Flash Crowdによる影響を軽減する複製サーバの配置先決定手法に関する研究

浅原 理人

慶應義塾大学大学院
理工学研究科開放環境科学専攻

博士 (工学) の学位請求論文

2010年度

Flash Crowd による影響を軽減する
複製サーバの配置先決定手法に関する研究

浅原 理人

論文要旨

インターネットの利用者数は年々増加傾向にあり，コンテンツの配信にかかる負荷が増加し続けている．それに伴って，安定したサービスの提供を阻害する *Flash Crowd* と呼ばれる現象がインターネット上で発生するようになった．Flash Crowd が発生すると，インターネットサーバのアクセス数が数分のうちに通常時の数倍から数十倍に達し，サーバは過負荷状態になる．Flash Crowd の発生要因は多岐に渡るため，Flash Crowd の発生を予測することは困難である．

大量のクライアントアクセスを処理する一般的な方法として，**複製サーバ**の配置による負荷分散が知られている．しかし，Flash Crowd 対策用に複製サーバを常時稼働させておくためには，Flash Crowd の規模をあらかじめ予測しておかなければならない．また，Flash Crowd の生じていない平常時には余剰サーバが多くなり，サーバの運用コストが増大するという問題がある．

本論文では，Flash Crowd による影響を軽減する複製サーバの配置先決定手法 ExaPeer Server Reposition (EPSR) を提案する．EPSR はあらかじめ世界中に設置された共有ホスティングマシン群で動作する．EPSR はこのホスティングマシン群に対して，サービスの需要に応じて複製サーバの配置先を動的に決定する機能を提供する．これにより，Flash Crowd の規模の予測を必要とせずに Flash Crowd への対策が行える．また，需要変動に応じた動的な複製サーバの配置ができるため，余剰サーバの運用コストが生じない．

Flash Crowd の影響を軽減するように複製サーバの配置先を決定するには，次の 3 つの課題がある．第一に，Flash Crowd によるアクセス増加がピークに達するまでの数分以内に配置先を決定する必要がある．第二に，通信負荷の分散とサービス遅延の低減のために，配置先とクライアントとの通信遅延が小さくなるように配置先を決定する必要がある．第三に，複製サーバによってホスティングマシンが過負荷にならないように，複製サーバの配置先を決定する必要がある．既存手法の多くは，Globule のような，全サーバのアクセスログといった収集に時間のか

かる大域情報に依存するものや，FCAN や CoralCDN のような，通信遅延やマシンの負荷および性能を考慮せずに配置先を決定するものである．そのため，Flash Crowd 対策に適用するには限界がある．

EPSR は複製サーバの配置に関する 3 つの課題の全てを解決する．EPSR は大域情報を必要としない Peer-to-Peer 技術を用いることで急激な需要の増加に対応する．個々のマシンは自律的に需要の変動を検出し，必要に応じて複製サーバの配置先候補となる．個々のマシンでの需要変動の検出と需要の発生地域の特定を実現するために，EPSR は物理ネットワークのトポロジを考慮したオーバレイネットワークを，ネットワーク座標系と分散ハッシュテーブルを組み合わせて構築する．個々のマシンは，オーバレイ上で隣接するマシンとのみメッセージを交換することでクライアントのアクセス経路を分析し，サービスへの需要が増加もしくは減少している地域を特定する．マシンはアクセス経路の分析から複製サーバの実行による効果を見積もり，配置先の候補となるべきか判定する．この際，個々のマシンは複製サーバの負荷とマシンの許容負荷を元に，マシンが過負荷にならないように隣接するマシンから追加の配置先候補を独立して選択する．

シミュレーションを行い，EPSR が Flash Crowd に対して有効であることを確認した．シミュレーションでは，約 3000 台のホスティングマシンに対して EPSR が 25 秒で Flash Crowd の検出と発生地域の特定を行い，配置先の選択を開始することを確認した．また複製サーバの配置先数が 300 秒で安定することを確認した．その際，EPSR は他方式と比較してよりクライアントとの通信遅延が小さくなるようなホスティングマシンを配置先として選出した．またシミュレーション結果は，EPSR が他方式よりも少ない複製サーバ数で，どのホスティングマシンも過負荷にならないように配置先が決定できることを示した．

A Study on the Mechanism for Finding Replica Server Locations for Alleviating Flash Crowds

Masato Asahara

Abstract

As the Internet population is increasing yearly, the burden of content distribution on the Internet has been augmenting. As the Internet is becoming the widely used media, *flash crowd* often occurs on the Internet and prevents an Internet server from serving a reliable service. With flash crowds, the load of an Internet server reaches to dozens of times the load of a typical day in a few minutes, and then the server becomes overloaded. It is quite difficult to predict flash crowds since there are various factors of flash crowds.

Positioning *replica servers* is well known as a popular load balancing strategy to provide a reliable service for a massive number of clients. Unfortunately, positioning over-provisioned replica servers to alleviate flash crowds has two drawbacks; one is the necessity for predicting the scale of flash crowds and the other is the expense of maintaining the over-provisioned replica servers.

This dissertation proposes a novel mechanism called *ExaPeer Server Reposition* (EPSR) for finding replica server locations for alleviating flash crowds. EPSR runs on a pool of hundreds or thousands of shared hosting machines all over the world. EPSR provides the hosting machines with the function that determines the number and the location of replica servers on the basis of demand fluctuations. EPSR enables service providers to provide their services in flash crowds without predicting the scale of flash crowds. Also, EPSR needs no over-provisioned replica servers because it enables to dynamically reposition replica servers in short-term demand fluctuations as well as long-term ones.

Finding replica server locations has three challenges of alleviating flash crowds. First, we must find replica server locations before flash crowd reaches its peak, e.g., in a few minutes. Second, a machine close to clients on the network, i.e., the network latency between the machine and the clients is short, should be a candidate for running a replica server to distribute network load and suppress service delay. Finally, a machine with

enough capacity to run a replica server must be a candidate to prevent a machine from becoming overloaded. Most of existing mechanisms have the limitation of applying to finding replica server locations for alleviating flash crowds. This is because some of them like Globule rely on collecting global information that requires a long period of time such as access logs of all the servers or some of them like FCAN and CoralCDN determine replica server locations without considering the network latency, the capacity of a machine or the load of a machine.

EPSR addresses the three challenges. EPSR takes a P2P-based approach to quickly respond to rapidly increasing demands; it does not rely on global information. Each machine autonomously detects demand fluctuations and becomes a candidate if necessary. To enable a machine to detect demand fluctuations and locate a high-demand area, EPSR builds a topology-aware overlay network by combining network coordinate systems and distributed hash tables (DHTs). Each machine analyzes access paths from clients by exchanging messages with only adjacent machines on the overlay and locates the areas in which the demand for a service is increasing or decreasing. It estimates the benefit of running a replica server from this access path analysis. Based on the load of a replica server and machine's capacity, each machine independently chooses more candidates from adjacent machines so that the machine does not become overloaded.

Simulation results demonstrate that EPSR is suitable for alleviating flash crowds. In the simulation with 3,000 over hosting machines, EPSR detected demand increase, located high-demand areas and began selecting candidate machines for replica servers within 25 seconds. And then, it correctly estimated the number of required replica servers and the number of candidate machines had been stabilized within 300 seconds. With EPSR, candidate machines were closer to clients than that with other methods. The results also demonstrate that EPSR selected candidate machines as no machines became overloaded even with fewer candidates than that with other methods.

