評価基準に対応する評価実行部に、名前解決要求が渡され、名前解決が実行される。

対応する評価基準が存在しない場合には、デフォルトの評価基準を用いる。現在のデフォルトの評価基準は、評価を行わなす、評価実後部が受け取った解決要求

[2]Tenbin is Experimental Nameserver for load Balanced INternet

表 1: ログの解析結果

総問い合わせ数	266420
総クライアント数	48
アドレス問い合わせ数	233659
応答種類数	10503
有効返答アドレス種類数	9570
複数アドレスホスト数	257

をそのままあらかじめ決めておいた既存のネームサーバに転送する。

以上の流れで各要求が解決され、結果がクライアントに返される。

今回は評価基準としては、プローブパケットを用いた能動的状態監視を行うものだけを実装した。また、監視主体は評価実行部に組み込んだ。

tenbin の記述言語としてはオブジェクト指向スクリプト言語 Ruby[11] を用いている。これは、スクリプト言語を用いることによる開発効率の高さ、オブジェクト指向言語であることによる記述の容易さを考慮したからである。

4.2 プロトタイプシステムによる評価

tenbin を筆者らの研究室のネットワークで動かし、一週間分のログを解析した。

4.2.1 選択候補にするホスト

我々の提案方式では、複数化されたサーバ群が、単一ホスト名・複数 IP アドレスまたは複数ホスト名・複数 IP アドレスで運用されていることを期待している。複数ホスト名・複数 IP アドレスについては自動的に調べる方法はないので、単一ホスト名・複数 IP アドレスで運用されているホストがどれくらいあるかを調査した。

tenbin のログを解析した結果表 1 に示すような内容が明らかになった。

tenbin は 10503 種類のホスト名に関する名前解決要求を受け付けている。このうち 257 のホスト名は、ネームサーバに複数アドレスが登録されていた。つまり、単一ホスト名・複数 IP アドレスで運用されていることが分かる。この 10503 のホストが、それぞれいくつの IP アドレスを持っていたかを図 8 に示す。

同一ホスト名に、複数の IP アドレスが登録されているホストは、複数のホストである可能性がある。しかし、ルータのように単体で複数のネットワークに繋

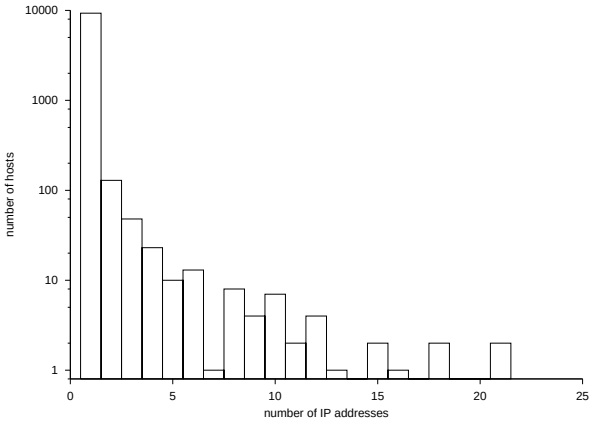

図 8: 各ホストが持つ IP アドレス数

がっているために、複数の IP アドレスを持っている場合もある。ファイアウォールを構成するルータのように組織の境界に存在する計算機で各種サービスを提供していることは充分考えられる。しかし、最近のネットワークの構成では、そのような計算機が多数 (3〜4 以上) のネットワークに繋がっている例は少いと予想する。よって、実際に複数の計算機を使ってサービスを提供している例は比較的多いと考える。

しかし、これらのホストは同一ネットワーク上に複数設置されている場合も多い。このような場合、その複数 IP アドレス間で選択を行っても、ネットワーク状況に対応するという点からは、効果が薄い。そこで、これらネームサーバに複数の IP アドレスが登録されているホストがいくつのネットワークに繋がっているかを推定した。

IP を用いたネットワークでは、サブネットマスクが分からないとネットワークが異なるかどうかを判断することはできない。そこで、今回は IP アドレスの先頭 1 オクテットを見てアドレスのクラス A, B, C を判定し、そのナチュラルマスクを仮定した。一般的にはこの仮定が成立しないのは言うまでもないが、今回の評価に当ってはほぼ問題がない。この結果を 9 に示す。

