高品質インターネット　4－4
(2002. 7. 26)

複数のWebサーバにおけるDNSを用いた負荷分散

小野村哲也† 稲井 寛†

† 岡山県立大学情報工学部 〒719-1197 岡山県総社市窪木111
E-mail: †{onomu,inai}@c.oka-pu.ac.jp

あらまし 複数台のWebサーバに負荷を分散させる技術の1つとして，DNSを用いた方法がある．この方法は，DNSラウンドロビンと呼ばれるもので，URLに対応する1つのホスト名に対し，実際に稼働させる複数のWebサーバのIPアドレスを登録しておき，アドレス解決の要求がある度にラウンドロビンによって異なるIPアドレスを返すことで，クライアントからのページ閲覧要求の処理を分散させる．しかし，この方法では，各Webサーバの処理能力に関係なく純粋なラウンドロビンによりIPアドレスの回答を行うことや，クライアント側のDNSサーバがアドレス情報をキャッシュする仕組みにより，負荷分散が有効に機能しない場合が多い．そこで，本論文では，アドレス情報のキャッシュ期間(TTL)を調整することにより，効果的な負荷分散を行う方法について，シミュレーションにより検討を行っている．

キーワード WWW，負荷分散，DNS，TTL，ラウンドロビン

Load Balancing in Multiple Web Servers by using DNS Round Robin

Tetsuya ONOMURA† and Hiroshi INAI†

† Faculty of Computer Science and System Engineering, Okayama Prefectural University
Kuboki 111, Soja-shi, Okayama, 719-1197 Japan
E-mail: †{onomu,inai}@c.oka-pu.ac.jp

Abstract Load balancing in multiple Web servers by using DNS is called DNS Round Robin. This method registers IP addresses of Web servers for one host name corresponding to URL, and return an IP address in Round Robin fashion whenever it receives a query of address resolution. In this way, accesses from clients are distributed each Web server. However, in this method, there are many cases that load balancing does not work effectively. So, this paper examines performance of an effective load balancing method which adjusts cache period of address information.

Key words WWW, Load Balancing, DNS, TTL, and Round Robin.

1. まえがき

近年，インターネットでは，ユーザ数の急激な増加に伴って，Web サービスを提供するサイトへのアクセスが増大している．アクセス件数の多い Web サイトでは，サーバの処理能力が追いつかず，最悪の場合には，サーバがダウンしてサービスの停止を招く可能性がある．そこで，この解決策として，同一コンテンツを提供するサーバを複数台設けて負荷を分散させることが試みられている．複数台のサーバを設置した場合には，1 台のサーバがダウンしても，残りのサーバが処理を引き継ぐことで耐障害性を向上でき，更には，アクセス件数の増加と共にサーバを増設できるという拡張性におけるメリットもある．

このように，サーバを複数台設置する場合においては，ユーザからのリクエストをどのように各サーバに振り分けて負荷を分散させるかが問題となる．振り分けの方法として，(1) 専用の負荷分散装置（ロードバランサ）[5] [6] を設置する方法，(2)DNS を用いる方法（DNS ラウンドロビン）[1]，(3)OS に負荷分散の機能を持たせる方法などがあるが，それぞれに一長一短がある [7]．これらの中で，本論文では，(2) の DNS ラウンドロビンを検討の対象とする．

DNS ラウンドロビンでは，クライアントが Web サーバにアクセスするとき，必ずサーバ側の DNS サーバから Web サーバの IP アドレスの情報を引き出す仕組みを利用している．サーバ側の DNS サーバに，1 つのホスト名に対して実際に稼働させる複数の Web サーバの IP アドレスを登録し，アドレス解決の問い合わせごとに順番に異なる Web サーバの IP アドレスを返すことによって，クライアントからのアクセスを分散させる．この方式による負荷分散では，クライアントからの要求を処理する Web サーバの処理能力に関係なく，純粋なラウンドロビンでしかアドレスを循環させない点や，クライアント側の DNS サーバがアドレス解決の情報をキャッシュする仕組みにより，負荷の分散が有効に機能しない場合が多い [1] [8]．