目 次

第 1 章　序論 **1**

1.1　本研究の背景と動機 1

1.2　ExaPeer 3

1.2.1　ExaPeer の概要 3

1.2.2　ExaPeer 実現の課題 5

1.3　本研究の目的 6

1.4　本研究の課題 7

1.5　本研究の提案 10

1.5.1　提案手法の概要 10

1.5.2　評価方法 11

1.6　本研究の貢献 12

1.7　本論文の構成 13

第 2 章　関連研究 **15**

2.1　固定配信網方式 15

2.2　オフライン計算方式 18

2.3　オンデマンド負荷軽減網方式 21

2.4　通信遅延考慮型配信網方式 25

2.5　まとめ 29

第 3 章　ExaPeer Server Reposition **30**

3.1　設計上の要点 30

3.2　概要 32

3.3　ネットワーク座標 33

3.4　EPSR オーバレイネットワーク 36

3.5　需要発生地域の特定 38

3.5.1　基本メカニズムの改良 39

3.6 複製サーバの配置先の決定 43
3.6.1 基本メカニズムの改良 48
3.7 まとめ 52

第 4 章 実験 **55**
4.1 実験方法 55
4.2 EPSR による複製サーバの配置先決定の評価 57
4.2.1 複製サーバの配置先数の推移と決定に要した時間 59
4.2.2 クライアントと複製サーバの配置先間の通信遅延 60
4.2.3 性能制約に対するホスティングマシンの負荷 62
4.3 EPSR 各部の評価 63
4.3.1 需要発生地域特定部の評価 63
4.3.2 ネットワーク座標系，VPE および VDR の効果 79
4.3.3 Dummy Query および Dummy Connection の効果 82
4.4 まとめ 85

第 5 章 議論 **86**
5.1 EPSR の適用範囲 86
5.2 EPSR のパラメータの決定方針 88
5.3 スケーラビリティ 89
5.3.1 ホスティングマシン数に対するスケーラビリティ 89
5.3.2 サービス数に対するスケーラビリティ 93
5.4 サービス間での公平性 93
5.5 セキュリティ 93
5.6 まとめ 94

第 6 章 結論 **96**
6.1 本研究のまとめ 96
6.2 今後の展望 98
6.2.1 EPSR の応用 98
6.2.2 今後の複製サーバ配置先決定手法 100

謝辞 **101**

論文目録 **103**

参考文献 **105**

図 目 次

1.1 ExaPeer 上の複製サーバに接続する手順. 4
1.2 ExaPeer による複製サーバの動的再配置. 5

3.1 EPSR の動作概念図. 34
3.2 APC degree に基づいたマーカの設定. 37
3.3 基本メカニズムではマーカ設定が失敗する例. 40
3.4 Virtual Path Expanding (VPE). 41
3.5 Virtual Demand Raising (VDR). 42
3.6 マーカとなったホスティングマシンが過負荷となる例. 46
3.7 APC degree と Load indicator に基づいた複製サーバ数の増加. . . 47
3.8 複製サーバ数が誤って減少しうる例. 49
3.9 Dummy Query によって複製サーバ数が需要の規模に適した数に保たれる例. 50

4.1 サービス A およびサービス B のクライアント数の推移. 57
4.2 EPSR による複製サーバの配置先候補点数の推移. 59
4.3 配置先候補点との RTT の累積頻度グラフ. 61
4.4 同時接続数の分布. 62
4.5 発生させたリクエスト数の推移. 65
4.6 EPSR によって設定されたマーカ数の推移. 66
4.7 EPSR と ROOT_ONLY におけるホップ数の比較. 67
4.8 EPSR と ROOT_ONLY における ラウンドトリップ時間 (RTT) の比較. 68
4.9 ホップ数の累積頻度グラフ. 69
4.10 RTT の累積頻度グラフ. 70
4.11 需要発生地域が移動する状況における EPSR が設定したマーカ数の推移. 73

4.12 需要発生地域が移動する状況で EPSR を使用した際のホップ数の推移. 74
4.13 需要発生地域が移動する状況で EPSR を使用した際の RTT の推移. 75
4.14 ホップ数の累積頻度グラフ（CLS(TS) の場合）. 76
4.15 RTT の累積頻度グラフ（CLS(TS) の場合）. 77
4.16 ホップ数の累積頻度グラフ（CLS(Meridian) の場合）. 78
4.17 RTT の累積頻度グラフ（CLS(Meridian) の場合）. 78
4.18 VPE，VDR および GNP の効果. 81
4.19 Dummy Query を有効および無効にした際の RTT の累積頻度グラフ. 83
4.20 Dummy Connection を有効および無効にした際の同時接続数の累積頻度グラフ. 84

5.1 各メッセージのトラフィック量とホスティングマシン数との関係. 92

表 目 次

1.1 複製サーバの配置先決定手法の定性的比較. 13

2.1 固定配信網方式を採る手法の定性的比較. 18
2.2 オフライン計算方式を採る手法の定性的比較. 21
2.3 オンデマンド負荷軽減網方式を採る手法の定性的比較. 25
2.4 通信遅延考慮型配信網方式を採る手法の定性的比較. 28

3.1 EPSR の構成要素と課題の対応. 53

4.1 EPSR による複製サーバの配置先決定の評価で使用した各種パラメータ. 59
4.2 アクセス頻度を変動させた際の需要発生地域の特定部に対する評価で使用した各種パラメータ. 65
4.3 需要発生地域を移動させた際の需要発生地域の特定部に対する評価で使用した各種パラメータ. 72
4.4 ネットワーク座標系，VPE および VDR の効果の評価で使用した各種パラメータ. 80
4.5 Dummy Query および Dummy Connection の効果の評価で使用した各種パラメータ. 82

第1章　序論

1.1　本研究の背景と動機

インターネットの利用者数は年々増加傾向にあり，コンテンツの配信にかかる負荷が増加し続けている．インターネット (The Internet) とは，インターネットプロトコル [1] を利用してコンピュータネットワークを相互に接続した地球規模のネットワークを指す．インターネットは初め主に学術研究用途に利用されていたが，1990 年代を境に商業目的の利用が多数を占めるようになった．1995 年 9 月にはインターネット利用登録要求の 97 % が商用目的のものとなっている [2]．これ以降，インターネットの存在が一般に知られるようになり，一般の利用者が増加することとなった．Internet World Stats によれば，インターネットの利用者数は 2000 年度において 3.6 億人であったのに対して，2010 年度現在では 20 億人と約 5.5 倍増加した [3]．インターネットの利用が促進されることで，インターネット上でのコンテンツ配信による負荷が増大した．Cisco Systems が発表した資料 [4] によると，2009 年の月当たりのトラフィック量は 1995 年当時の 85 倍に当たる約 15 エクサバイトであった．同資料は，2014 年の月当たりのトラフィック量は 2009 年の約 4 倍の 64 エクサバイトに到達すると予測している．

インターネットが普及するにつれて，安定したサービスの提供を阻害する *Flash Crowd* [5–12] と呼ばれる現象がインターネット上で発生するようになった．Flash Crowd とは，インターネットサーバへの突発的なアクセス集中を指す．Flash Crowd が発生すると，インターネットサーバのアクセス数が数分のうちに通常時の数倍から数十倍に達することが知られている [10]．インターネットサーバがアクセス負荷に耐えられるだけの性能を有していない場合，サーバは過負荷状態になり最悪停止してしまう．サーバの性能とは，サーバが稼働する物理マシンの搭載 CPU 性能やメモリ量などや，その物理マシンが接続するネットワークの帯域などを指す．

Flash Crowd の発生要因は多岐に渡るため，Flash Crowd の発生を予測することは困難である．Flash Crowd の発生要因は第三者によるサイト情報の 1) 公開情報

一覧への掲載，2) 放送，3) 個人間情報伝播などがある．1) の公開情報一覧への掲載とは，例えばニュースサイトへの投稿やポータルサイトへの登録などがある．一例として Slashdot Effect [5,13–15] と呼ばれる，Slashdot [16] のような知名度の高いサイトに掲載されたサイトのアクセス数が急激に増加する現象がある．2) の放送とは，例えばテレビによる報道が挙げられる．3) の個人間情報伝播とは，例えば Facebook [17] などのソーシャル・ネットワーキング・サービス (SNS) や Twitter [18] などのコミュニケーション・サービスによって，個人間のコミュニケーションを通じて情報が広まっていくことである．これらの情報伝達手段によって短時間に多くの人へサイト情報が伝わり，サイト情報を得た人々が一斉にサーバへアクセスすることで Flash Crowd が発生する．この情報伝播は第三者によって行われるものであるため，サーバ運用者が Flash Crowd の発生を予測することは難しい．

大量のクライアントアクセスを処理する一般的な方法として，**複製サーバ**の配置による負荷分散が知られている．複製サーバとは，オリジナルのサーバを構成するプログラムやデータを複製することで，オリジナルのサーバと同一の処理を実行できるようにしたサーバである．複製サーバを複数の物理マシンで稼働させ，クライアントアクセスをそれぞれの複製サーバに振り分けてサーバ性能の総和を増加させる．理想的には，用意した物理マシンと接続するネットワークの性能の総和に見合うクライアントアクセスを処理できる．Flash Crowd の発生を想定して，大量の複製サーバをあらかじめ配置しておくことができれば，Flash Crowd によるサーバの異常停止を回避することができる．

