DNS フィルタ方式によるミラーサーバ選択法の提案と実装

横田 裕思 † 木村 成伴 ‡ 海老原 義彦 ‡
† 筑波大学大学院 工学研究科
‡ 筑波大学 電子 · 情報工学系

A Proposed of DNS Filter Methods to Select Suitable Mirror Servers and Its Implementation

YOKOTA Hiroshi† KIMURA Shigetomo‡ EBIHARA Yoshihiko ‡
† Doctoral Program in Engineering, University of Tsukuba
‡ Institute of Information Sciences and Electronics, University of Tsukuba

概要

サーバに対する過度の負荷集中への対処として，ミラーサーバを用いたサーバの複数化による負荷分散が広く行われている．このシステムでは，どのサーバを選択するかにより，通信効率が大きく左右される．一般に，ユーザが適切なサーバを選択することは困難であるため，DNS ラウンドロビンなどを用い，サーバ周辺に負荷を分散させる方式がとられていた．しかし，これはあくまでもサーバ側から見た負荷分散であり，ここで選択されたサーバは，サーバがネットワーク的に分散して配置している場合は特に，クライアントにとって真に最適な選択が行なわれるとは限らない．そこで本論文では，ローカル DNS サーバにフィルタ機能を持たせ，ユーザにとって適切なサーバを自動的に選択する DNS フィルタ方式を提案し，これを実装する．また，DNS ラウンドロビン方式と提案方式におけるファイル転送時間を測定し，提案方式の評価を行なう．

1 はじめに

近年のインターネットの急激な普及に伴い，一部の主要なサーバに対してアクセスが過度に集中する傾向が見られる．また，そのためにサーバ周辺のネットワークの混雑が激しくなるという問題も生じている．

サーバへの負荷を分散させる方法の一つとして，ミラーサーバを用いる方法がある．ミラーサーバとは，基となるマスタサーバから全てのデータをあらかじめコピーしておき，マスタサーバと全く同じ情報を提供するものである．これにより，特定のサーバやその周辺のネットワークに負荷が集中することを防ぐことができる．しかし，ミラーサーバを用いた場合，どのサーバを選択するのが望ましいかを何らかの手段で調べる必要がある．このような方式として，文献 [1] で挙げられているように，サーバ及びその周辺のネットワークに分散化システムを設置することにより分散を行なう方法が主として提案されている．しかし，これらのシステムでは，定期的にミラーサーバ群の負荷情報を収集する必要があるため，システムが大規模になる傾向がある．また，これらの方式は，サーバ側にとっての負荷分散であるため，その選択結果がクライアントにとって必ずしも最適であるとは限らなかった．

この問題を改善するため，本論文ではローカル DNS[2] サーバにミラーサーバの取捨選択を行なうフィルタ機能を持たせ，これによって，ユーザにとって適切なサーバを自動的に選択する DNS フィルタ方式を提案し，これを実装する．また，DNS ラウンドロビン方式と本提案方式におけるファイル転送時間を測定し，提案方式の評価を行なう．

2 サーバの選択方式

ミラーサーバを用いて分散されたサーバにユーザが効率的にアクセスするためには，どのサーバにアクセスするのが最も効率的に転送で

きるかをあらかじめ調べる必要がある．文献 [1] 等によれば，現在これを実現する方法として，ユーザが手動でサーバを選択する方式や，DNS によるラウンドロビン方式などのいくつかの方式がある．本節では，これらのうち，主な手法について概説する．

2.1 クライアント側で負荷分散をする

2.1.1 手動によるサーバ選択

最も原始的なサーバ選択方式は，各ユーザが最適なサーバを手動で選択することである．典型的なサーバの選択基準として，以下に示すものがある．

- ネットワーク的に近いサーバをユーザ自身が知っているならば，それを選択
- そうでなければサーバにかかる負荷などの情報を得て，その情報を基に選択

この方式は，全てのユーザがサーバの負荷状況を把握し，ネットワークに関する十分な知識を持っていることを必要とする．しかし，一般のユーザがこのような知識を持っていると期待することは大変難しい．

2.1.2 特別なクライアントを用いる

これはあらかじめ内部に特定のサーバ及びプロトコル向けの負荷分散機能を内蔵させた特別なクライアントを用いて負荷分散を行なうものである．当然，他のサーバやプロトコルにこの機能を用いることはできない．

2.2 中間システムで負荷分散

2.3 中継サーバ方式

手動方式を自動化する方法として中継サーバ方式がある．この方式では，まずミラーサーバ群の前に中継サーバを置き，ユーザをこのサーバにアクセスさせる．中継サーバはミラーサーバ群の中から適切なサーバを選択し，クライアントと選択されたサーバの間の通信を中継する．この方式はサーバ自体の負荷分散は可能だが，中継サーバにアクセスが集中するため，ネットワークの負荷分散はできない．また，プロトコル毎に中継サーバを作る必要がある上に，中継サーバそれ自体の性能がミラーサーバシステム全体のボトルネックになることがある．