この問題に対して，文献 [3] において，DNS サーバがアドレス情報をキャッシュする期間である TTL(Time To Live) を調整することにより，効果的な負荷分散を行う手法が提案されている．[3] の手法では，TTL 算出において，ある DNS サーバを参照しているクライアントからのページ閲覧要求の負荷を利用しているが，この負荷を適確に推定することは困難であると考えられる．また，[3] の手法により算出される TTL の値は，例えば，数分間のように非常に短いものとなるため，インターネット上を流れるアドレス解決要求のトラヒックが増大する可能性が高い．

これに対して，本論文で提案する手法では，TTL の算出の際に確実に推定することが可能な Web サーバの処理能力のみを用いている．さらに，算出された TTL の平均を，文献 [9] で適切な値として紹介されている 1 時間とすることが可能であるため，アドレス解決要求の増加によるトラヒックの増大を抑えることができる．

以下，2. では，DNS を用いた負荷分散方法について説明し，3. では，本論文で提案する TTL の算出方法ついて述べる．4. では，この TTL 算出法の有効性を評価するためのモデルを構築する．5. では，構築したモデルを用いてシミュレーションを行い，TTL を調整した場合の各 Web サーバのレスポンスタイムについて検討を行う．最後に，6. で，本研究で得られた結果のまとめ及び今後の課題について述べる．

2. DNS ラウンドロビン

DNS ラウンドロビンによる負荷分散の様子を図 1 に示す．図 1 の場合では，www.A.com というホスト名に対し，3 台の Web サーバの IP アドレス x.y.z.1, x.y.z.2, x.y.z.3 をサーバ側の DNS サーバに登録している．クライアントから Web サイトにリクエストがある場合，クライアントはまず最寄りの DNS サーバ（以降，クライアント側の DNS サーバという）にアドレス解決要求を出す．このとき，クライアント側の DNS サーバに Web サーバのアドレス情報がキャッシュされていなければ，クライアント側の DNS サーバは，サーバ側の DNS サーバにアドレス解決要求を送る．サーバ側の DNS サーバはアドレス解決の要求があるたびに順番に異なる Web サーバの IP アドレスを答えることで 1 台の Web サーバへアクセスが集中するのを防いでいる．

上述のように，DNS サーバは一度解決したことのあるアドレス情報をキャッシュすることによって，アドレス解決要求によるトラヒックの増大を回避している．キャッシュされた情報の有効期間は TTL と呼ばれており，その値は大元の DNS サーバ（図 1 の例では Web サーバ側の DNS サーバ）に登録されている．取得後，TTL を経過したアドレス情報はキャッシュから削除される．したがって，TTL を短く設定すると，クライアント側の DNS サーバは頻繁にアドレス解決にくるため，サーバ側の DNS サーバにはアドレス解決要求が集中し，その結果，アドレス解決に要する時間がかかってしまう．しかし，その反面，Web サーバの IP アドレスの循環が頻繁になり，効果的な負荷の分散が実現できる可能性が高くなる．逆に，TTL を長く設定すれば，アドレス解決の要

求頻度は減り，ネットワーク上のトラヒックは減少するが，Web サーバの IP アドレスの循環の間隔が大きくなり，均等に負荷を分散させるのは困難となる．

したがって，DNS ラウンドロビンにより負荷分散を行う場合は，キャッシュ時間である TTL の設定が重要となる．そこで，本論文では，効果的な負荷分散を行うために適切な TTL 値を算出する方法について検討する．