しかし，Flash Crowd に対応するために複製サーバを常時配置し稼働しておくことには 2 つの問題がある．第一に，配置しておくべき複製サーバの数や位置の予測は困難である．先に述べたように，Flash Crowd の発生要因は多岐に渡る．そのため，要因によって発生する Flash Crowd の規模や発生地域も様々である．よって，配置しておくべき複製サーバの数や位置を事前に決定することは難しい．第二に，複製サーバの稼働コストの問題がある．文献 [19] では，大量の計算機資源によってインターネットサービスのスケーラビリティを向上させる技術 [20–27] は知られているが，一方でサービスに与える計算機資源はその量に合わせてコストが発生すると指摘している．これは複製サーバを設置する場合も同様である．複製サーバを稼働するためには，物理マシンとネットワークを用意し設置しなければならない．そのため，Flash Crowd に対応するために配置した余剰な複製サーバの稼働にはマシンの設置費用，運用費用やネットワーク使用料などの追加費用が常時生じる．

近年では，複製サーバを配置するための計算機資源を共有するシステムが，複製サーバの動的配置を可能にすることで上記の問題の解決を図っている．このようなシステムの例として Amazon Elastic Compute Cloud (Amazon EC2) [28] などのホスティングサービスが挙げられる．計算機資源を共有するシステムでは，**ユーティリティコンピューティング** [29] という概念に則り，サービス提供者が複製サーバの配置のために計算機資源を共有する．サービスの提供者は世界中にあらかじめ設置された大量のホスティングマシンを共有し，需要に応じて複製サーバをホスティングマシン上に配置することによりサービスの提供を行う．このようなシステムでは，サービスの需要が低い時は必要最小限の複製サーバを配置することでシステムの利用コストを抑え，サービスの需要が高いときは複製サーバの数を増加しなければならない．

1.2 ExaPeer

ユーティリティコンピューティングに則った理想的な，複製サーバを配置するための計算機資源を共有するシステム（ホスティングマシン共有システム）の動作例を示す．本論文では，この理想的なホスティングマシン共有システムを ExaPeer と呼ぶ．

1.2.1 ExaPeer の概要

ExaPeer は世界中に設置した数百台から数千台の信頼できるホスティングマシンから成る．ExaPeer を利用するサービス提供者は設置されたホスティングマシンを共有する．ExaPeer では利用者のサーバを実行するために，仮想化技術 [30] によってハードウェア資源を仮想化する．サービス提供者はあらかじめ複製サーバの実行に必要なプログラムやデータを複製しホスティングマシンに配備しておく．ExaPeer は配備されたプログラムやデータから複製サーバをホスティングマシン上で実行し，個々のサービスをクライアントに提供する．また ExaPeer はクライアントに対して接続すべき複製サーバの IP アドレスを提供する機能を持つ．

ExaPeer は Flash Crowd 発生時において適切なホスティングマシン上で複製サーバを起動する．以下では ExaPeer の動作の流れを示す．図 1.1 は，クライアントが ExaPeer 上の複製サーバに接続する手順を示す．クライアントは ExaPeer に対して，受けたいサービスを提供している複製サーバの IP アドレスを要求する（図 1.1

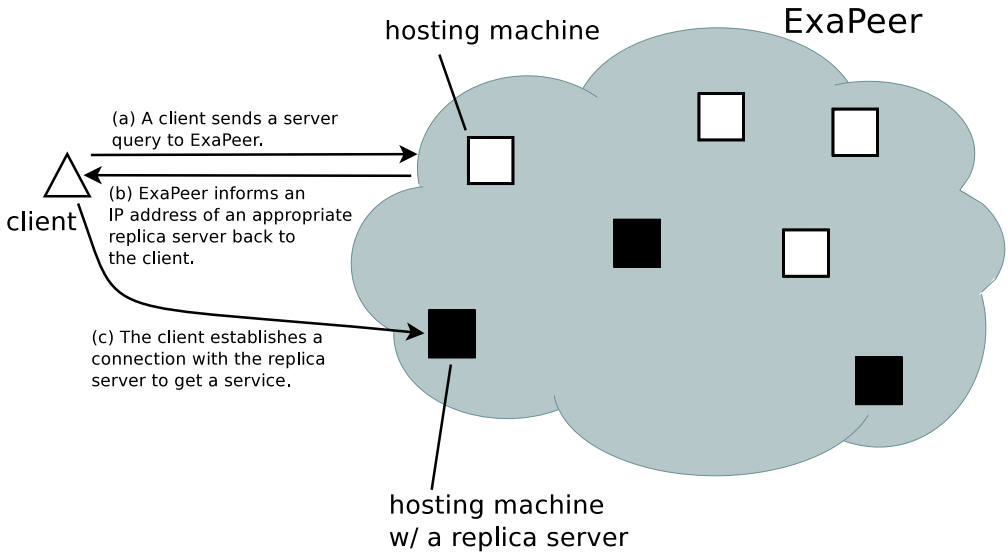
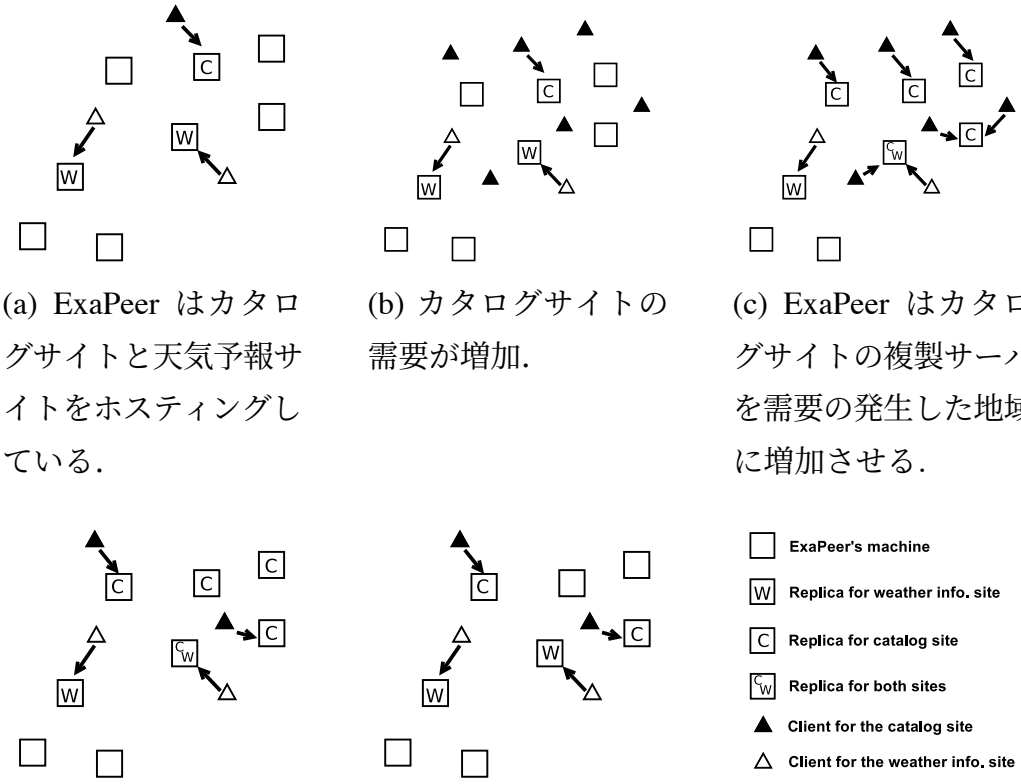


図 1.1: **ExaPeer 上の複製サーバに接続する手順．**クライアントは ExaPeer に接続すべき複製サーバの解決を依頼する．ExaPeer はクライアントとの通信遅延が小さくかつ新たなクライアントが接続可能な複製サーバの IP アドレスをクライアントに通知する．クライアントは受信した IP アドレスが示す複製サーバに接続しサービスを受ける．

(a))．ExaPeer はクライアントに対して適切な複製サーバを特定し，その IP アドレスをクライアントに通知する（図 1.1 (b)）．クライアントは ExaPeer から受け取った IP アドレスを元に，サービスを受けるための接続を複製サーバに対して確立し，複製サーバからサービスを受ける（図 1.1 (c)）．

図 1.2 は ExaPeer が複製サーバを動的に再配置する様子を示す．サービス提供者は ExaPeer を利用するために，提供したいサービスのサーバを構成する要素を ExaPeer 上のホスティングマシンに配備する．これには例えば，サーバプログラムや提供するデータファイル，ライブラリ，オペレーティングシステムなどが含まれる．図 1.2 の例では，ExaPeer はカタログサイトと天気予報サイトを同時に提供している（図 1.2 (a)）．カタログサイトの需要が増加すると（図 1.2 (b)），ExaPeer は EPSR によってカタログサイトのクライアント群の近傍にあるホスティングマシンを複製サーバの配置先と決定し，複製サーバを実行する（図 1.2 (c)）．クライアントは最寄りの複製サーバから直接サービスの提供を受ける．カタログサイトの需要が減少すると（図 1.2 (d)），ExaPeer は EPSR によって需要の減少をすぐに検出し，不要なカタログサイトの複製サーバを停止する（図 1.2 (e)）．