このことから、単一ホスト・複数 IP アドレスで運用されているサーバについては、その全ての IP アドレスを候補として選択するのではなく、適当なネットマスクを仮定し、その結果異なるネットワークに繋がっていると判断されたものについてを選択候補にするのが有効である。

以上のようにクライアント側の判断で、複数化されたサーバから選択候補を求めることは実現可能であると考える。

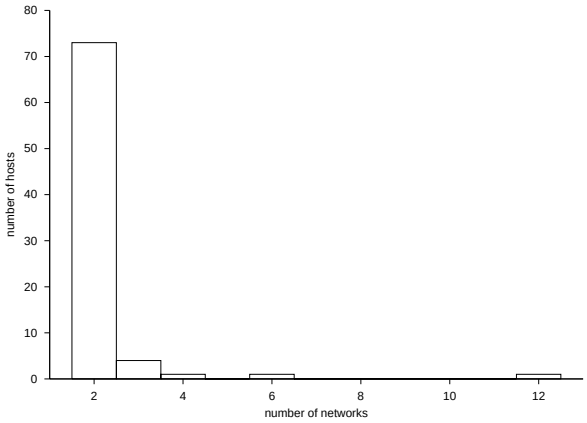

図 9: 各ホストが繋がっているネットワーク数 (推定)

4.2.2 能動的状態監視

能動的状態監視を行う場合、その対象となるサーバ群の数が増えると、ネットワークに与える影響も大きくなる。そこで、複数のネットワークに設置されたサーバ群のみを監視対象とすることを考える。図 9 から判断すると、監視対象となるサーバ群の数はそれほど多くはならない。よって、影響は無視できると考える。

次にプローブパケットによるサーバ選択結果の変化を調査した。ここでは、RingServer プロジェクト [12] が運用する FTP サーバ ftp.ring.gr.jp. を対象とした。ftp.ring.gr.jp. は 12 個のサーバが 12 個のネットワークに繋がっている[3]。約 30 分毎にラウンドトリップタイムを計測した。サーバ毎の選択された回数を表 2 に示す。また、選択された上位 2 つのサーバについてのラウンドトリップタイムの変化を図 10 に示す。

表およびグラフから分かるように、選択結果の変化は頻繁ではない。よって、状態監視の間隔はあまり短くしても効果は少ないことが分かる。

4.2.3 オーバヘッド

クライアントとネームサーバの間に tenbin が入ることによるオーバヘッドを計測した。

tenbin が受け付けた要求のうち、既存のネームサーバに要求を転送した処理について tenbin の実行処理時間を調査した。その結果、tenbin が入ることによるオーバヘッドは平均 9.1 ミリ秒ということが分かった。このオーバヘッドは無視しても問題ないと考える。

[3]RingServer のサーバは 13 個である (1999/8/31 現在)。しかし、ftp.ring.gr.jp. というホスト名に登録されているアドレスは 12 個であるため、12 個のアドレスについて計測した。

表 2: ホスト毎の最善と判断された回数

ホスト名	回数	割合 (%)
ring.nacsis.ac.jp.	180	61.6
ring.ocn.ad.jp.	45	15.4
ring.asahi-net.or.jp.	19	6.5
ring.etl.go.jp.	14	4.8
ring.so-net.ne.jp.	13	4.5
ring.aist.go.jp.	9	3.1
ring.crl.go.jp.	7	2.4
ring.shibaura-it.ac.jp.	3	1.0
ring.jah.ne.jp.	2	0.7