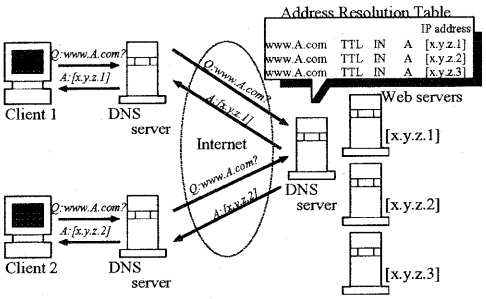

図 1　DNS ラウンドロビンによる負荷分散

3. TTL の算出方法

以降では，説明の都合上，クライアント側の DNS サーバとその DNS サーバをアドレス解決に使っているクライアントから構成される計算機の集合をドメインと呼ぶことにする．前章で述べたように，クライアント側の DNS サーバは Web サーバのアドレス情報を TTL の期間キャッシュしているので，この期間中ドメイン内の全てのクライアントは同じ Web サーバにページ閲覧要求を送信することになる．ここで，仮に各ドメイン内クライアント数が等しく，かつ各 Web サーバの処理能力も等しいとすると，サーバ側の DNS サーバは TTL を一定値に設定して，純粋なラウンドロビンにより Web サーバのアドレス回答を行うことにより，効果的な負荷の分散を実現することができる [3]．

しかし，現実にはドメイン内クライアント数は互いに等しいとは考え難く，また，Web サーバの処理能力も互いに異なっている場合があるものと思われる．Web サーバの処理能力が異なっている場合には，各処理能力に応じた重み付けラウンドロビンによりサーバの割当てを行えばよいと考えられる．そこで，本論文では，サーバ側の DNS サーバは重み付けラウンドロビンに従って Web サーバのアドレス回答を行うこととしている．一方，本章で定義したドメインは，実際には互いに重なり合っている場合が多いものと考えられるため，ドメイン内クライアント数を特定することは困難であるものと思われる．これは，インターネット上に散在する DNS サーバが階層構造を成しており，下位の DNS サーバでアドレス解決ができない場合には順次上位の DNS サーバに問合わせを行うためである．そこで，本論文では，Web サーバの割当てや TTL の算出に際して，特定の困難なドメイン内クライアント数やドメインごとのトラヒック量などは除外し，確実に特定可能な Web サーバの処理能力を考慮に入れることにする．

本論文では，サーバの割当てについては，上述のように重み付けラウンドロビンとする．また，TTL の設定については，処理能力の低い Web サーバのアドレス情報ほど短い TTL を設定する．この理由は，クライアント数の多いドメインからのページ閲覧要求が処理能力の低い Web サーバに送信される期間が長く続くと，そのサーバのレスポンスタイムが大きくなる期間も長くなると考えられるからである．したがって，処理能力の低い Web サーバのアドレス情報には短い TTL を設定して，サーバ割当ての見直しを頻繁に行うべきである．これに対して，処理能力の高い Web サーバのアドレス情報については，TTL をやや長めの値に設定しても，レスポンスタイムに及ぼす影響はそれほど大きくないものと思われる．そこで，ここでは，TTL を Web サーバの処理能力に比例した大きさに設定することとした．

処理能力の最も低い Web サーバのアドレス情報の TTL を $\overline{T}$ とする．また，Web サーバ i の処理能力を C_i とする．このとき，Web サーバ i のアドレス情報の TTL である T_i を次のようにして算出することにする．

$$T_i = \begin{cases} \frac{\alpha C_i \overline{T}}{\min C_j} & \left(\frac{C_i}{\min C_j} > 1\right) \\ \overline{T} & \left(\frac{C_i}{\min C_j} = 1\right) \end{cases} \qquad (1)$$

但し，α (≥ 1) はパラメータで，この値を大きく設定すると，処理能力の高い Web サーバのアドレス情報の TTL も大きくなる．

パラメータ α の適切な値については，具体的な数値例をあげて 5. で検討する．ここでは，α の値が既に決定されているとした場合の $\overline{T}$ の値について考える．前章で述べたように，TTL を短くするとインターネット上を流れるアドレス解決要求のトラヒック量が増大するため，TTL は比較的大きな値としておく必要がある．文献 [9] によると，TTL として 1 時間程度が適当であるとされている．本論文では，Web サーバごとに TTL の値が異なるため，それらの平均値 L がある値，例えば，1 時間となるように $\overline{T}$ を設定する．