(a) ExaPeer はカタログサイトと天気予報サイトをホスティングしている．

(b) カタログサイトの需要が増加．

(c) ExaPeer はカタログサイトの複製サーバを需要の発生した地域に増加させる．

(d) カタログサイトの需要が減少．

(e) ExaPeer は負荷の低い余剰なカタログサイトの複製サーバを停止する．

図 1.2: **ExaPeer による複製サーバの動的再配置．** ExaPeer は需要変動に応じて自動で需要発生地域のホスティングマシンを複製サーバの配置先として選択する．

1.2.2 ExaPeer 実現の課題

ExaPeer は Flash Crowd のような急激な需要増加が発生しても，サービスの質を下げないように複製サーバの配置先を変更する基盤である．複製サーバを適切に配置するために，ExaPeer は複製サーバの配置先を適切に決定する手法が必要である．Flash Crowd に対して有効な複製サーバの配置先決定手法があれば，ExaPeer は Flash Crowd の発生に合わせて複製サーバを起動するホスティングマシンを知ることができる．ExaPeer はこの決定に従って複製サーバを起動すればよい．

Flash Crowd に対して有効な複製サーバの配置先決定手法がない場合，過去の経験に基づいてアドホックに複製サーバを配置しなければならなくなる．この場合

必ずしも Flash Crowd による影響を軽減するような配置になるとは限らない．例えば配置先として決定したホスティングマシンが需要の発生地域から遠い場合，複製サーバに接続したクライアントはサービスの遅延が大きくなり，サービスの質が低下する．また，もし配置先として決定したホスティングマシンの数が需要に対して少なかった場合，複製サーバが負荷に耐えきれず停止してしまう．従って Flash Crowd に対して有効な複製サーバの配置先決定手法は ExaPeer の実現に必要不可欠なものであるといえる．

1.3 本研究の目的

本研究では，ホスティングマシン共有システムに対して Flash Crowd の発生時に適切な複製サーバの配置先を決定できる手法を提供することを目的とする．Flash Crowd に対して適切な複製サーバ配置先の決定とは，Flash Crowd 発生時に生じる負荷を分散しサービスを継続して提供し続けられるように，複製サーバを配置すべきホスティングマシンを選択することである．

Flash Crowd 発生時に増大する負荷は大別して，1) サーバに対する負荷と 2) ネットワークに対する負荷がある．サーバに対する負荷とは，接続してきたクライアントに対してサービスを提供するために行うサーバ上の処理によってサーバに発生する負荷である．クライアントに対してサービスを提供するためには，サーバは必要に応じて，格納しているデータからクライアントが要求する情報を取り出しコンテンツを生成する．従来は静的に生成しておいたコンテンツが多かったが，近年ではクライアントの要求への応答性を高めるために，コンテンツを動的に生成することが多い．このときサーバではコンテンツを生成するために CPU やメモリなどに負荷が発生する．Flash Crowd 発生時に特定のサーバへアクセスが集中した場合，コンテンツの提供に必要な CPU やメモリなどの資源が不足し，サービスに遅延が生じる恐れがある．

一方，ネットワークに対する負荷とは，クライアントがサーバからサービスを受けるために行った通信によって発生する通信負荷である．クライアントはサービスを受けるためにサーバと通信するため，クライアントと接続を受けたサーバとの間のネットワークはサービスに応じて通信負荷が生じる．もし Flash Crowd 発生時にクライアントが特定のサーバに集中して接続した場合，そのサーバまでの経路にあるネットワークはトラフィックが集中し負荷が増大する．このような場合，

輻輳などによって通信速度が低下し，サービスの質が低下する恐れがある．

本研究は以上に挙げた Flash Crowd に起因する 2 つの負荷を分散できるように，複製サーバの配置先を決定できる手法を示す．なお，本研究が対象とする複製サーバは次の 2 つの条件を満たすものとする．第一に，平常時は需要が少ない単体のサーバを本研究の対象とする．これはそのようなサーバは平常時の負荷が小さいため負荷分散の設備を用意しておらず，Flash Crowd による影響が特に大きいと考えられるためである．第二に，クライアントとの通信時に構成要素が変更されないサーバを本研究の対象とする．本研究はフロントエンドサーバなどの，CPU やメモリ，ネットワークに対する負荷が大きいサーバに対して複製サーバの配置先を決定する．クライアントとの通信時に更新を行うデータはストレージとの入出力処理による負荷が大きいため分離し，キーバリューストア [31–35] などの，入出力処理による負荷の分散に特化した手法を適用する．本研究の適用範囲の議論は第 5 章に詳しい．

1.4 本研究の課題

複製サーバの配置先決定手法は，次の 3 点の課題を解決しなければならない．第一に，**高速性**がある．Flash Crowd を予測することは難しいため，Flash Crowd の発生後に複製サーバの配置先を計算する必要がある．Flash Crowd は発生から数分程度で需要増加のピークに達するものがあるため，複製サーバ配置先の決定は数分以内で完了しなければならない．例えば文献 [10] が報告する Flash Crowd は，需要の増加がピークに達するまでに用した時間が 40 秒から 15 分程度であった．

第二に，**通信遅延**を考慮する必要がある．通信負荷を分散させるために，クライアントとの通信遅延が小さいホスティングマシンに複製サーバを配置するべきである．需要が増加した際にサーバが過負荷状態になる事態を回避するだけでよいならば，複製サーバの配置先を単純に増加させるという方法がある．ところが単純に配置先を増加させた場合，負荷分散のためにクライアントは通信遅延の大きい複製サーバに接続させられることになる．一般に通信遅延の大きいサーバはネットワーク上で遠い位置にあることが多いため，クライアントとサーバ間の通信で生じるパケットの経由するネットワークが多くなる．これはネットワークに与える負荷を増加させる．また，サービスを受けるまでの遅延を増加させることになり，サービスの質の低下を招く．

第三に，マシンの**性能制約**を考慮する必要がある．複製サーバによってホスティングマシンが過負荷にならないように，複製サーバの配置先を決定しなければならない．複数のサービスで複製サーバ用のホスティングマシンを共有するシステムでは，ホスティングマシンの負荷と性能制約は状況に応じて異なる．もし複製サーバの配置先決定手法が性能制約を考慮せず，アドホックに配置先を決定すると，複製サーバを配置したことによって配置先のホスティングマシンが過負荷となることがある．そのような場合複製サーバの動作に遅延が生じたり，最悪複製サーバやホスティングマシンが停止してしまう恐れがある．

これまでに複製サーバの配置先を決定する手法が数多く提案されてきた．既存手法は，1) 固定配信網方式，2) オフライン計算方式，3) オンデマンド負荷軽減網方式，4) 通信遅延考慮型配信網方式の 4 方式に大別される．

固定配信網方式 [36–42] は，需要の予測にしたがってホスティングマシン間で配信網を静的に構築しておき，この配信網を使用して複製サーバの配置先を決定する手法である．ホスティングマシンはオリジナルサーバを起点とする配信網を構築し，この配信網を利用して需要の発生した地域の近傍にあるホスティングマシンを配置先として選出する．固定配信網方式はあらかじめ構築した配信網に従って配置先を決定するため，高速性を満たす．しかし，固定配信網方式は通信遅延を必ずしも小さくしない．予測した需要変動に対して静的に配信網を構築するため，予測によっては複製サーバの配置がクライアントとの通信遅延を小さくするものであるとは限らないからである．また固定配信網方式の多くは性能制約を考慮せず，需要の有無のみで配置先を決定する．性能制約を考慮するものも，通信遅延の場合と同様の理由で，必ずしも需要の分布に沿った配置になるとは限らない．

オフライン計算方式 [43–53] は，収集した需要分布に関する情報を特定の計算サーバ上で集中処理し，最適な複製サーバの配置先の近似解を算出する手法である．オフライン計算方式では，全ホスティングマシンからクライアントのアクセス記録を収集し，この記録から需要の分布を算出し複製サーバの配置先を決定する．オフライン計算方式は通信遅延を考慮して配置先を決定する．また，性能制約を考慮した配置先の決定も可能である．しかし，オフライン計算方式は高速性を満たすことが難しい．オフライン計算方式は計算を行うサーバが全ホスティングマシンのアクセス記録や性能情報，負荷情報を収集しなければならず，収集を行う特定のサーバがアーキテクチャ上のボトルネックとなってしまうからである．

