WIDE Technical-Report in 2006

2-stepアルゴリズム: ネットワーク変動に対するサーバ選択アルゴリズムの安定性について

wide-tr-mawi-2step-algorithm-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on this document, please contact to ad@wide.ad.jp

2-step アルゴリズム: ネットワーク変動に対するサーバ選択アルゴリズムの安定性について

宮地利幸* 長健二朗† 篠田陽一‡

概 要

現在サーバ-クライアントモデルの通信ではベストサーバセレクションが広く利用されている。しかし、ベストサーバセレクションは効率は良いがネットワーク変動に非常に弱い。また既存のレシプロカルアルゴリズムはネットワーク変動に強いが効率が悪い。

我々は、既存のアルゴリズムの性質を検証するためにシミュレーションを行った。また、問題を視覚的に認識するためネットワーク変動がおこった際のサーバ負荷の変化を可視化した。

この結果、レシプロカルアルゴリズムは、ある程度の確率でコストの大きなサーバを選択するため、性能が低下することがわかった。そこで、我々は、性能の良い少数のサーバから構成されるワーキングセットを選択し、そのなかから、サーバを一定の確率で選択する 2 ステップの方式を提案する。この方式がサーバのコスト変動への適応性、負荷分散、スケーラビリティ、効率の面において非常に優秀であることをシミュレーションにより示した。

1 背景

現在サーバクライアント型サービスが広く利用されており、この型のサービスでは、クライアントはサーバを何らかの方法で決定し、選択されたサーバへリクエストを送る。

サーバ選択アルゴリズムとして、ベストサーバセレクションが広く利用されているが、サーバの負荷が偏りがちである。サーバの負荷の偏り自体は効率を考えると仕方がなく、サーバ配置の観点からは負荷の高いサーバがはっきりわかることでサーバの増強や追加を行うことで負荷を分散できる可能性がある。

しかし、負荷の偏りがあった場合にネットワーク変動が起きるとクライアントが一斉に移動し、クライアントが再選択するサーバも集中した場合、さらなるネットワーク変動の原因となる可能性がある。

*toshi-m@jaist.ac.jp
†kjc@iijlab.net
‡shinoda@jaist.ac.jp

ベストサーバでは上記の問題が顕著にあらわれ、他の手法はサーバの負荷をある程度分散できるが効率面に問題がある。

本論文では、ネットワーク変動に強く、効率が良いサーバ選択アルゴリズムを提案する。

2　従来のサーバ選択アルゴリズム

本章では、従来から利用されている一般的なアルゴリズムについて説明、考察する。

2.1　ベストサーバセレクション

ベストサーバセレクションはホップ数や RTT などのコストから、利用できるサーバ中で最適なものを選択する方式である。

クライアントは最も効率の良いサーバを一意に選択するため効率は常に最良である。

しかし、問題点としてネットワーク変動の増幅や、振動の原因となることが上げられる。

2.2　ユニフォームセレクション

ユニフォームセレクションはコストに関わらず無作為に利用可能なサーバからサーバを選択する方式である。

この方式では、負荷の集中がおこらず、それに起因する問題もおこらない。

しかし、あるクライアントからのサーバ性能を考慮しないため性能は悪い。

2.3　レシプロカルセレクション

サーバ性能のコストの逆数に比例する確率でサーバを選択する方式である。サーバを選択する確率をきめる関数によりその挙動は変化する。

性能の低いサーバもある程度の確率で選択されるため、サーバの集中をある程度に押さえられ変動にも強い。

一方、コストが悪いサーバも選択されることから性能もベストサーバに比べて良くない。

2.4　実例：DNS の場合

DNS[1][2] は比較的少ないサーバ群から実際に問い合わせをするサーバを選択する。

DNS の実装として広く利用されている、BIND[3] バージョン 8 および 9 で利用されているアルゴリズムははレシプロカルセレクションの一種と考えられる。また、DJBDNS[4], Microsoft Windows Internet Server はユニフォームセレクションを採用している。