2.4 IPv6 による Anycast

次世代インターネット・プロトコルである IPv6 には，負荷分散に使用できる “Anycast” という機能がある．これはネットワーク上でパケットを運ぶ個々のルータが，広範囲に散らばった複数のサーバのうちで最も適していると判断したある 1 台のサーバに向けてパケットを送り届けるというものである．

これは IP の機能であるため，プロトコルに全く依存せず応用範囲が大変広いが，Anycast の詳細がまだ不透明である上，Anycast を正しく理解できるルータが配置されている必要がある．そのため，現時点ではまだ現実的ではない．

2.5 サーバで負荷分散

2.5.1 HTTP リダイレクション

HTTP サーバの機能の一つに任意のアクセスをインターネット上の任意のサーバへ振り向ける機能がある．これを「リダイレクション」と呼ぶ．この機能を利用したサーバの選択方式もある．

すなわち，図 1 のように振り向け機能を持った代表サーバを置き，まず初めにユーザをそこ

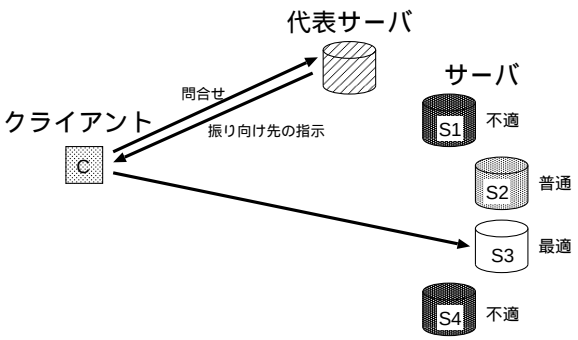

図 1: HTTP リダイレクションを用いた方式

にアクセスさせ，代表サーバがユーザの HTTP アクセスを適切なサーバへ振り向けさせる．この方式は非常に柔軟な設定が可能だが，HTTP のプロトコルに依存しているために他の種類のプロトコルでは使えない．更に，一旦選択させると各サーバの状況が変わってもそれに追従できないという欠点がある．

2.6 DNS を用いる負荷分散

2.6.1 DNS サーバを用いたラウンドロビン方式

一般に，ユーザがサーバを指定する際は，サーバの IP アドレスではなく，“www.example.com” 等の分かりやすい名前を用いる．DNS サーバはこの名前からそれに対応する IP アドレスを調べるサーバである．

DNS サーバでは，ある名前に対して複数の IP アドレスを登録できる．例えば図 2 の例では，あるホスト名に対して 4 つのサーバの IP アドレス S1 ~ S4 が登録されている．この名前の問い合わせに対して， DNS サーバは登録された IP アドレス全てを提示するが，その後， IP アドレスのリストは循環される．この方式をラウンドロビン方式と呼ぶ．クライアントは提示された IP アドレスの中から一つの IP アドレスを選択するが，その選択は完全にランダムであり，各サーバは均等に選ばれる．現在，この方式を用いてミラーサーバの負荷分散が実現されている．しかし，この方式はサーバの選択がサーバの状況とは関係なく決定されるため，最適なサーバが選択された場合と，そうではないサーバが選択された場合の処理時間の差が大きくなる．

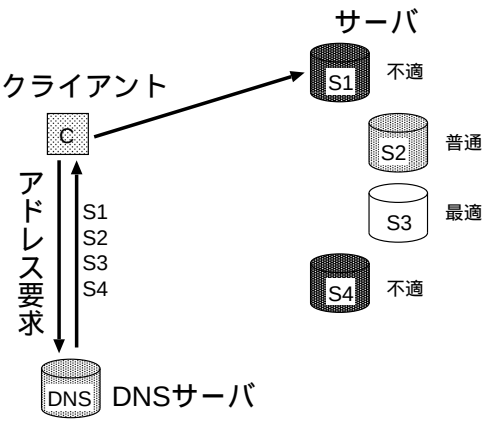

図 2: ラウンドロビン方式による負荷分散

2.7 TENBIN

TENBIN[3] は九州大学で開発された負荷分散 DNS サーバである．これはローカルの DNS サーバを置き換える形で設置する．また，これとは別にターゲットとなるサーバ群を定期的に調査するシステムを設ける．クライアントから調査対象サーバの IP アドレス及びその他の情報要求があると，ローカル DNS サーバは調査システムからの情報を基に最適なサーバの情報を選択して返答する．この選択にはさまざまな手法を採用できるが，文献 [3] では BGP-4[4] を用いてサーバ付近のルータから収集した経路情報に基づいた方式などが提案されている．但し，負荷分散の対象は調査システムがターゲットにしているシステムに限定される．