重み付けラウンドロビンにより Web サーバの割当てを行う場合，N 台の Web サーバの中からサーバ i が選択される確率 P_i は

$$P_i = \frac{C_i}{\sum_{k=1}^{N} C_k} \qquad (2)$$

であるから，TTL の平均 L は

$$L = \sum_{i=1}^{N} P_i T_i = \sum_{i=1}^{N} \frac{C_i T_i}{\sum_{k=1}^{N} C_k} \qquad (3)$$

式 (3) に式 (1) を代入すると $\overline{T}$ についての方程式が得られるので，これを解いて $\overline{T}$ を求める．

4. シミュレーションモデル

ここでは，提案する DNS ラウンドロビンの性能評価を行うためのシミュレーションモデルについて述べる．図 2 のように，あるドメインに所属するクライアントからリクエストが発生し，リクエストは 1 つの Web サイトに送信される．その Web サイトでは複数台の Web サーバが設置されており，リクエストの処理を行うサーバは前章で提案した方式に従って決定される．尚，本モデルでは，DNS ラウンドロビンのみの有効性を検討するために，TCP コネクションの確立や DNS によるアドレス解決に要する遅延，ネットワーク上の伝送遅延を無視している．

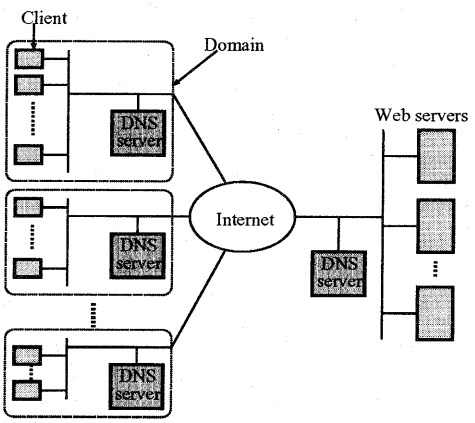

図 2 シミュレーションモデルの概要

4.1 ドメイン内のクライアント数

ドメイン内クライアント数の分布は，文献 [3] と同様，ジップの法則 [2] に従った分布と仮定している．すなわち，i 番目のドメインに属するクライアント数の比率は $1/i^{1-x}$ で表されるものとしている．ここで，$x(0 \leq x \leq 1)$ はパラメータであるが，クライアント数の多いドメインと少ないドメインでのクライアント数の差が大きくなるように $x = 0$ とした．

4.2 クライアントからのリクエスト発生モデル

多くのクライアントは 1 回のセッションで，Web サイトの提供する複数の HTML ページにリクエストを送信する．1 つのページには画像ファイルなど様々なファイルが存在していることが多いため，クライアントが 1 ページのリクエストを行った場合，Web サーバに対して複数のファイル転送要求がバースト的に発生することになる．そこで，本論文では，図 3 のようなリクエスト発生モデルを考えた．クライアントから Web サーバへは，1 ページ分のリクエスト（複数のファイル転送要求）が転送され，ページ閲覧のための時間が経過すると再び Web サーバへリクエストが転送される．ここで，1 ページ当りのファイル数は 5〜15 個の一様分布，1 セッションでのリクエストページ数は平均 12 ページの指数分布，リクエスト発生間隔は平均 25 秒の指数分布に従うこととする [4]．また，1 セッションが終了すると，そのクライアントは現在所属しているドメインを退去し，新たなクライアントが別のドメイン内に発生するものとする．そして，その新たな所属先は前節で述べたジップの法則に従った割合により決定されるものとしている．したがって，クライアントの総数 M は常に一定に保たれている．

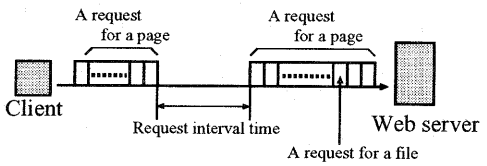

図 3 リクエスト発生モデル

4.3 Web サーバでの処理

各クライアントから出されたファイル転送要求は，図 4 のようにセッションごとに待ち行列に並べられる．Web サーバはこれらの要求を 1 ファイルごとにラウンドロビンで処理する．各々のファイル転送要求の処理に要する時間については，簡単化のため一定としている．