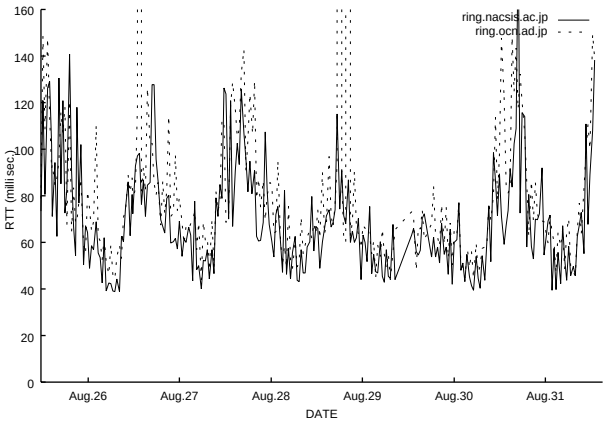

図 10: ラウンドトリップタイムの時間変化

4.2.4 質問処理プロセッサ分離の効果

tenbin を動かすに当って、既存のネームサーバ群には一切変更を加える必要が無かった。

また tenbin を利用するに当っては、DHCP および PPP クライアントに関しては一切変更が不要であった。DHCP サーバおよび PPP サーバがクライアントに通知するネームサーバの IP アドレスを tenbin を動かしているホストのものに変更するだけで済んだ。DHCP や PPP を利用していないホストに関しては、そのホストが利用するネームサーバの設定を手動で変更する必要があった。ただし、各ホスト毎に一度だけ、ホスト管理者が行うのみであった。よって、ユーザはこの変更を意識する必要はなかった。

以上のように、質問処理プロセッサを分離したことにより、以下の 2 つの特徴を実現できた。

- 既存のネームサーバ群には変更を加えない
- ユーザは tenbin の存在を意識しない

4.3 ネームサーバを用いることによる限界

ネームサーバを用いた負荷分散の限界について考察する。

4.3.1 サービスの判定方法

3.1.1 でも述べたように、名前解決要求の段階では、その要求がどういうサービスへのアクセスなのかを知ることができない。これを知るためには TCP/UDP が用いるポート番号の情報が必要になるが、名前解決要求にはその情報は含まれない。

例えば、www.foo.com というサーバで、WWW サービスと FTP サービスが提供されている場合を考える。このような状況で、WWW サービスについてのみ www.jp.foo.com という複数化されたサーバが提供されているとする。この場合、ネームサーバはクライアントが利用しようとしているサービスが判断不可能なので、どのようなサーバ評価基準を用いても、www.foo.com の代りに www.jp.foo.com を選択することは不可能である。

この問題への解決策として RFC2219[10] で提案されている、サービス名に対応したホスト名の別名定義を仮定することを考えている。この仮定を評価するために、我々の専攻で動かしている squid web キャッシュサーバのアクセスログを解析してみた。

総アクセス数 1702134 から得られた、http によるユニークなアクセス先ホストは 30068 であった。このうち、‘www’ と言う文字列から始まるホストは 18479 であった。これは 61% に相当する。

この結果からは ‘www’ で始まるホスト名を持つが WWW サービスを提供しない場合、および、‘www’ でサービスで始まるホストの WWW 以外のサービスを利用する場合が明らかではない。しかし、RFC2219 を仮定する方法は実用範囲だと考える。

筆者らは、今後のサービス提供においては RFC2219 に沿ってホスト名を付けることを提案する。

4.3.2 期待と異なるホストへのアクセス

複数化されたサーバのうちの特定の一台を利用したい場合もある。このような場合に、本方式のネームサーバを用いていると、他のサーバが選択されてしまい、その一台を利用することができない可能性がある。この点については、解決方法が二つある。

一つ目は、ホスト名を用いてアクセスせずに IP アドレスを用いてアクセスする方法である。IP アドレスを直接取扱うのは、一般ユーザにとっては容易ではない。しかし、そもそも特定の一台を利用したい場合というのは、非常に特殊な場合と考えている。例えば、システム管理者による保守作業などの管理業務の場合が多いと考える。よって、IP アドレスを用いる方法で問題無いと考える。

二つ目は、複数化されたサーバにそれぞれユニークな別名を付けておくことである。こうすると、IP アドレスを用いずに特定の一台にアクセスすることが可能となる。しかしこの場合、一般ユーザがこのユニークなホスト名を用いてサービスを利用すると、サーバ選択が機能しないという問題点がある。