2.8 DNS Balance

著者らはこれまで，ミラーサーバにおける負荷分散システムとして DNS Balance[5][6] を設計，実装した．これは DNS サーバにロードバランサー機能を付加したもので，DNS サーバは各サーバの負荷状態を一定期間毎に収集し，このデータを基にクライアントに適しているサーバの IP アドレスを一つまたは複数返答する．その際，ある特定のサーバに処理が集中し過ぎないように，選択にはランダムな要素を加えている．

本方式によりサーバ側のある程度の負荷分散が実現された．しかし，TENBIN と同様に，この負荷分散は調査対象となるシステムに限定される．また，サーバとクライアントの間のネットワーク的な距離を考慮できないなどの問題点があり，本方式だけではクライアントにとって適切なサーバが必ずしも選択されなかった．

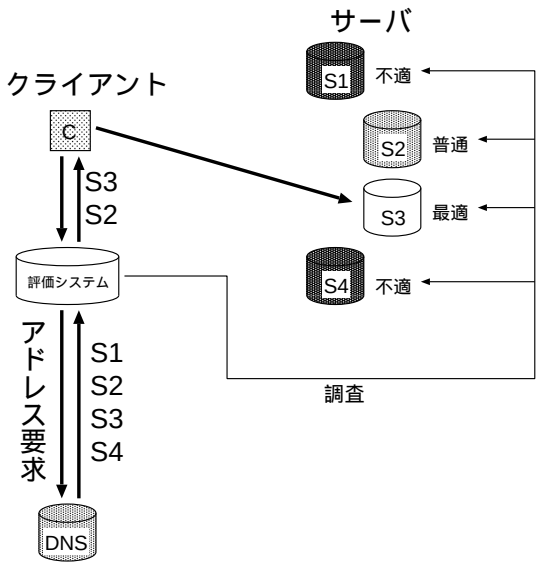

図 3: DNS フィルタ方式

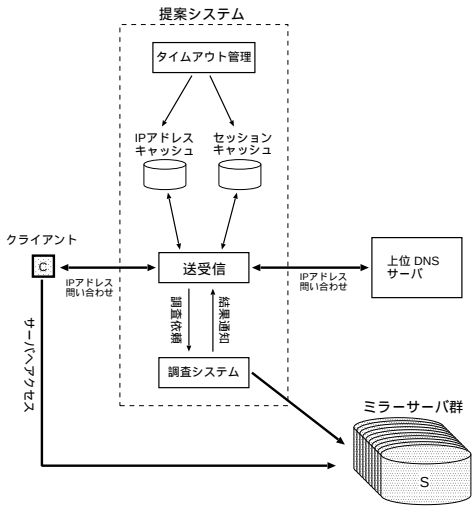

図 4: 提案方式の構造

3 DNS フィルタ方式

以上で述べた方式は 2.1.1 節の手動による方式を除き，全てサーバやその付近で負荷分散を行なうものである．文献 [1] には，サーバ及びその近辺に更に複雑な仕組みを置いて分散を行なう仕組みが幾つか紹介されている．しかし，これらの仕組みは複雑な分大規模になりがちで，その分散結果がサーバではなく，クライアントにとって必ずしも最適とは限らない．クライアントにとって良いサーバとはクライアント自身が自らの判断に基づいて自らが判別する必要があり，この点については 2.1.1 節の手動による方式が近いと言える．

そこで，本論文では，この手動による方式を自動的に行なうことを目的とする．具体的には，図 3 に示すように，クライアントの LAN に属する DNS キャッシュサーバとクライアントの間に評価システムを置く．そして，このシステムが DNS サーバから得られた IP アドレス群からクライアントにとって適切ではないものを取捨選択し，適切なサーバの IP アドレスだけをクライアントに返す．本提案方式を DNS フィルタ方式と呼ぶ．

さて，この評価システムは，クライアントが属するドメインにあるローカルな DNS サーバと置き換えることを想定している．通常，クライアントはローカルな DNS サーバに IP アドレスの問い合わせを行なうことから，これを提案システムで置換することでフィルタリング操作を容易に実装できる．また，この評価システムでは DNS サーバから IP アドレスが取得されてからそれらのサーバに対する調査を行なうため，TENBIN や DNS Balance とは異なり，評価対象を特定システムのみに限定されることがないという利点を持つ．

さて，提案システムは図 4 の点線枠内にある 3 つの処理システムと 2 つのキャッシュから構成することができる．以下では，その各々について説明する．

3.1 DNS パケットの送受信システム

まず，送受信システムではクライアントからの DNS パケットを受け取り，IP アドレスキャッシュに情報があれば上位 DNS サーバに問い合わせることなくキャッシュ中の情報をそのまま返す．

情報がなかった場合は，上位サーバに問い合わせてその返事を待つ．返事を受け取ったらセッションキャッシュを調べ，該当の情報があればクライアントにそのまま返事を返すと共に，その接続状況をセッションキャッシュから削除する．該当する情報がなければ，DNS サーバからの返事は捨てられる．前者の場合，DNS サーバから得た IP アドレス全てを IP アドレスキャッシュに登録し，これらの IP アドレスが即座に