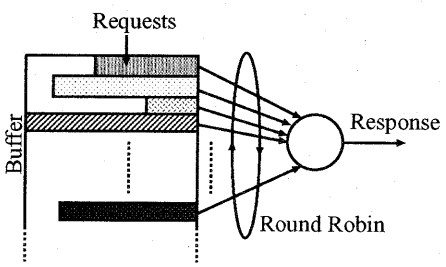

図 4 Web サーバの処理モデル

5. 数 値 例

本章では，各 Web サーバのレスポンスタイムを評価測度として，3. で提案した TTL 算出法の有効性について検討する．ここで，レスポンスタイムとは，クライアントの 1 ページ分のリクエストが Web サーバのバッファに

格納されてから処理されるまでの時間としている．

以降，特に断らない限り，サーバ数 $N = 8$，ドメイン数 $D = 100$，クライアント数 $M = 2000$，TTLの平均値 Lを3600秒とする．表1に本章で数値例として用いた各Webサーバの処理能力比の一覧を示す．この表では，処理能力の最も高いWebサーバの処理能力を1としている．そして，処理能力1とは，1秒間に500個のファイル転送要求を処理することができる能力としている．

以降で示すグラフでは，縦軸にレスポンスタイム，横軸に各WebサーバのID番号を表示している．これらのWebサーバの処理能力比は，表1に並べている処理能力比に順に対応している．したがって，グラフ上の曲線が水平に近いほど効果的な分散が行われていることになる．

表1　各Webサーバの処理能力比

Case1	1,1,1,1,0.8,0.8,0.8,0.8
Case2	1,1,1,1,0.5,0.5,0.5,0.5
Case3	1,1,1,1,0.4,0.4,0.4,0.4
Case4	1,1,1,1,0.3,0.3,0.3,0.3
Case5	1,1,1,1,0.2,0.2,0.2,0.2
Case6	1,1,1,1,0.1,0.1,0.1,0.1
Case7	1,1,1,1,0.5,0.5,0.1,0.1
Case8	1,1,1,1,0.4,0.4,0.1,0.1
Case9	1,1,1,1,0.3,0.3,0.1,0.1
Case10	1,1,1,1,0.2,0.2,0.1,0.1

まず，3.のTTL算出法でパラメータとして残しておいたαの値がレスポンスタイムに及ぼす影響を調べることにする．ここでは，表1のCase1〜6のように4台のWebサーバの処理能力を1に固定し，残りの4台のWebサーバの処理能力を0.8から0.1に変化させた場合のレスポンスタイムについて考察する（図5〜10）．$\overline{T}$を決めるときに，αを大きく取れば$\overline{T}$は小さい値となり，高い処理能力のWebサーバのTTLは長くなる．したがって，αの上限値は，$\overline{T} = 1$として式(3)から求まる値となる．各図中の表記Aが，αの上限値である．ここで，$\overline{T} = 1$としたのは，BIND4には，TTL値を0に設定したアドレス情報を返すことができないものが存在するため[1]である．処理能力の差が大きな場合には，Aが小さく，処理能力の差が小さい場合には，Aは大きくなる．

図5〜10にそれぞれCase 1〜6の場合の各サーバのレスポンスタイムを示す．また，比較のため，各グラフには，TTLを一定値の3600秒とした場合のレスポンスタイムも示している．図5 (Case 1)，6 (Case 2)のように処理能力の差が小さい場合には，αを小さな値に設定して，処理能力の低いサーバのTTLもある程度大きくしておく必要がある．特に図5 (Case 1)のように各サーバの処理能力にほとんど差がない状況で，αに大きな値を設定してしまうと，処理能力の低い5〜8のサーバのTTLは非常に短くなってしまうため，事実上1〜4の4台のサーバでほとんどのリクエストを処理しなければならなくなる．この結果，サーバ1〜4のレスポンスタイムが大きくなっている．Case 1のような場合において，クライアント数が増加し，到着するリクエストが多くなるとサーバ1〜4のレスポンスタイムは更に大きくなると考えられる．したがって，サーバの処理能力の差が小さく，負荷が高い場合にはαの値を小さく設定する必要がある．一方，図7 (Case 3)〜10 (Case 6)のように処理能力の差が大きい場合には，αを大きな値に設定して，処理能力の低いサーバ5〜8のTTLを小さくし，これらのサーバに負荷をかけないようにする必要がある．