オンデマンド負荷軽減網方式 [13,54–62] は，Flash Crowd の発生時に使用する負荷軽減用ネットワークをホスティングマシン間で動的に構成し，このネットワーク

上で複製サーバの配置先を決定する手法である．本方式は Flash Crowd の発生時，クライアントの接続先を Peer-to-Peer 技術などを用いて構成した負荷軽減用ネットワークに切り替える．そして負荷軽減用ネットワーク上で各ホスティングマシンが，クライアントのアクセスもしくはオリジナルのサーバの指示に応じて複製サーバを起動する．オンデマンド負荷軽減網方式は高速性を満たす．Peer-to-Peer 技術などの自律分散型システムの実現技術によって，各ホスティングマシンは配置先決定に必要な情報を与えられ，自律的に複製サーバを自身上に配置するべきか判定できるからである．しかし，オンデマンド負荷軽減網方式は通信遅延を考慮せずに複製サーバの配置先を決定する．また本方式はホスティングマシンが過負荷にならないように負荷分散を行うが，性能制約を考慮して配置先を選択するとは限らない．

通信遅延考慮型配信網方式 [12,63–68] は通信遅延を考慮したオーバレイネットワークをホスティングマシン間で構築し，これを利用して複製サーバの配置先を決定する手法である．本方式では，クライアントのリクエストを Peer-to-Peer 技術などの自律分散型システムの実現技術を用いて構成したオーバレイネットワーク上で転送し，この際にリクエストの転送が集中するホスティングマシンを複製サーバの配置先として選出する．通信遅延考慮型配信網方式は高速性を満たす．オンデマンド負荷軽減網方式と同様に，Peer-to-Peer 技術などによって各ホスティングマシンが自律的に自身を複製サーバの配置先とするか決定できるからである．また通信遅延考慮型配信網方式は通信遅延を，オーバレイネットワーク上での経路選択の制限において削減するように複製サーバを配置する．クライアントリクエストの転送遅延が小さくなるようにオーバレイネットワークを構築することで，クライアントリクエストが複製サーバに到達するまでのオーバレイ上での転送遅延を小さくする．しかし，物理ネットワークにおいてクライアントと複製サーバ間の通信遅延が必ずしも小さくなるとは限らない．また，通信遅延考慮型配信網方式はホスティングマシンの性能に関係なく複製サーバの配置先を決定する．

以上のように，既存手法は Flash Crowd による影響を軽減する複製サーバの配置先決定手法が解決すべき 3 つの課題の全てに対応することが難しい．よって，既存手法を Flash Crowd 対策に適用するには限界があり，新たな手法が必要であるといえる．

1.5 本研究の提案

本研究では，Flash Crowd による影響を軽減する複製サーバの配置先決定手法 *ExaPeer Server Reposition* (EPSR) を提案する．EPSR は複製サーバの配置先決定までの高速性，クライアントと配置先となるホスティングマシン間の通信遅延，および配置先となるホスティングマシンの性能制約すべてを考慮した複製サーバの配置先決定手法である．EPSR によってホスティングマシン共有システムは，Flash Crowd による需要増加に合わせて高速に，発生した負荷を処理できる数の配置先マシンを選び出すことが可能になる．また EPSR によってホスティングマシン共有システムは，通信遅延の増加を抑えながら Flash Crowd による負荷を複製サーバと接続する各ネットワークに分散することが可能になる．EPSR は配置先を決定する際，直接接続することになるクライアントとの通信遅延が小さくなるようなホスティングマシンを複製サーバの配置先として決定する．さらに EPSR によってホスティングマシン共有システムは，どのホスティングマシンも過負荷にならないように複製サーバの配置先を決定することが可能になる．EPSR はホスティングマシンの現在の負荷情報と性能情報を元に，ホスティングマシンが過負荷にならないように複製サーバの配置先を動的に増加する．

1.5.1 提案手法の概要

EPSR は各ホスティングマシンが収集可能な局所情報のみを使用して配置先を決定することで，高速性，通信遅延，性能制約の 3 点の課題を達成する．EPSR は高速性を実現するために，全サーバのアクセスログや性能情報，負荷情報のような大域情報を使用せずに配置先を決定する．個々のホスティングマシンは自律的に需要の変動を検出し，必要に応じて複製サーバの配置先候補となる．ホスティングマシンの独立な需要変動の検出と需要の発生地域の特定を実現するために，EPSR はホスティングマシン間でオーバレイネットワークを構築し，需要変動に関する情報を局所的に共有する．

次に EPSR は通信遅延を低下させるために，ホスティングマシン間のオーバレイネットワークの構築時に物理ネットワークのトポロジを考慮する．これを実現するために EPSR は，ネットワーク座標系 [69–77] と分散ハッシュテーブル (Distributed Hash Tables, DHT) [78–84] のひとつである CAN [79] を組み合わせてオーバレイネットワークを構築する．個々のホスティングマシンは，オーバレイ上で隣接す

るマシンとのみメッセージを交換することでクライアントのアクセス経路を分析し，サービスへの需要が増加もしくは減少している地域を特定する．ホスティングマシンはアクセス経路の分析から複製サーバの実行による効果を見積もり，複製サーバの配置先となるかどうか決定する．

最後に EPSR は性能制約を満たすために，ホスティングマシンが自律的判断に基づいて需要発生地域へ複製サーバの配置先を追加できるようにする．個々のホスティングマシンは複製サーバの負荷情報と自身の性能情報を元に，ホスティングマシンが過負荷にならないように隣接するマシンから追加の配置先を独立して選択する．物理ネットワークのトポロジを考慮したオーバレイネットワークにおける位置関係を用いるため，隣接するホスティングマシンも需要の発生地域に近いことが保証される．これにより，追加した複製サーバの配置先との通信遅延が必要以上に大きくなることを防ぐ．

1.5.2 評価方法

本研究では，EPSR をオーバレイ構築ツールキット Overlay Weaver [85,86] に実装し，シミュレーションによる評価を行う．シミュレーション中で Flash Crowd を想定した負荷を発生させ，需要の急激な増加に対して EPSR が適切な配置先の決定が行えることを確認する．具体的には次の指標について計測を行う．第一に，高速性について確認するために EPSR が決定した複製サーバの配置先数の推移と決定に要する時間を測定する．第二に通信遅延について確認するために，配置先のホスティングマシンとクライアント間の通信遅延とオーバレイ上のホップ数を計測する．第三に性能制約について確認するために，配置先となるホスティングマシンの負荷を測定する．これにより EPSR が Flash Crowd に合わせて，ホスティングマシンが過負荷になることや通信遅延が増加することのないように，配置先を決定することを確認する．さらに EPSR が使用している要素技術の効果の測定も行う．

シミュレーションでは，需要が増減した際の EPSR の動作を確認するために，クライアント数を増減させて測定を行う．また需要の発生地域が移動した際の EPSR の動作を確認するために，クライアントの位置を変動させて測定を行う．シミュレーションでは，Transit-Stub モデル [87] に基づく擬似ネットワークによるものと，インターネットでの通信遅延の実測データの 2 つのデータセットを使用する．また比較のため，シミュレーションではオフライン計算方式によって計算された近

似解に基づく配置と，発見的手法による配置の場合で計測を行う．

1.6 本研究の貢献

本研究の第一の貢献は，複製サーバの動的再配置を行うホスティングマシン共有システムに対して，Flash Crowd による影響を軽減するように複製サーバの配置先を決定する手段を提供することである．これにより，サービスの提供者は EPSR が適用されたホスティングマシン共有システムを利用することで，Flash Crowd 発生時における急激な需要の変動に合わせて柔軟に複製サーバの配置を行うことが可能になる．

本研究の貢献は，特に提供開始直後の規模の小さいサービスを運営する者にもたらされる．サービスの提供開始直後は需要の見積もりが難しく，需要のピーク時に必要な複製サーバの数を決定することが困難であるためである．また需要を見積もることが困難であることから，そのサービスに割り当てるべきコストを決定することも難しい．EPSR は需要変動を検出するための仕組みを備えているため，サービスの提供者による需要の見積もりは不要になる．また EPSR は需要変動に応じて配置先の決定を行うため，需要の規模に合わせた複製サーバの配置が可能になる．さらにサービス提供者が Flash Crowd 対策用の複製サーバを過剰に個別設置する必要がなくなるため，サービス提供者が負担する複製サーバの運用コストの低減も期待できる．