BIND で利用されている実践的手法は、多くの場合において効率を向上させるが、検討の余地がある。クライアント群からみてコストの差が小さな 2 台のサーバが存在する場合に、サーバ負荷が大きく片寄る可能性があることや、多くのサーバを利用するサービスには適当でないからである。これについては [5] で述べられている。

3　既存の手法のシミュレーションによる評価

我々は、既存の手法の評価を行うため、ある程度の規模のネットワークでコスト変動が起こるシミュレーションを行なった。

3.1　シミュレーショントポロジおよびシミュレーション内容

サーバ負荷に偏りがあるトポロジを機械的に生成するために、スケールフリーなネットワーク構造を元にしたトポロジ生成を行なった。トポロジ構築のためのルールを以下に示す。

- 1 台目のノードを設置
- 2 台目のノード以降は既存のノードを 1 台選択し、そのノードに接続するように設置する。2 台目以降のノードは生成時に既存のノードに接続されるが、このノードは既存のノードからのエッジの数に比例した確率で選ばれる。エッジを多く持つノードはより高い確率でさらに多くのエッジを得る。
- ノードを 10 台設置するごとにサーバを設置する。
- サーバはその時点でエッジを最も多く持つノードにのみ接続される。選択したノードにすでにサーバが接続されていた場合は、次にエッジが多いノードに接続する
- サーバのクライアント数が 20 を超えるとノードを 2 つに分割し、サーバを新たなノードにサーバを新たに接続する。ノードに接続されていたエッジの半数は新たなノードに接続しなおされる。
- ノードを 100 台設置するごとにその時点でのエッジの数に比例して、2 台のノードを選択し、その 2 台を接続するエッジを生成する。すでにエッジで接続されていた場合は、改めて選択しなおす。

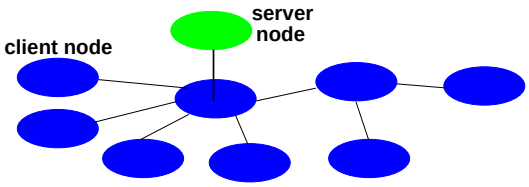

図 1: Concept of the simulation topology

- ただしトポロジ生成時はベストサーバセレクションによりクライアント数を算出することとする

図 1 にサーバを設置する際の概念を示す。

ある程度大規模なネットワークを構築するため 500 ノードを設置した。サーバの分割などにより最終的にノード数は 510、サーバは 60 台となった。

サーバコストはエッジに重みをつけることにより表現した。エッジコストの初期値は 10 とする。

このトポロジ上で各アルゴリズムのシミュレーションを行った。シミュレーションは、サーバと接続されたノード間のコストを変更し、すべてのクライアントから各アルゴリズムを用いてサーバを選択し、各種データを取得する。1 ステップごとに一台のサーバをランダムに選択し、このサーバと接続するノード間のコストを $1 \leq c \leq 40$ の値に変化させる。この時 c の値はランダムで決定する。各クライアントは各ステップ毎 100 回サーバセレクションを実行し、サーバをクライアントが選んだ回数をサーバの負荷とする。以上の規則でコストを変化させ次のステップで元に戻す。このステップの組みを 50 回、合計 100 ステップ行った。

3.2 視覚化

サーバ負荷を視覚でとらえることにより各アルゴリズムでの負荷集中の度合いをわかりやすくし、コストが変動した場合のサーバ負荷の移動や新たなサーバの設置によるサーバ負荷の変化を認識しやすくする。

トポロジの描画には Tulip[7] を利用した。Tulip はグラフを視覚化するためのツールであり、ノードの表示位置決定および表示を行う。

Tulip を用いて視覚化したトポロジを図 2 に示す。図中の青いノードはクライアントを示し、それ以外の色のノードはサーバが設置されたクライアントであり、その色でサーバ負荷を表している。緑色のサーバの負荷は 600 未満、黄色のサーバ負荷は 600-1099、オレンジのサーバ負荷は 1100-1599、そして赤色は 1600 以上の負荷をもつのサーバである。