以上の結果から，Webサーバの処理能力が2種類の場合で，式(1)のようにTTLを調整して効果的な負荷分散を実現するためには，サーバの処理能力の差が小さい場合ではパラメータαを小さな値（例えば1）に設定し，処理能力の差が大きい場合ではαを大きな値（例えば上限値A）に設定する必要がある．αの適切な値については，事前にシミュレーション等を行って決定しておく必要があるが，一般的な傾向として，αを小さな値に設定すべき処理能力差の範囲は，負荷が高い場合，また，処理能力の高いサーバ数が少ない場合には，狭くなると考えられる．例えば，ここで示した数値例では，Case 1と2の場合にαを小さな値に設定すればよいが，負荷が高くなるあるいは処理能力の高いサーバ数が少ないと，Case 2の場合でもαに大きな値を設定しなければならない場合が出てくると考えられる．

ここまでは，サーバの処理能力が2種類の場合について検討したが，現実には3種類以上の場合も考えられる．そこで，以降では，サーバの処理能力が3種類の場合について検討する．Case 7〜10では，処理能力が1，0.1のサーバがそれぞれ4，2台で，残りの2台の処理能力を0.5から0.2の範囲で変化させている．これらの場合のレスポンスタイムを図11に示す．いずれの場合も，処理能力0.1のサーバが存在するため，αの値を上限のAに設定している．図から，サーバ5，6の処理能力が低くなるに従って，これらのサーバのレスポンスタイムが大きくなっていることが分かる．これは，最も処理能力の低いサーバ7，8に合わせて$\alpha = A$としたため，サーバ5，6のTTLが大きめの値となってしまい，これらのサーバに負荷がかかることとなったためと考えられる．

この問題を解決するために，ここでは，処理能力の低い4台のサーバ5〜8をグループ化して，TTLを算出することを考える．グループ化されたサーバの各々の処理能力は，グループ内のサーバの処理能力の平均値とした．

このグループ化を一般的に表現すると次のようになる．ある処理能力（cとする）以下の処理能力を持つサーバの集合を$\mathcal{S}$とする．$\mathcal{S}$に属するサーバの処理能力を次に定める$\overline{C}$とする．

$$\overline{C} = \frac{\sum_{i \in \mathcal{S}} C_i}{n[\mathcal{S}]} \tag{4}$$

但し，$n[\mathcal{S}]$は集合$\mathcal{S}$の要素の数である．

Case 7〜10でグループ化を行った場合と行わなかった場合のレスポンスタイムの比較をそれぞれ図12〜15に示す．図12のCase 7の場合においては，図11から元々グループ化を行う必要がなく，グループ化を行うとかえって処理能力の低いサーバ7，8のレスポンスタイムが増大する．これは，グループ化によりサーバ7，8に以前より大きなTTLが設定されたためと考えられる．これに対して，図13〜15のCase 8〜10の場合においては，グループ化を行うことにより，中間の処理能力を持っているサーバ5，6のTTLが小さくなって，これらのサーバのレスポンスタイムが減少し，効果的な負荷の分散が実現できていることが分かる．

以上の結果から，サーバの処理能力が3種類以上あり，処理能力の最大値と最小値の差が大きい場合は処理能力の低いサーバをグループ化して扱った方がよい場合がある．そのとき，このグループに属する各サーバの処理能力はグループ内での平均値とすればよい．グループ化すべき処理能力cの値は，事前にシミュレーション等で把握しておく必要があるが，目安としては，図5〜10と図12〜15の結果の比較から，サーバの処理能力が2種類の場合でパラメータαを大きくしなければならない場合の処理能力の値と思われる．