4.3.3 ホスト名のみによる選択

サービスを利用するためには、ホスト名以外の情報も必要である。TCP/UDP のポート番号や、サービスによってはコンテンツのパス名が必要な場合もある。複数化されたサーバ群でこれらが異なっている場合、我々の提案方式では対応することができない。

例えば、最近 IRC サーバには接続クライアント数の増加に対応するために、一台で複数のポートでサービスを提供するものがある。しかし、全てのサーバがそうなっているわけではないため、これらのサーバ群をまとめて選択候補とすることはできない。FTP や WWW で、複数化されたサーバ間でパス名が異なる場合にも同様に

提供するポートや、パス名が共通であるサーバ間で

は本方式が適用可能である。

5 まとめと今後の課題

本論文では複数化したサーバの選択方法をモデル化し分類を行った。また、ネームサーバに柔軟なサーバ評価基準を組み込むことにより、広い適用性を持ち、広域分散的に動作するサーバ選択システムを提案した。本システムは、多様なサーバ評価基準を持ち、様々な状況に対応することができる。本方式をプロトタイプを用いて評価し、その有効性を示すための予備調査を行った。

今後の課題としては、以下の点がある。

1. 広域に分散したサービスを用いての評価
2. 様々な評価基準の設計・実装
3. 評価基準のより良い決定方法の検討
4. サーバ側とクライアント側に配置した場合の検討

今回の予備調査の結果を元に、広域に分散配置されたサーバを対象として、本提案方式と既存の方式について比較する。

現在の tenbin は、プローブパケットを用いた能動的状態監視しか実装していない。サービスを利用した能動的状態監視や受動的状態監視についても実装し評価する必要がある。

本論文では我々のシステムをクライアント側に配置することに主眼を置いて議論を行った。しかし、我々のシステムはサーバ側でのサーバ振り分けにも適用可能である。その場合、サーバ側に置いたシステムとクライアント側に置いたシステムでの連携が必要となる。そのための方式についても検討が必要である。

6 謝辞

本研究を進めるにあたってデータ収集に協力頂いている、東京大学の江崎浩氏ならびにインテック・システム研究所の中川郁夫氏に感謝致します。

参考文献

[1] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, D. Culler. *Using Smart Clients to Build Scalable Services.*, Usenix '97.

[2] D. Andresen, T. Yang, V. Holmedahl and O. Ibarra., *SWEB: Towards a Scalable WWW Server on MultiComputers*, Proccedings of the 10th International Parallel Processing Symposium, April, 1996.

[3] 重近、中村、笹川、村井: "長野オリンピックのネットワークと情報提供システム," 情報処理 Vol.39 No.2, 1998

[4] E. Guttman, *Service Location Protocol: Automatic Discovery of IP Network Services*, IEEE Internet Computing, Vol. 3, no. 4, July-August, pp. 71–80

[5] 栗原, 廣津, 高田, 菅原: "URL Resolver における柔軟な経路選択メカニズム," 第 7 回マルチ・エージェントと協調計算ワークショップ, http://www.kdel.info.eng.osaka-cu.ac.jp/%7Ekitamura/macc98/proc/kurihara.html, December 1998

[6] 馬場、山口: "DNS を用いた広域負荷分散の実装," 情報処理学会研究報告 98-DSM-9, pp 37–42. May 1998

[7] V. Cardellini, M. Colajanni, P.S. Yu, *DNS dispatching algorithms with state estimators for scalable Web-server clusters*, World Wide Web Journal, Baltzer Science, vol. 2, no. 3, July 1999, pp. 101-113.

[8] 下川、吉田、牛島: "ネームサーバを用いた負荷分散方式," 情報処理学会第 57 回全国大会講演論文集 pp. 3-477 –3-478, October 1998

[9] 下川、吉田、牛島: "ネームサーバを用いた自律分散的負荷分散," 信学技法, CPSY98-160, January 1999

[10] M. Hamilton, R. Wright, *Use of DNS Aliases for Network Services*, RFC 2219, October 1997

[11] Y. Matsumoto, *Ruby the Object-Oriented Script Languate*, http://www.netlab.co.jp/ruby/

[12] Ring Server プロジェクト, http://www.ring.gr.jp/