回答できるようにする．更に，IP アドレスを示す A レコードの問い合わせに対して複数の A レコードの回答があった場合は，次節の調査システムを起動し，IP アドレスのフィルタリングを行なう．このフィルタリングの結果をクライアントに回答しても良いが，調査には時間が要すると想定される．そのため，フィルタリング前の結果をクライアントに回答しておき，調査終了後に問い合わせたクライアントに対して，フィルタリング後の IP アドレスを回答する実装も可能である．次節の実装システムでは，この方式を採用している．

ところで，DNS サーバが提供する情報にはそれぞれ有効期限が記入されている．通常は，サーバの IP アドレスが頻繁に変わることはないので，DNS システムではキャッシュを多用し，各情報の有効期限を長めに設定することで無駄な問い合わせを減らしている．しかし，提案方式による DNS サーバの場合，有効期限をある程度短くし，接続すべきサーバを再度調べさせる必要がある．そのため，有効期限がある一定以上の値であった場合，これを切り詰めて短くする．有効期限を短くした場合でも，ネットワーク的にすぐ近くに存在する上位の DNS サーバのキャッシュに問い合わせることが可能なので，これによるネットワークへの悪影響は少ない．

3.2 クライアントが接続すべきサーバの調査システム

調査システムでは，送受信システムが IP アドレスキャッシュに登録した IP アドレスのリストから，適切な IP アドレスを一つまたは複数選択する．そして，その調査結果を IP アドレスキャッシュに反映する．ただし，全ての IP アドレスへのアクセスが不可能であった場合は，全てのサーバの IP アドレスをキャッシュに登録したままにする．IP アドレスのリストを完全にキャッシュから削除してしまうと，クライアントから再アクセスされた場合，再調査を行なう必要がある．しかし，次回に同じ調査をしたとしても回答が得られないことが予想される．従って，キャッシュを削除することは，ネットワークに不必要なオーバヘッドをかけることになる．

3.3 IP アドレスキャッシュ

IP アドレスキャッシュにはホスト名とそれに対応する IP アドレス，及び，それぞれの情報の有効期限が保存される．

3.4 セッションキャッシュ

DNS の問い合わせに一般的に用いられる UDP プロトコルは，TCP のようにセッションを作ることができない．このため，DNS システムではパケットに ID 番号を含めるなどして個々の問い合わせを区別し，疑似的にセッションを維持する．この接続情報を保存するのがセッションキャッシュである．このキャッシュには個々の接続情報のほか，それぞれの接続情報の有効期限も設定される．

3.5 キャッシュデータのタイムアウト管理システム

DNS の情報には有効期限があり，IP アドレスキャッシュ中の古くなった情報は廃棄する必要がある．また，ある問い合わせに対して一定期間以上回答がなかった場合も，セッションが切れたと判断して，問い合わせに関する情報をセッションキャッシュから廃棄する必要がある．

そのため，タイムアウト管理システムは常に各キャッシュを監視して，有効期限が切れたデータをキャッシュから取り除く．

4 DNS フィルタ方式の実装

前節の検討を踏まえ，提案システムを “DNS Trick” という名前で実装した．本システムの実装には Ruby[7] を用いた．Ruby はサーバの設計に便利なマルチスレッドや，エラー処理に便利な例外処理構文を持っており，今回の実装には最適であると判断した．

本システムで採用したサーバ選択システムは以下の 2 つである．これらは調査の精度と速度に関して違いがある．

4.1 ICMP ECHO パケットに対する応答速度を用いた方法

まず，高速かつ低精度な方法として ICMP ECHO パケットを用いる方式を使用する．すなわち，調査の依頼があると，各ミラーサーバに向けて同時に ICMP ECHO パケットを送信する．これを受信したサーバはそれに対する回答として ICMP ECHO REPLY パケットを返す．これにより，最も回答の早かった一つまたは複数のサーバを選択する．

4.2 netselect を用いた方法

netselect[8] は， Avery Pennarun が作成した ICMP と UDP を用いて精密に応答速度を検証するツールである．但し，その調査には約 30 秒程度かかる．第 2 の方法では，この netselect の出力を用いて選択を行なう方式を用いる．

5 評価実験

ミラーサーバを積極的に活用したシステムとして ,RingServer プロジェクト [9] の RingServer がある．これは，インターネット等の高速ネットワーク環境を対象に，大規模なソフトウェアライブラリとソフトウェアの分散共同開発の支援を行う共通基盤技術の構築を目的としたプロジェクトで，その研究の一環として日本各地に分散された，同一の内容を持つサーバを運営している．本節では，この RingServer から提案方式と 2.6.1 節の DNS ラウンドロビン方式 (従来方式) の各方式でファイルを転送させた時の，転送時間の比較を行い，提案方式の評価を行なう．なお，提案方式において DNS サーバから得られた IP アドレスから選択する IP アドレスの個数は 1 個とし，IP アドレスの最大有効期限を 15 分とした．また，セッションキャッシュの有効期限は 60 秒とした．