本研究の貢献はまた，ホスティングマシン共有システムを運用する者にももたらされる．EPSR によってホスティングマシン共有システムは，使用者のサービスの需要をまかなうために必要な配置先を，Flash Crowd のような急激な需要増加の際に知ることができる．これによりホスティングマシン共有システムは，Flash Crowd に対して効果的に複製サーバを配置することができる．また，EPSR はクライアントの通信遅延を下げるように配置先を選出するので，複製サーバへ接続したクライアントのサービス遅延の低下とネットワーク負荷の分散および低減に貢献する．さらに EPSR は，ホスティングマシンが過負荷にならないように配置先を決定するため，ホスティングマシン共有システム自身のサービス停止の回避に貢献する．

第 1.4 節で述べた既存の複製サーバ配置先決定手法と本研究の提案手法 EPSR の特徴を表 1.1 に示す．既存手法では，第 1.4 節で挙げた複製サーバの配置先決定

表 1.1: **複製サーバの配置先決定手法の定性的比較．**

	高速性	通信遅延	性能制約
固定配信網方式	○	△	△
オフライン計算方式	×	○	○
オンデマンド負荷軽減網方式	○	×	△
通信遅延考慮型配信網方式	○	△	×
本研究の提案手法 (*EPSR*)	○	○	○

手法の 3 つの課題をすべて解決することが難しい．従ってこれらの手法を Flash Crowd 対策のためのホスティングマシン共有システムに適用するには限界がある．一方 EPSR は既存手法と異なり，配置先の決定手法の課題として挙げた，高速性，通信遅延，性能制約の 3 点を解決する．これにより，ホスティングマシン共有システムが Flash Crowd の発生に合わせて動的に複製サーバの配置先を決定することができるようになったといえる．

1.7 本論文の構成

本論文は全 6 章からなる．**第 1 章**では本研究の背景，動機および目的について述べ，本論文で提案する複製サーバの配置先決定手法 EPSR について概観し，本研究の学術的貢献について説明した．

次に**第 2 章**では本研究の関連研究をまとめる．既存手法を 1) 固定配信網方式，2) オフライン計算方式，3) オンデマンド負荷軽減網方式，4) 通信遅延考慮型配信網方式の 4 種類に大別する．それぞれの手法についてまとめ，EPSR との違いを明らかにする．

そして**第 3 章**では，Flash Crowd による影響を軽減する複製サーバの配置先決定手法 EPSR について述べる．まず EPSR の設計要件と概要を述べる．次に EPSR が使用した要素技術について述べる．そして EPSR の設計について述べる．

第 4 章では，EPSR の効果を確認するために行ったシミュレーションについて述べ，EPSR の効果を明らかにする．

第 5 章では EPSR の定性的議論を行う．EPSR の適用範囲，EPSR のパラメータの決定方針，スケーラビリティ，サービス間での公平性，セキュリティについて議

論を行い，EPSR の定性的評価を行う．

最後に**第 6 章**で本論文をまとめ，今後の研究の方向性を示す．

第2章　関連研究

本章では本研究の関連研究についてまとめる．既存の複製サーバの配置先選出手法は 1) 固定配信網方式，2) オフライン計算方式，3) オンデマンド負荷軽減網方式，4) 通信遅延考慮型配信網方式の 4 種類に大別される．本章では第 1 章で挙げた高速性，通信遅延，性能制約の 3 つの課題全てを解決することがこれら 4 種の方式では難しいことを示し，本研究との違いを明らかにする．

2.1　固定配信網方式

固定配信網方式は複製サーバの配置先を決定するために使用する専用ネットワークを静的に構築する．固定配信網方式では専用ルータの設置や設定を行うことにより，配信用の専用ネットワークをホスティングマシン間であらかじめ構築しておく．配信用のネットワークは効率を考慮して，オリジナルのサーバを根とする木構造になっていることが多い．サーバに対するクライアントのリクエストはこの配信用ネットワークを介してオリジナルのサーバに配送される．中継点のホスティングマシンやルータがリクエストの転送を監視することで需要の有無を検出し，転送の集中する点を複製サーバの配置先とする．

固定配信網方式は高速性を満たす．固定配信網方式は配置先選出用の専用ネットワークを静的に構築する．配置先の決定順序はこの専用ネットワークに基づいて静的に決定されているため，配置先の決定は比較的短時間で可能である．しかし，固定配信網方式による通信遅延の低減効果は限定的である．配置先選出用に静的に構築したネットワークは過去のクライアントの分布に応じて決定されたものであるため，現在のクライアントの分布に対して適切であるとは限らないからである．配置先を変更するためには配信網を変更する必要があるが，固定配信網方式は配信網の動的な変更に対応していないため難しい．最後に，固定配信網方式では性能制約を考慮しないものが多い．本方式はクライアントのアクセスに応じて複製サーバを貪欲に配置するものであり，性能に応じた配置先の選択は原則

として行わない．

以下では固定配信網方式の例を示す．RaDaR [36] は，インターネットのルーティングプロトコルの一種である Border Gateway Protocol (BGP) [88,89] と Open Shortest Path First (OSPF) [90–92] に従ってオリジナルのサーバを根とするツリー構造を静的に構築する．RaDaR では複製サーバの配置先を決定するために replicator と呼ばれるモジュールを配置する．replicator は各 Autonomous System (AS) 単位と，OSPF の各 area 単位で配置され，オリジナルのサーバへのルーティングに沿ってツリー構造のネットワークを作る．複製サーバの負荷が増加すると，replicator はルータから入手したパケットのルーティング情報を元に，ルーティング経路にもっとも近くなるようなホスティングマシンを複製サーバの配置先とする．RaDaR は BGP や OSPF によって設定されたルーティング情報に基づいてツリー構造の配信用ネットワークを静的に構築する．そのため，Flash Crowd による負荷の増加に合わせて柔軟に配信用ネットワークの構成を変更することが難しい．例えば，ある area 内の急激な負荷の上昇に対応するために，area をまたがって複製サーバの配置先を決定することはできない．

SPREAD [37] は，各サービスごとにそのオリジナルサーバを終端とする鎖状のネットワークをホスティングマシン間で静的に構成する．SPREAD ではこのネットワークによって構築された階層構造を，アプリケーション層マルチキャスト [93] による配信木として扱うことで配置先を決定する．ホスティングマシンはこの配信木上でクライアントのリクエストをオリジナルサーバに向けて転送する．この際，各ホスティングマシンはクライアントのアクセス頻度に応じて複製サーバの配置先となるかどうか決定する．SPREAD では各オリジナルサーバごとに階層構造を静的に構築しておく必要がある．そのため，需要の発生している地域が移動すると構造が不適切になり，特定のホスティングマシンに負荷が集中する恐れがある．

Tang と Chanson [38] および Li ら [39] は，En-Route [37,94,95] 型配信網における配置先決定アルゴリズムを提案している．En-Route 型配信網とは，ホスティングマシン間でクライアントのリクエストをオリジナルサーバまで転送する過程で複製サーバを探索する型の配信網を指すもので，先に挙げた SPREAD はその一例である．これらの手法は En-Route 型配信網における配置先の決定をグラフの最適化問題として定式化し，最適配置を動的計画法を用いて計算する．本手法は，クライアントが受ける通信遅延などのコストと，複製サーバを配置することによってホスティングマシンが受けるコストの合計が最小になるように配置先を決定する．

本手法は配信網が静的に構築されていることを前提としている．そのため，発生した Flash Crowd に対してあらかじめ構築しておいた配信網が適切でないと，複製サーバの配置先の妥当性は低下する．

Akamai [40] は代表的な商用コンテンツ配信網 (Content Delivery Network, CDN) [96] である．Akamai はクライアントのアクセスを契機に，コンテンツのキャッシュやコンテンツ生成のためのアプリケーションプログラムをあらかじめ設置したホスティングマシン上に配置する．ホスティングマシンはインターネットサービスプロバイダ (ISP) 単位で設置し，クライアントのアクセスは最寄りのホスティングマシンに DNS (Domain Name System) リダイレクト[1]される．ホスティングマシンは DNS リダイレクトされたクライアントのアクセスを中継する際にコンテンツのキャッシュ生成を行う．このリダイレクトは BGP と独自のネットワーク計測によって得られた詳細なネットワーク情報に基づいて行われる [9,97–100]．Akamai は高速なネットワーク接続をホスティングマシン間で静的に構築し，このネットワーク上でクライアントに最も近いホスティングマシンを特定する．そのため Akamai の場合，Flash Crowd によるアクセス増加を予測してホスティングマシン間のネットワークを構成しておく必要がある．