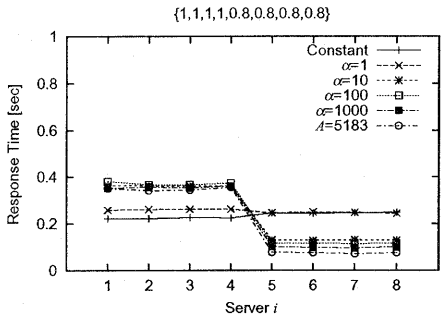

図5　各Webサーバのレスポンスタイム (Case1)

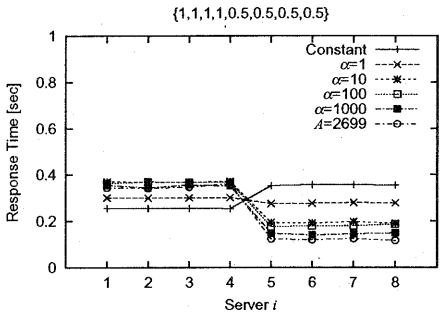

図6　各Webサーバのレスポンスタイム (Case2)

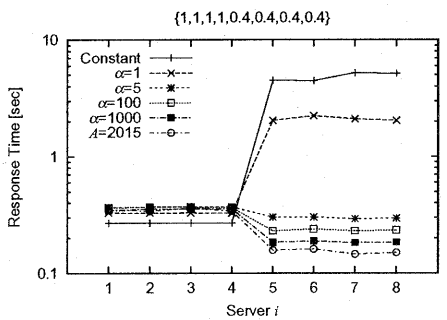

図7　各Webサーバのレスポンスタイム (Case3)

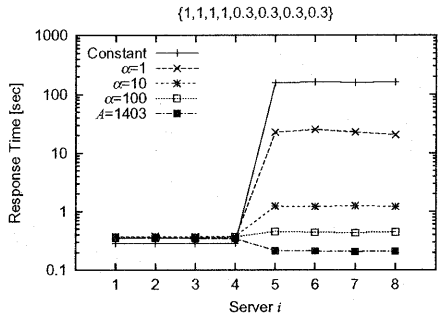

図 8　各 Web サーバのレスポンスタイム (Case4)

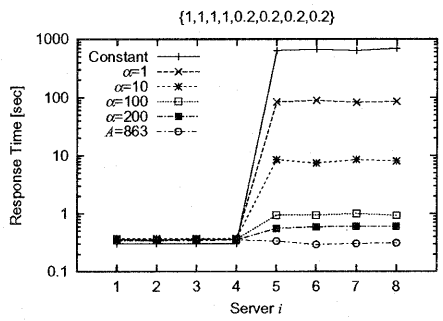

図 9　各 Web サーバのレスポンスタイム (Case5)

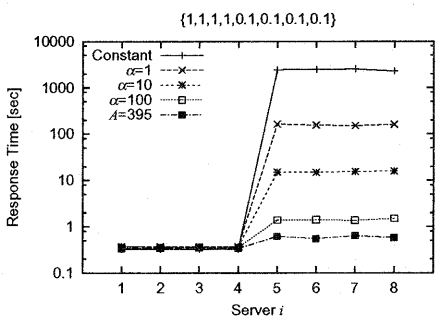

図 10　各 Web サーバのレスポンスタイム (Case6)

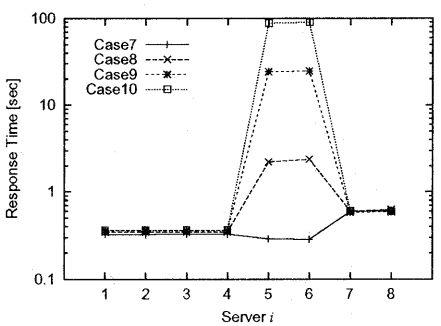

図 11　Case7 から Case10 の各 Web サーバのレスポンスタイム

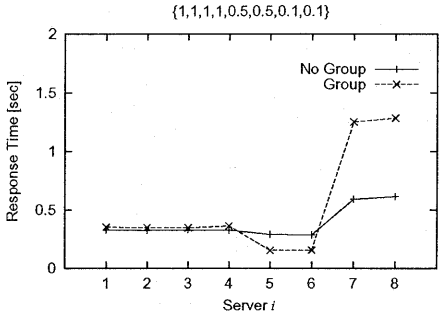

図 12　各 Web サーバのレスポンスタイム (Case7)