5.1 巨大ファイルの転送

まず，当研究室において RING サーバから http プロトコルにより巨大なファイルを取得させ，提案方式と従来方式を用いた時の転送時間の比較を行なう .転送させるファイルは`http://-www.ring.gr.jp/archives/GNU/emacs/emacs--20.7.tar.gz` で，ファイルサイズは約 14.3 メガバイトである．このファイルを提案方式と従来方式で同時に，一定時間毎に転送させ，その転送時間を測る．ここで，提案方式が ICMP ECHO パケットを用いた方式の場合は，ファイル転送後，5 分待機を行ない，再度転送を繰り返した．netselect を用いた方法の場合は，指定した時間に自動的にプログラムを実行する cron プログラムを用いて，30 分おきにファイル転送を行なった．そしてこれらを 24 時間繰返し実行させた．

ICMP ECHO パケットを用いた場合の転送時間と従来方式と提案方式における差をそれぞれ図 5 と図 6 に示す．また，netselect を用いた場合のそれを図 7 と図 8 に示す．この結果より，ほとんどの場合で従来方式よりも提案方式の方が転送時間が少なくなっていることが分かる．また，各方式のファイル転送 1 回当たりの平均転送時間を比較すると，ICMP ECHO パケットを用いた提案方式では 96.90 秒，その時の従来方式では 279.7 秒であった．また，netselect を用いた提案方式では 213.2 秒，その時の従来方式では 281.7 秒であった．これは，図 9 に示す netselect を用いた場合の各サーバからのファイル転送回数を見ると分かるように，提案方式ではクライアントからネットワーク的に距離の近いサーバが数多く選択されているのに対し，従来方式では距離に関係なく選択されていることから，ネットワーク的に遠い距離にあるサーバをより多く選ぶ従来方式の方がファイル転送時間が多くなったためである．

一方，一部の転送時に提案方式の方が極めて転送速度がかかる場合が見られた．例えば，図 8 では転送時間が 3780 秒である時刻がある．これは，上位 DNS サーバにキャッシュされた個々の IP アドレスの有効期限にずれがあり，クライアントからネットワーク的に近い IP アドレ

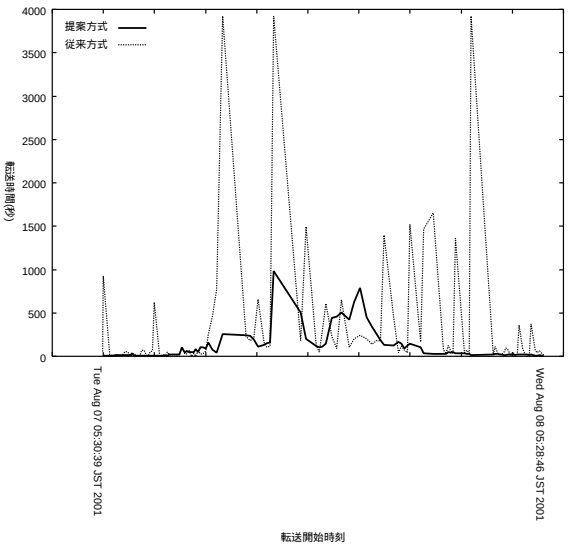

図 5: 巨大ファイルの転送時間比較 (ICMP ECHO パケットを用いた場合)

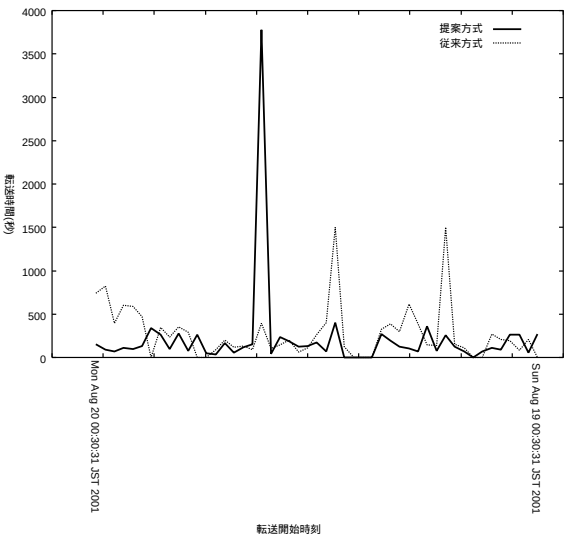

図 7: 巨大ファイルの転送時間比較 (netselect を用いた場合)