Coppens ら [41] は固定配信網における複製配置用のアルゴリズム COCOA (Cooperative Cost Optimization Algorithm) を提案している．COCOA はクライアントの複製探索手順を活用する，貪欲法を基本としたアルゴリズムである．COCOA は複製探索手順と複製配置手順とで成る．COCOA の複製探索手順では，クライアントはサービスを提供するために十分な性能を持つホスティングマシンから，通信遅延が小さくスループットの高いものを選択する．このとき，複製を取得するまでにクライアントが要した遅延などのコストをホスティングマシンが記録する．COCOA の複製配置手順では，ホスティングマシンは記録した中で最もコストの高かった複製を選択し，もし十分な計算機資源を有している場合，自身を配置先とする．COCOA は配信網が静的に構築されていることを前提としているため，需要の発生地域が急激に移動した場合配置の妥当性が低下する．

Mulerikkal と Khalil [42] はクライアントの計算機資源をホスティング用途に活用する CDN である Distributed Content Delivery Network (DCDN) を提案している．DCDN は複製の配置先を計算するサーバを，全体の配置情報を管理するマスタサーバとローカルネットワークの配置情報を管理するローカルサーバとの二段階に構

[1]DNS サーバのレコードを変更し，FQDN (Fully Qualified Domain Name) 名に対応する IP アドレスを切り替える技術．

表 2.1: **固定配信網方式を採る手法の定性的比較．**

	高速性	通信遅延	性能制約
RaDaR [36]	◯	△	×
SPREAD [37]	◯	△	×
Tang と Chanson の手法 [38]	◯	△	△
Li らの手法 [39]	◯	△	△
Akamai [40]	◯	△	△
COCOA [41]	◯	△	△
DCDN [42]	◯	△	△

成する．マスタサーバはネットワーク全体をローカルサーバが設置された地域ごとに分割し，複製の配置先を地域ごとに管理する．ローカルサーバはホスティングマシンを管理し，複製の配置先情報を持つ．DCDN での配置先の決定は，ホスティングマシンの数とクライアントのリクエスト数から作成したモデルを待ち行列理論に基づいて解くことで行う．DCDN はマスタサーバとローカルサーバの階層化を静的に行い，計算機資源を提供するクライアントやローカルサーバの情報をあらかじめ登録する必要がある．よって Flash Crowd による急激なクライアント増加に合わせて DCDN の構成を変更することは難しい．

固定配信網方式の特徴を表 2.1 にまとめる．RaDaR および SPREAD は通信遅延の小さいホスティングマシンへ貪欲に複製サーバを配置するため，性能制約を満たすことが難しい．他の手法は性能制約を考慮して複製サーバの配置を行うが，いずれも静的に構築した配信網にしたがって複製サーバの配置先を決定するため，発生した Flash Crowd に対して決定した配置先が必ずしも通信遅延や性能制約を満たすとは限らない．

2.2 オフライン計算方式

オフライン計算方式は収集したクライアントのアクセス情報を集中処理することによって配置先を決定する手法である．オフライン計算方式では，オリジナルのサーバやホスティングマシンから過去のクライアントのアクセス記録を特定の配置先計算用サーバに収集する．配置先計算用サーバでは収集した情報に基づい

てクライアントの位置を特定し，あらかじめ用意したホスティングマシンの配置情報と照合して配置先を決定する．配置先の算出には，貪欲法などの発見的アルゴリズムや，グラフ理論に基づく全体最適化アルゴリズムなどを利用する．

オフライン計算方式は配置先決定手法の課題のうち，通信遅延と性能制約を満たす．これらはアルゴリズムの目的関数や制約条件として与えることで達成できる．しかし，オフライン計算方式は高速性を満たすことが難しい．これはオフライン計算方式が全クライアントのアクセス記録や全ホスティングマシンの性能情報および負荷情報といった，全体情報を要求するからである．これにより，オフライン計算方式が情報収集および計算用に特定の中央サーバを必要とするため，アーキテクチャ上中央サーバがボトルネックとなる．本方式で高速性を満たすには，中央サーバが行うホスティングマシンからのアクセス記録やホスティングマシンの性能情報および負荷情報の収集を数十秒以内に完了する必要がある．これは配置先の決定を Flash Crowd によるクライアント増加がピークに達するまでに行わなければならないからである．Flash Crowd によるクライアント増加は短いもので 45 秒であったと報告されている [10]．例えば 1 台のアクセス記録や性能情報，負荷情報のサイズが 1 MB であるホスティングマシンが 1000 台あった場合，約 1 GB の転送をインターネットを介して数十秒で完了する必要がある．ホスティングマシンやクライアントの増加に従って全体情報の情報量は増加するため，中央サーバがボトルネックとなるオフライン計算方式は高速性を満たすことが難しいといえる．

以下ではオフライン計算方式の例を示す．Khan と Ahmad [43] は発見的手法に基づく 10 の配置先決定手法の比較と解析を行っている．これらの方式はクライアントとの通信遅延の平均値が最小となるように配置先を静的に決定するものである．Khan と Ahmad は 6 つの A-Star アルゴリズム [43,101] と，2 つの bin packing アルゴリズム [101]，貪欲法と遺伝的アルゴリズムをそれぞれ 1 つずつ [102,103] 取り上げている．これらのアルゴリズムは複製サーバの配置先を静的に求めるものであり，計算量が比較的大きい．例えば貪欲法による HotSpot アルゴリズム [102] では，N 台のホスティングマシンに M 個の複製サーバの配置を決定する際の計算量は $O(N^2 + min(NlogN, MN))$ となる．

動的に複製サーバの配置先を決定するアルゴリズムとして貪欲法の計算量を削減したものがある．Tse [44,48] はホスティングマシンが N 台のときの計算量が $O(logN)$ である配置先決定アルゴリズムを提案している．Tse のアルゴリズムはクライアントのアクセス数と配置した複製サーバの占有領域を制限値以内に抑え

るように配置先を決定する．しかし Tse のアルゴリズムは通信遅延を考慮した配置先決定アルゴリズムではない．

Tang と Xu [45,47] は，Quality of Service (QoS) による制約を考慮した配置先決定アルゴリズムを提案している．多くの配置先決定アルゴリズム [95,102,104] が通信遅延の平均値を一定以下に抑えるものであるのに対して，Tang と Xu のアルゴリズムは通信遅延を制約以内に抑えるように配置先を決定する．Tang と Xu のアルゴリズムはホスティングマシン間のネットワークをグラフでモデル化し，動的計画法を用いることで多項式時間で制約に沿う配置先を決定する．また Tang と Xu のアルゴリズムはホスティングマシンへの複製サーバの保存コストを考慮する．しかし，本アルゴリズムは複製サーバが与える負荷に関しては考慮しない．

Karlsson と Karamanolis [46] は，要求する性能を最小コストで実現するような発見的配置先決定手法を，複数の手法から選択する手段を提案している．Karlsson と Karamanolis の手段は複製サーバの配置問題を整数計画問題として定式化し，再配置システムの構成や需要の変動値，クライアントの通信遅延の目標値を入力することで適切な配置手法を選択する．本手段は入力としてサービスの需要の変動値を必要とする．これはクライアントのアクセス記録の収集を必要とする．また，再配置の際に複製サーバの移動などで発生するコストは考慮するが，個々のホスティングマシンの性能制約は考慮しない．

Globule [49–51] は，クライアントのアクセス情報やホスティングマシンの位置情報から最適な配置を決定する．Globule は HotZone アルゴリズム [52,53] によって最適配置の近似解を算出する．HotZone アルゴリズムは貪欲法に基づく HotSpot アルゴリズム [102] を改良するもので，N 台のホスティングマシンに M 個の複製サーバを配置する際の計算量を HotSpot アルゴリズムの $O(N^2 + min(NlogN, MN))$ から $O(Nmax(logN, M))$ まで削減する．HotZone アルゴリズムはネットワーク空間を，クライアントの分布密度が均一なクラスタに分割し，クラスタ単位で配置先を計算することで計算量を削減する．ネットワーク空間はネットワーク座標系のひとつである Global Network Positioning (GNP) [69] によってユークリッド空間へ写像し分割を行う．HotZone アルゴリズムは計算開始時に全てのクライアントの位置情報を収集しておく必要があるため，Flash Crowd 発生時の配置先の決定には向かない．