サーバ負荷の違いを色の変化で示したことで、サーバ負荷の移動が目で捕らえられるようになった。

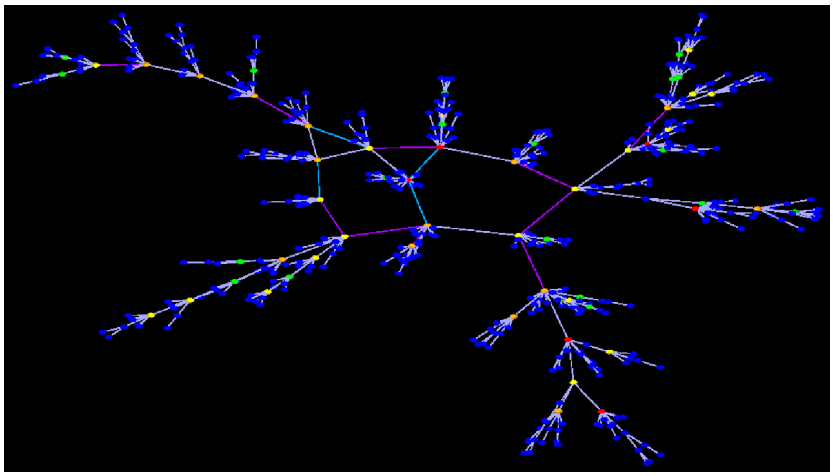

図 2: The simulation topology

3.3　ベストサーバセレクション

図 3 にベストサーバセレクション利用時の各クライアントから選択されたサーバまでのコストの平均と最大値を示す。また、図 4 にステップ毎の各サーバに接続しているクライアント数を示す。

図 3 から、平均コストは 22 程度となり、ノード間のコストが 10 で固定であることを考えると非常に効率がいいことがわかる。しかし、図 4 から、負荷が集中しているサーバ周辺のコストが変動した際に多くのクライアントが別の一台のサーバに移動していることがわかる。

新たにサーバを追加してももっとも性能がよいサーバへ不可が集中するため、サーバの新規追加では負荷分散が困難である。

これらの結果は我々の予想および、実インターネット上で観測されている状況と同一である。

3.4　ユニフォームセレクション

図 5 にユニフォームセレクション利用時の各クライアントから選択されたサーバまでのコストの平均と最大値を、図 6 にステップ毎の各サーバに接続しているクライアント数を示す。

ユニフォームセレクションではコスト平均が 77 程度とベストサーバセレクションの約 3.5 倍の値を示しているが、サーバの負荷は非常に平均的であり、コストの変動が起きても、クライアントの移動はほとんどみられない。性能は期待できないが、サーバ負荷の変動が小さいという事前の考察を裏付ける結果が得られたといえる。