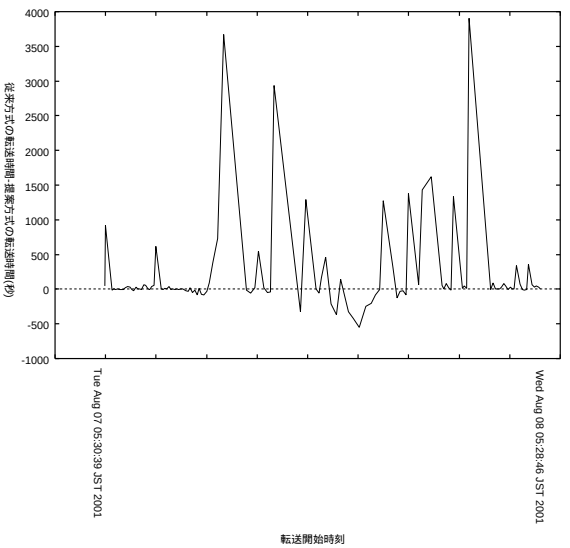

図 6: 従来方式と提案方式の巨大ファイルの転送時間の差 (ICMP ECHO パケットを用いた場合)

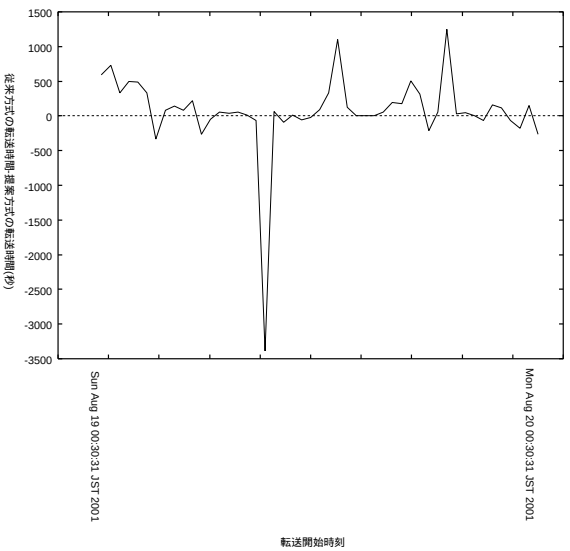

図 8: 従来方式と提案方式の巨大ファイルの転送時間の差 (netselect を用いた場合)

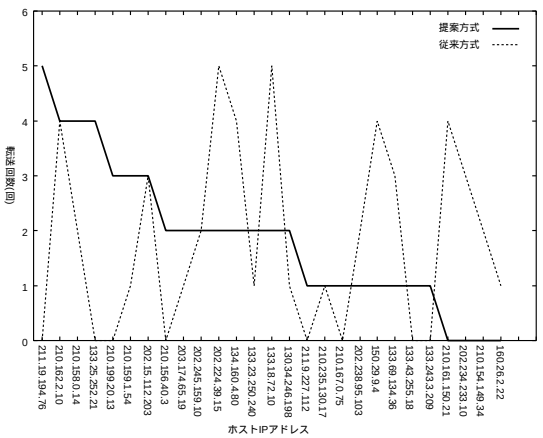

図 9: 巨大ファイル転送時の転送回数比較 (netselect を用いた場合)

スの多くがタイムアウトしたことによる．このため，ネットワーク的に遠いサーバの集合内からあるサーバが選択され，ファイル転送時間が増大することとなった．

ところで，ICMP ECHO パケットと netselect を用いた場合を比較すると，後者の方が転送時間が大きいことが分かる．netselect は ICMP ECHO よりも精密な応答速度を測定すると期待されるが，本提案方式では有効に働かなかった．これは，ICMP ECHO 方式と netselect 方式において，調査のために各サーバに送出されるパケットに対して，ICMP ECHO 方式ではほとんどのサーバが応答を返したのに対し，netselect 方式の場合は半数以下のサーバのみしか応答を返さなかった．このため，netselect 方式ではうまくホストを選択できなかったものと考えられる．netselect は，わざと通信エラーを引き起こすようなパケットを送出してそれに対する応答からサーバ選択を行なうため，セキュリティ上の観点からこのようなパケットが途中で排除されたと推定される．

5.2 多数ファイルの転送

次に，当研究室において RING サーバから http プロトコルにより多数のファイルの一括転送を行ない，提案方式と従来方式を用いた時の転送時間の比較を行なう．転送対象は`http://-www.ring.gr.jp/pub/doc/RFC/`にある`rfc1000.-txt` から `rfc1100.txt` の 101 個のファイルである．これらのファイルのファイルサイズは最小の `rfc1061.txt` の 29 バイトから最大の `rfc-1000.txt` の 315315 バイトまでさまざまであり，1 ファイルあたり平均で約 60 キロバイトある．これらのファイルを提案方式と従来方式で同時に，一定時間毎に転送させ，その転送時間の合計を測定する．ここで，提案方式は ICMP ECHO パケットを用いた方法と netselect を用いた方法共に，cron プログラムを用いて 5 分毎にファイル転送を行ない，これを 24 時間繰返し実行させた．