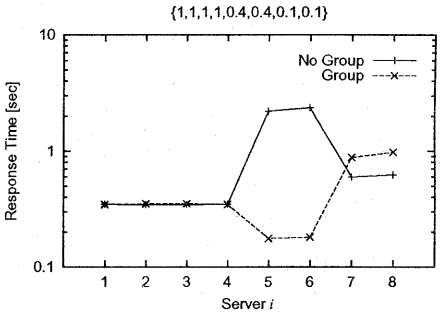

図 13　各 Web サーバのレスポンスタイム (Case8)

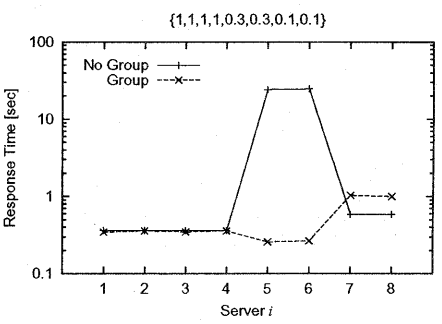

図 14　各 Web サーバのレスポンスタイム (Case9)

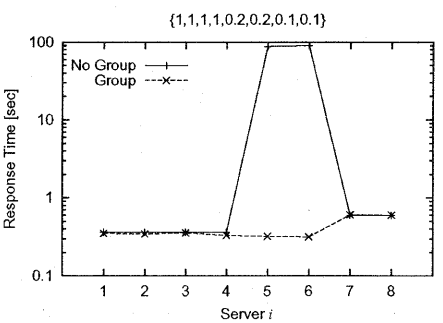

図 15　各 Web サーバのレスポンスタイム (Case10)

6. おわりに

処理能力の異なる複数台のWebサーバに対して，DNSラウンドロビンにより負荷分散を行う場合では，各Webサーバの処理能力の差が大きい場合において，処理能力の低いWebサーバのレスポンスタイムが大きくなる．

そこで，本研究では，この問題を解決するため，サイト側のDNSサーバに設定するTTLの調整法について考察した．シミュレーションによる評価を行った結果，TTLを調整することで各Webサーバの処理能力の差が大きい場合においても，レスポンスタイムを均等にすることができた．

今後の課題として，今回省略したネットワーク遅延やアドレス解決に要する時間等を考慮した場合の有効性を調査する必要があると考えられる．

文　献

[1] P. Albitz, C. Liu 著，小舘 光正 訳，高田 広章，小島 育男 監訳，"DNS & BIND 第4版"，株式会社オイラリー・ジャパン，2002.

[2] G. K. Zipf, "Human Behavior and the Principles of Least Effort", Addison-Wesley, 1949.

[3] M. Colajanni, P. S. Yu and D. M. Dias, "Scheduling algorithms for distributed Web servers", *Proceedings of ICDCS '97*, pp. 169–176, 1997.

[4] V. Cardellini, M. Colajanni, P. S. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-server Systems", *Proceedings of ICDCS '99*, pp. 528–535, 1999.

[5] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable Web server clustering technologies", *IEEE Network*, May/June 2000, pp. 38–45, 2000.

[6] T. Bourke, Server Load Balancing, O'Reilly, 2001.

[7] 松本 敏明，滝沢 泰盛，"振り分ける"，日経コミュニケーション，No.328，pp. 120–123，2000.10.16.

[8] A. Iyengar, J. Challenger, D. Dias and P. Dantzig, "High-performance Web site design Techniques", *IEEE Internet Computing*, March/April 2000, pp. 17–26, 2000.

[9] R. S. Engelschall, "Webサイトの負荷分散テクニック", UNIX Magazine, 1998.9, pp. 54–62, 1998.