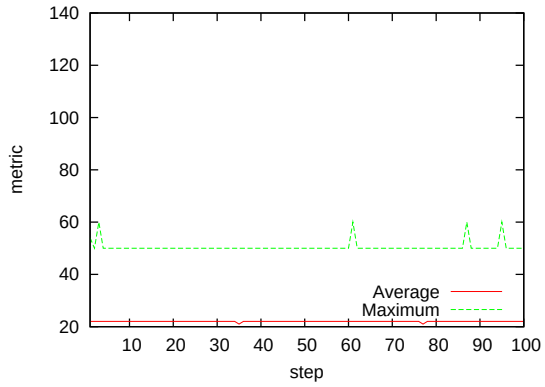

図 3: Average and maximum costs of best-server algorithm

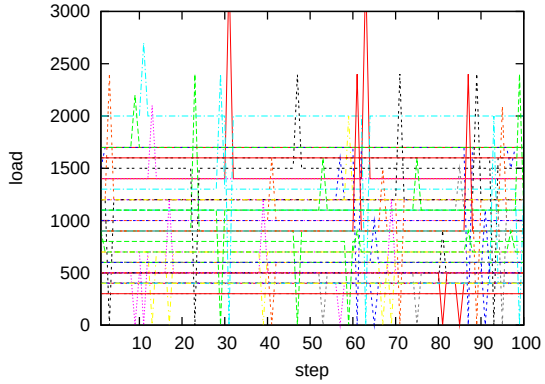

図 4: Server load of best-server algorithm

3.5 レシプロカルセレクション

レシプロカルセレクションでは、利用するアルゴリズムを決定する必要があり、我々はコストを c として $1/c$ と $1/c^2$ の関数を利用した。

3.5.1 関数 $1/c$ を利用したレシプロカルセレクション

関数 $1/c$ を利用したときの、最大値と平均値を図 7 に、各ステップ毎のサーバに接続するクライアント数を図 8 に示す。平均値は 67.5 程度であり、ベストサーバセレクションの約 3 倍、ユニフォームセレクションの 14%程度の性能向上しか望めない。一方サーバ変動に関してはユニフォームサーバセレクションよりは、大きいがベストサーバセレクションと比べれば十分小さいといえる。

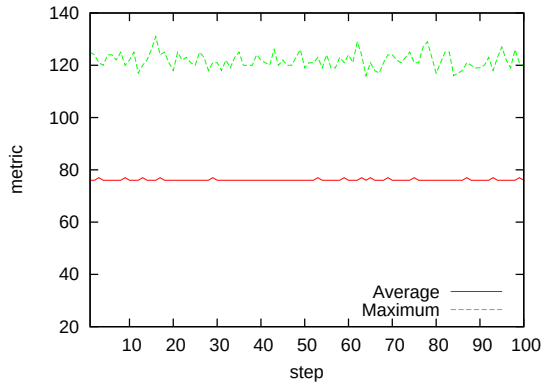

図 5: Average and maximum costs of uniform algorithm

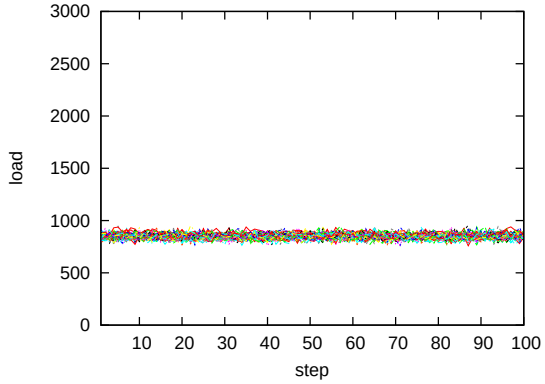

図 6: Server load of uniform algorithm

3.5.2 関数 $1/c^2$ を利用したレシプロカルセレクション

関数 $1/c^2$ を利用したときの、最大値と平均値を図 9 に、各ステップ毎のサーバに接続するクライアント数を図 10 に示す。

コストがより小さいサーバを選択する確率が 2 次関数的に向上するため、$1/c$ を利用した場合よりも、よりベストサーバセレクションに近い挙動を示す。しかし平均値は 55.5 程度とベストサーバセレクションの値の 2 倍のコストがかかっている。一方サーバの負荷の集中については、あるサーバのコストが大きく変化しても、負荷は複数のサーバに分散し吸収されるため、別のサーバに大きな影響を与えることはない。

3.6 各手法の比較と考察

これまでの考察およびシミュレーションで確認できた各サーバセレクション手法の特性を図 11 に示す。ベストサーバセレクションを利用した場合は、

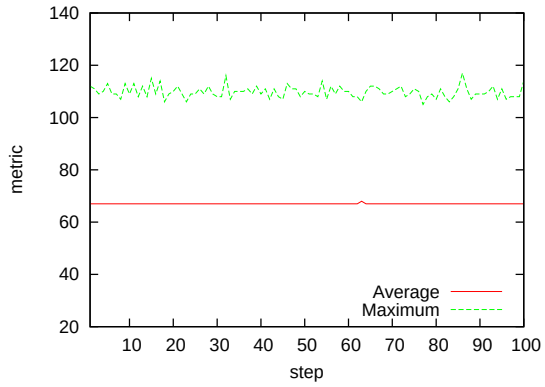

図 7: Average and maximum costs of reciprocal algorithm($1/cost$)