ICMP ECHO パケットを用いた場合の総転送時間と従来方式と提案方式における差をそれぞれ図 10 と図 11 に示す．また，netselect を用いた場合のそれを図 12 と図 13 に示す．

この結果からもほとんどの場合で従来方式よりも提案方式の方が転送時間が転送時間が少なくなっていることが分かる．また，各方式のファイル転送 1 回あたりの平均総転送時間を比較すると，ICMP ECHO パケットを用いた提案方式では 96.63 秒，その時の従来方式では 477.9 秒であった．また，netselect を用いた提案方式では 573.6 秒，その時の従来方式では 2432 秒であった．図 14 に示す netselect を用いた場合の各サーバからのファイル転送回数を見ると，提案方式ではクライアントからネットワーク的に距離の近いサーバが数多く選択されているのに対し，従来方式では距離に関係なく選択されている．従って，前節の場合と同様に，ネットワーク的に遠い距離にあるサーバをより多く選ぶ従来方式の方が，ファイル転送時間が多くなった．

ところで，図 9 と図 14 を比較すると，サーバの選択傾向が似ているにも関わらず，一つの巨大ファイルを転送した時よりも多数ファイルを転送した時の方が，従来方式と提案方式のファイル平均転送時間の差が大きい．後者の実験では，サーバへの接続回数が多いため，従来方式の場合に，ネットワーク的に遠いサーバに接続する回数が増加することに起因する．

さて，前節の結果と同様に，一部の時刻で提案方式で極端に悪くなる時があった．この原因は前節の場合と同じであるが，全転送回数に対する提案方式の方が転送時間が少ない時の

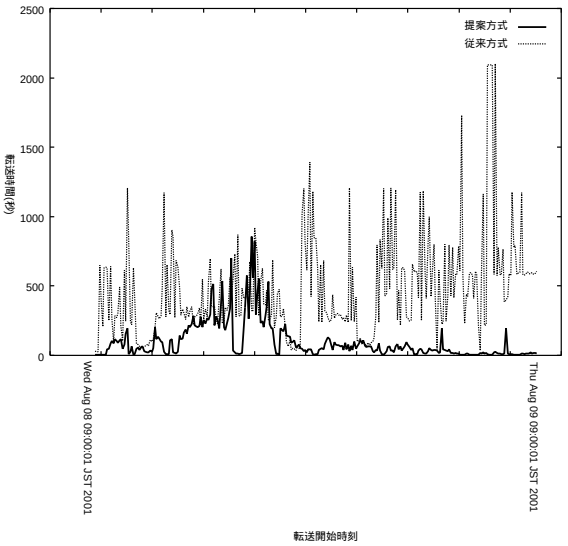

図 10: 多数ファイルの総転送時間比較 (ICMP ECHO パケットを用いた場合)

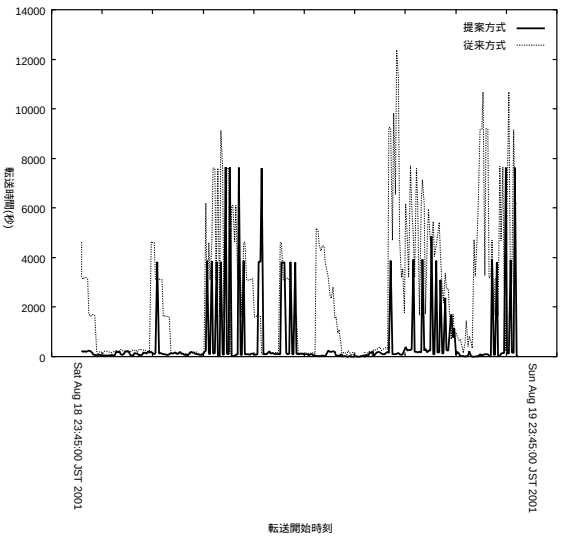

図 12: 多数ファイルの総転送時間比較 (netselect を用いた場合)

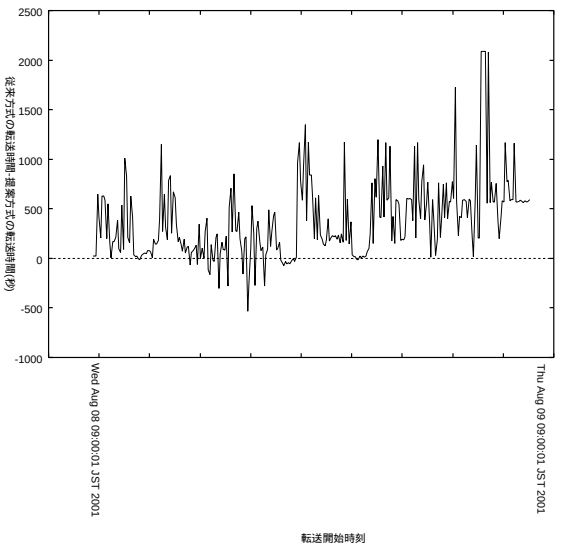

図 11: 従来方式と提案方式の多数ファイルの総転送時間の差 (ICMP ECHO パケットを用いた場合)