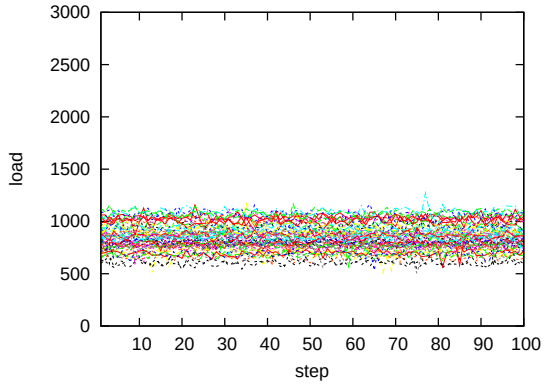

図 8: Server load of reciprocal algorithm($1/cost$)

各クライアントが最もコストが小さいサーバを一意に決定するため、サービスの提供を受けられるまでの時間が非常に小さい。しかし、負荷が集中しているサーバ周辺でネットワークの変動があると、その負荷を別の一台または少数のサーバで吸収することになるため、別のサーバへの影響が非常に大きいといえる。また、クライアント群からみた最良のサーバが一意に決まるため、サーバ追加で思うような負荷分散をすることが難しい。

ユニフォームセレクションを利用した場合は、サービスの提供を受けられるまでの時間は非常に長いが、ネットワーク変動が起きても、影響があるサーバの数は非常に少なくてすむ。ユニフォームセレクションを利用している場合は、サーバをクライアントが集中している場所に置いても性能向上ができない

レシプロカルセレクションは上記 2 手法の中間に位置する手法であるが、存在するすべてのサーバを利用するため、ある程度の確率でコストの大きなサーバを選択することになり、ベストサーバセレクションと比べ効率は低い。

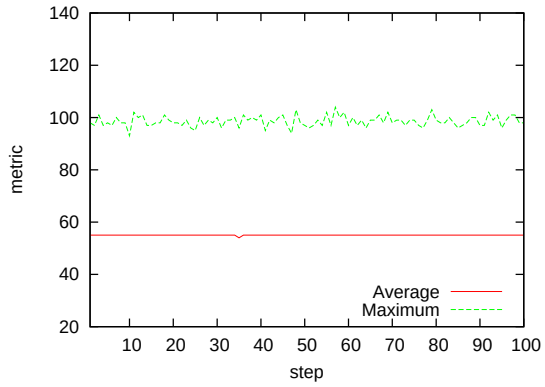

図 9: Average and maximum costs of reciprocal algorithm($1/cost^2$)

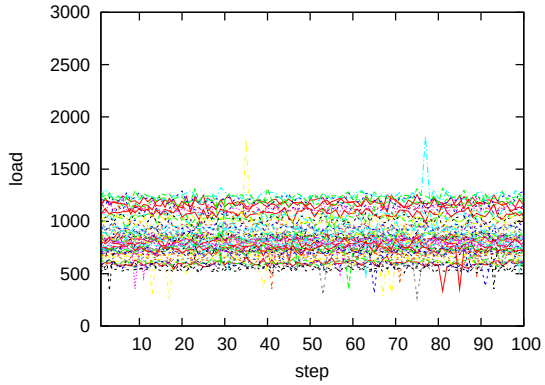

図 10: Server load of reciprocal algorithm($1/cost^2$)

これらの手法の考察から、一般的にネットワーク変動を伝搬させないためには、複数のサーバから確率的に利用するサーバを選択すれば良いが、従来の手法では十分な性能が期待できないということが言える。