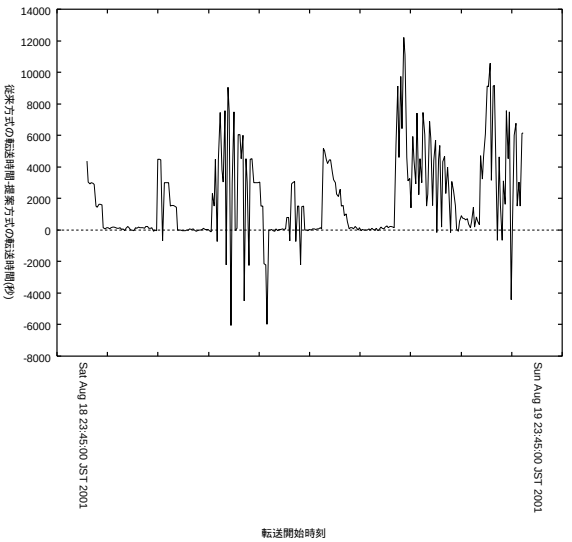

図 13: 従来方式と提案方式の多数ファイルの総転送時間の差 (netselect を用いた場合)

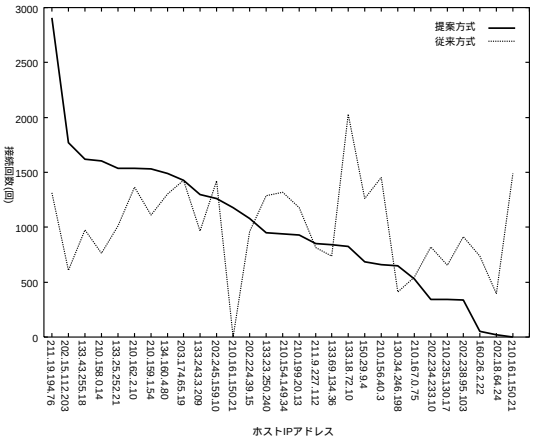

図 14: 多数ファイル転送時の転送回数比較 (netselect を用いた場合)

回数の割合は前節の場合よりも増加している．すなわち，ICMP ECHO パケットを用いた場合は，55.9%から 82.7%に，netselect の場合は 59.2%から 77.2%になった．これは IP アドレスキャッシュの効果により，問い合わせの回数が多い多数ファイルの転送の方が適切なサーバを選べたためである．

6 まとめ

本論文では，ミラーサーバシステム等の複数サーバシステムにおいてサーバを選択する DNS フィルタ方式を提案，実装し，その有効性を確かめた．

今後の課題としては，提案システムによるアクセス時間へのオーバーヘッドや負荷試験等を行なう．また，更に優れたホスト選択手法の開発も必要である．今回作成したシステムはクライアントのローカル DNS として設置して，ネットワーク的な距離を基準にサーバを評価するものであった．一方，2.8 節で述べた DNS Balance はサーバの負荷状況を評価するものであり，サーバの効率的運用を目指すものである．このシステムと本論文のシステムを連係させることでサーバ及びクライアント双方により良いシステムが構築できるものと思われる．

謝辞

本論文の執筆に当たり，本システムの評価に協力していただいた RingServer プロジェクトの管理者グループの皆様，そして，提案方式の実装に使用させていただいた，さまざまなソフトウェアの作者の方々に深く感謝します．

参考文献

[1] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, Vol. 3, No. 3, pp. 28–39, 1999.

[2] P. Mockapetris. Domain Names — Implementation and Specification, 1987. RFC1035.

[3] T. Shimokawa, N. Yoshida, and K. Ushijima. DNS-based Mechanism for Policy-added Server Selection. In *Int'l Conf. on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, CD-ROM*, 2000. http://www.tenbin.org/.

[4] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4), 1995. RFC1771.

[5] 横田 裕思. 分散環境向け負荷分散 DNS サーバの設計と大規模サーバシステムにおける性能評価. 筑波大学大学院修士課程理工学研究科修士論文, 2001.

[6] 横田 裕思. DNS Balance. http://www.-netlab.is.tsukuba.ac.jp/~yokota/izumi/-dns_balance/.

[7] まつもと ゆきひろ. オブジェクト指向スクリプト言語 Ruby. http://www.ruby-lang.org/.

[8] Avery Pennarun. Netselect. http://www.-worldvisions.ca/~apenwarr/netselect/.

[9] RingServer Project. http://www.ring.gr.-jp/.