次章ではこの問題について考察し、あらたな選択手法を提案する

4　2 ステップサーバセレクション

ネットワーク変動が起きてもほかのサーバに大きな影響をおよぼさないためには各クライアントは利用するサーバを一意に決定するのではなく複数のサーバの中から一台のサーバを選択をすれば十分であると考えられる。しかし、これまでの手法では性能面でベストサーバセレクションを大きく下回る。

これまでのサーバセレクションの問題点は、1 つのアルゴリズムで、性能の向上と負荷分散の性能向上を目指していたことにある。そこで我々は、性能の向上と負荷分散を行うアルゴリズムを分離した、2 段階でのサーバセレクションを提案する。

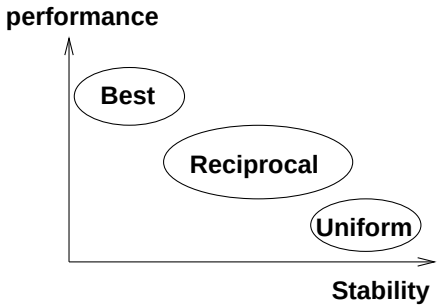

図 11: 各サーバセレクション手法の特性

第一段階では、コストがあまりにも大きいサーバを選択しないために、まずある程度のコストで利用できる W 台で構成されるサーバセットを選択する。第二段階では、第一段階で構成した W 台のサーバセットで負荷分散を行い、実際に利用するサーバを選択する。これによりコストがある程度小さいサーバのみを分散して利用することができる。

4.1 ワーキングセットの選択

ワーキングセットの選択は、少数の性能の良い少数のサーバに利用するサーバを絞ることで性能の向上のために行う。

柔軟にサーバメトリックの変動に対応するためには短い間隔でメトリックのチェックを行う必要がある。しかし、利用できるすべてのサーバのコストをある程度の間隔で検査する必要があるため、非常に大きな負荷となる。実際に利用されるサーバはサーバセットに含まれるものと、そのあとに続くコストが小さなサーバ群であるため、コストが小さなサーバほどチェック間隔を短くすることでそのコストを低減する。

存在するサーバの数を N とすると、最も性能の良いサーバのランクは 1 となり、最も性能の悪いサーバのランクは N となる。ここでランクを i とすると、サーバ i の確認間隔 $q(i)$ は以下の式で表される。

$q(i) = \frac{C}{i}$ ここで C を変更することで耐規模性が変化する。サーバ N 台分のコストの合計確認回数は $Q(N)$ は以下の式で表される。

$Q(N) = \sum_{i=1}^{N} q(i) = C \sum_{i=1}^{N} \frac{1}{i}$

N が大きくなるほど曲線は水平に近くなるため、N が大きくなっても十分な耐規模性をもつといえる。また、ワーキングセットに含まれるサーバのコストチェックがもっとも回数が多いが、実際にリクエストを出す際にコストを確認すればさらに負荷を軽減できる。

4.2 ワーキングセットからのサーバ選択

ワーキングセットからサーバを確率的に選択することで、サーバ負荷を分散する。これによりネットワーク変動があったときの影響を小さくできる。我々のアルゴリズムでは、サーバセットからのサーバ選択にレシプロカルサーバセレクションを採用した。

4.3 シミュレーション結果

前述のトポロジ上で我々の提案する手法の性能評価を行った。本シミュレーションでは、W の値は 4 とした。本方式では、サーバセットに含まれていたサーバに問題がおこれば次点のサーバと入れ替えがおこるため、サーバセットは小さくて良いためである。シミュレーションでのサーバコストには、round-trip time(rtt) を利用した。サーバの処理時間やネットワークのディレイを含み、実際にクライアントがサービスをうける時間であるためである。また、一時的な変動の影響を排除するため、smoothed rtt(srtt) を利用した。srtt は以下の式で計算される。

$srtt = \alpha \times srtt + (1 - \alpha) \times rtt$

α には一般的に 0.7-0.8 が利用されるが迅速にコストの変化に適応するため今回は 0.6 という値を用いた。

ワーキングセット中からのサーバ選択のためのレシプロカルアルゴリズムには、サーバの負荷が偏り過ぎないよう 1/c のアルゴリズムを採用した。

ベストサーバセレクションと提案手法の最大値と平均値を図 12 に、各ステップ毎のサーバに接続するクライアント数を図 13 に示す。図 12 より、平均値は 30 程度であり、ベストサーバセレクションと比較して、性能低下を 36%ほどに押さえられていることがわかる。また、図 13 から、サーバへの負荷集中はある程度見られるものの、ある程度の性能を求めると負荷の集中は避けられない。負荷が集中しているサーバ周辺のコストが変動した際に、その負荷がごく少数のサーバに移動することによる影響を避けることの方が重要である。ただし、本手法の負荷の偏りは、ベストサーバセレクション利用時と比べて小さい。

図 14 は 575 ステップ前後で、あるクライアントから見た性能が高い、7 台の負荷をプロットしたものである。532 ステップ目でサーバ 5 の負荷が激減しているが、この時サーバ 5 と接続されているノード間の *srtt* が 10 から 38 へ大幅に増加している。これにより、サーバ 5 に接続していたクライアント群は別のサーバへ移動しているが、一台のサーバへ移動するのではなく複数のサーバへ移動しているため、サーバ 5 以外のサーバの負荷はサーバ 5 の負荷ほど大きく変化しない。

また 578 ステップ目では、サーバ 50 のコストが 10 から 3 へ変化し、サーバセット外であったサーバ 50 がサーバセット中に移動し、負荷が急激に上昇

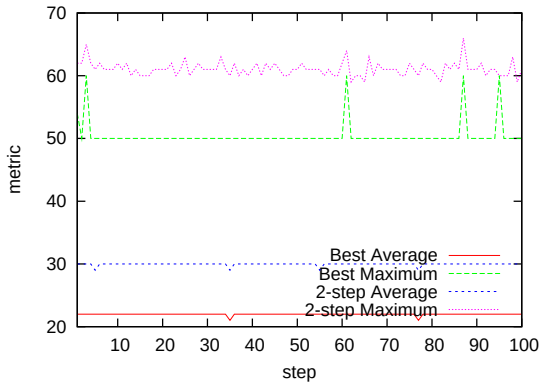

図 12: The maximum and average values of the 2-step algorithm compared with the best-server algorithm

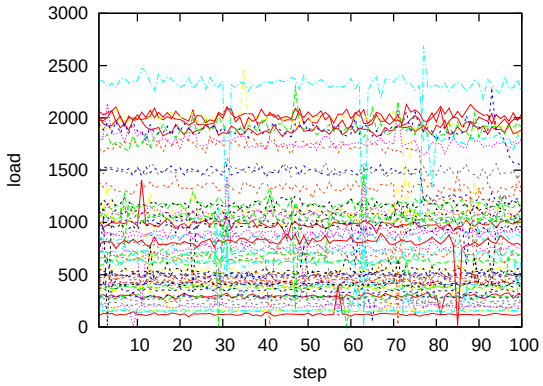

図 13: The server load of the 2-step algorithm

しているが、この負荷減少分も複数のサーバに分散しているのがわかる。

5　考察

今回のシミュレーションの妥当性をシミュレーションで利用したトポロジとサーバまでのコスト変動について考察する。

5.1　トポロジ

トポロジは 3.1 章に記述した手順に従い自動的に生成した。この生成手順で構築したトポロジには最終的に、510 台のノードと 60 台のサーバから構成されている。

構築されたトポロジは、多くのノードに接続されたノードがリングをつくり、その他のノード群がそこに接続されている。サーバはリング周辺に密に

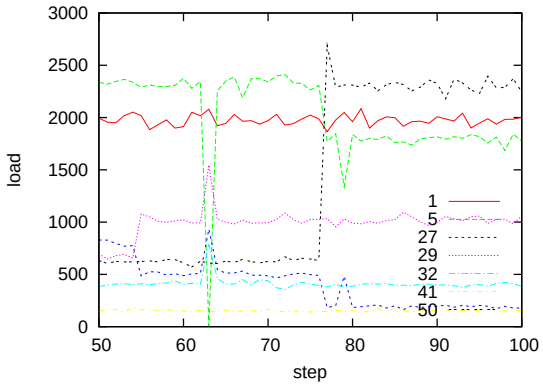

図 14: The loads of 7 servers

分布しており、実際のトポロジに近いといえる。

5.2 サーバコスト変動

シミュレーション部分は、初期状態から様々なサーバのコストを変化させ、その影響を確認するものである。

本シュミレーションでは、サーバは常にクライアントから要求された処理を行えるだけの性能を持っており、あるサーバを利用するクライアントの数が増えても、負荷に影響をおよぼさない点で実環境とは異なるといえる。実環境ではサーバの負荷が増大するにしたがって性能が低下する場合、性能低下は *rtt* が大きくなるように見える。あるサーバの *rtt* が増大するモデルは今回のシミュレーションそのものであり、シミュレーションの結果からこのような状態が起きた際の各アルゴリズムの傾向は予想できる。

6 まとめ

現在広く利用されているベストサーバセレクションでは、負荷が集中しているサーバ周辺のネットワークに変動がおこった際に、クライアントが別の少数のサーバに再接続することにより、更なるネットワーク変動の原因となる可能性となる問題点を指摘した。また、既存の手法のサーバ負荷の変化を視覚的に把握するためにサーバ負荷を色の変化で示すようなトポロジを表示するシミュレーションを行った。その上で、従来のサーバセレクションアルゴリズムについて考察を行い、ベストサーバセレクションに近い性能を持ちつつ、負荷が集中するサーバ周辺のネットワーク変動が起きた場合でも、別の複数のサーバによりその影響を吸収できる 2 ステップアルゴリズムを提案した。

サーバの負荷の偏りを小さくすれば、サーバリソースを適切に利用でき、ネットワーク変動にも強くなるが、性能が低下する。ある程度の負荷の偏りは許容し、ネットワークが変動した際に、負荷が集中していたサーバの負荷を以下に分散して吸収するかが重要である。また、負荷の偏りは新たなサーバの追加や、サーバの再設置などで解消できる。

2 ステップアルゴリズムでは、サーバの負荷の偏りはある程度あるが、ネットワーク変動が起きたときにも、影響があったサーバの負荷をある程度分散して吸収できる。また、ベストサーバセレクションと異なり、各クライアントが利用するサーバを一意に決定しないため、ある程度の負荷分散効果も期待できる。

2 ステップアルゴリズムでは、サーバ負荷の偏りはある程度認められるが、各クライアントがサーバを一意に決定しないため、ある程度の負荷分散でき、ネットワーク変動が起きたときでも、影響があったサーバの負荷を分散吸収することができる。つまりサーバのコスト変動への適応性、負荷分散、スケーラビリティ、効率を実現した。

今後はネットワークトポロジ、サービスとサーバセレクションの一般的な関係について議論する。また、クライアントによるサーバセレクションとサーバプレイスメント問題には関連性があるはずなのでそれについて検証する。

参考文献

[1] P.Mockapetris.: Domain names - concepts and facilities. RFC1034, IETF, November 1987.

[2] P.Mockapetris.: Domain names - implementation and specification. RFC1035, IETF, November 1987.

[3] ISC BIND, http://www.isc.org/

[4] DJBDNS, http://www.djbdns.org/

[5] Ryuji Somegawa, Kenjiro Cho, Yuji Sekiya and Suguru Yamaguchi.: The Effects of Server Placement and Server Selection for Internet Services. IEICE Trans. on Commun. Vol.E86-B No.2. February 2003. p.542–551.

[6] A.-L. Barabási, R. Albert, and H. Jeong,: Mean-field theory for scale-free random networks Physica, 272, 173-187 (1999).

[7] Tulip Software, http://www.tulip-software.org/