オフライン計算方式の特徴を表 2.2 にまとめる．Tse のアルゴリズムと Globule 以外の手法は，通信遅延を満たすが，高速性を満たすことが難しく，また性能制約を考慮しない．Tse のアルゴリズムは性能制約を考慮する配置を決定するが，高

表 2.2: **オフライン計算方式を採る手法の定性的比較．**

	高速性	通信遅延	性能制約
A-Star [43,101]	×	◯	×
bin packing [101]	×	◯	×
HotSpot [102]	×	◯	×
遺伝的アルゴリズム [103]	×	◯	×
Tse のアルゴリズム [44,48]	×	×	◯
Tang と Xu のアルゴリズム [45,47]	×	◯	×
Karlsson と Karamanolis の手段 [46]	×	◯	×
Globule [52,53]	×	◯	◯

速性を満たすことは難しく，また通信遅延を考慮しない．Globule は通信遅延および性能制約を考慮した手法であるが，高速性を満たすことが難しい．

2.3 オンデマンド負荷軽減網方式

オンデマンド負荷軽減網方式は，ホスティングマシン群を用いてサーバの負荷軽減用ネットワークを構築し，このネットワーク上で複製サーバの配置先を決定する手法である．Flash Crowd によってアクセスが増加すると，クライアントの接続先を負荷軽減用ネットワークに切り替え，ホスティングマシンに配置しておいた複製サーバからサービスを受けさせる．負荷軽減用ネットワークは負荷に応じて自律的に複製サーバの数を増加する．ホスティングマシンによる自律的な複製サーバ数の調整を実現するために，オンデマンド負荷軽減網方式ではアクセス数の増減をホスティングマシンが検知できるように負荷軽減用ネットワークを構築する．ホスティングマシンが負荷の増加を検出すると，負荷軽減用ネットワーク上で近傍のホスティングマシンを新たに複製サーバの配置先として選出する．Peer-to-Peer 技術を利用してこのような特性を持つ負荷軽減用ネットワークを構築することが多い．Peer-to-Peer 技術を用いることで，限られた数のホスティングマシンやクライアントから得た局所的な情報を元に，需要の増減を推測することができるようになるためである．また，より効果的に複製サーバを配置するために，自律的な調整に加えてオリジナルサーバによる積極的な配置手法を備えるものもある．

オンデマンド負荷軽減網方式はホスティングマシンが自律的に複製サーバ数の調整を行う専用ネットワークを動的に構築することで，複製サーバの配置手法における課題のひとつである高速性を満たす．オンデマンド負荷軽減網方式では，ホスティングマシンは自律的に需要の増減を見積もり，複製サーバを自身に配置すべきかどうか判断する．このため，需要変動に関する情報をインターネット全体から収集することなく配置先を決定できる．これにより数十秒単位での配置先の決定を実現する．しかし，オンデマンド負荷軽減網方式は通信遅延を小さくするとは限らない．負荷軽減用ネットワークは負荷分散の目的で構築されており，クライアントとの通信遅延は考慮されていない．そのため，負荷軽減用ネットワーク上で近傍のホスティングマシンが，クライアントとの通信遅延が小さいものであるとは限らない．また，オンデマンド負荷軽減網方式の性能制約への考慮は限定的である．オンデマンド負荷軽減網方式はクライアントのアクセスを，ホスティングマシンの負荷を考慮して振り分ける．しかし，ホスティングマシンの性能制約を考慮して配置先を選択するとは限らない．また，性能制約を考慮するものでも，オリジナルのサーバの指示によって配置先を決定する方式となる．この場合はクライアントのアクセス情報やホスティングマシンの負荷情報などを収集する必要が生じるため，高速性を満たさなくなる．

以下ではオンデマンド負荷軽減網方式の例を示す．FCAN [54] は，ホスティングマシンが自律的に配置先を決定するために，ホスティングマシン間で Peer-to-Peer ネットワークを構築する．FCAN では，オリジナルのサーバが Flash Crowd の発生を検知すると，ホスティングマシン間で Peer-to-Peer ネットワークを構築し，クライアントのアクセス先を DNS ラウンドロビン[2]などによって Peer-to-Peer ネットワーク上の任意のホスティングマシンに切り替える．FCAN では Peer-to-Peer ネットワークの構成に必要な他のホスティングマシンのリストをあらかじめ生成しておくことによって，短時間での切り替えを可能とする．Peer-to-Peer ネットワークが構築されると，ホスティングマシンはクライアントの要求する複製サーバを Peer-to-Peer ネットワーク上で探索する．このとき，クライアントの要求を受けたホスティングマシンは新たに複製サーバの配置先となる．また，FCAN では積極的に複製サーバの数を増加させるために，オリジナルのサーバが任意のホスティングマシンを配置先として選出する．しかし，いずれの場合も配置先として選択されたホスティングマシンとクライアントとの通信遅延が小さいとは限らない．ま

[2]ひとつの FQDN 名に対して複数の IP アドレスを割り当て，DNS クライアントからの要求に応じて異なる IP アドレスを返す技術.

た，FCAN は DNS ラウンドロビンなどによる負荷分散は行うが，性能制約を考慮した配置先の決定は行わない．

CoDeeN [55,56] は，PlanetLab [105,106] 上で稼働する Peer-to-Peer 型 CDN である．CoDeeN はホスティングマシン全てを制御する中央サーバを必要とせずに CDN を実現する．CoDeeN はクライアントによる web サーバへのアクセスを中継する forward proxy と，特定の web サーバが受けるアクセスを中継する reverse proxy の二段階で構成される．forwarding proxy は他の reverse proxy を監視し，適切な reverse proxy を選択してクライアントのリクエストを転送する．CoDeeN は個々の proxy が自分自身の負荷情報を他の proxy と交換することによって，性能以上の負荷を受けないようにクライアントアクセスを proxy に分配する．しかし，複製の配置は性能制約に関係なくアクセスを受けた proxy に配置される．また CoDeeN は通信遅延を考慮しないで配置先を決定する．CoDeeN で通信遅延が小さくなる時は，クライアントが通信遅延の小さい forwarding proxy を選択し接続できた場合に限られる．

DotSlash [57] は，複数の web サーバで複製配置用の計算機資源を共有するシステムである．DotSlash に参加する web サーバは，自身が備える計算機資源の余剰分を Flash Crowd が発生している web サーバにレスキューサーバとして提供することでサービスの停止を回避する．DotSlash ではクライアントが接続したレスキューサーバが複製の配置先となる．DotSlash は DNS ラウンドロビンと HTTP リダイレクトを併用することで，クライアントの接続先をレスキューサーバの負荷が均一になるように設定する．また DotSlash に参加する web サーバは，自身の負荷に応じて，DotSlash 内での役割を変更する．web サーバの負荷が低いときはレスキューサーバとなって計算機資源を他の高負荷な web サーバに提供し，負荷が高いときは他のレスキューサーバにクライアントアクセスを配分する．しかし DotSlash は配置先の決定時に性能制約を考慮しない．また DotSlash は通信遅延を考慮しないため，クライアントとレスキューサーバの通信遅延は大きいことがある．

Moreno ら [58] は，ストリーミング配信に対応した CDN である Streaming CDN (SCDN) を提案している．SCDN は複製の配置先をクライアントがリダイレクトされたホスティングマシンとし，キャッシュ方式で複製を配置する．クライアントのリダイレクト先は通信遅延やホスティングマシンの余剰性能を考慮して行われる．しかし，SCDN は配置先の決定時に性能制約を考慮しない．また SCDN では一度決定した配置先は変更されないため，クライアントは負荷分散のために通信遅延の大きいホスティングマシンに接続させられることがある．

Backslash [13] は，構造化オーバレイネットワークを利用した web サーバのキャッシュネットワークシステムである．Backslash はホスティングマシン間で分散ハッシュテーブルによる構造化オーバレイネットワークを構築し，この論理構造に基づいて複製の配置先を決定する．Backslash を用いた場合，クライアントのリクエストは任意のホスティングマシンが受け取り複製の探索を行う．クライアントのアクセス頻度が一定値以上になると，Backslash は複製の配置先を 3 通りの方法で決定する．第一の方法では，構造化オーバレイネットワークのルーティングに従って配置先を一意に決定する．第二の方法では，それぞれの複製に対してクライアントのリクエストが最後に転送されてきたホスティングマシンを配置先とする．第三の方法では，クライアントの要求を受け付けたホスティングマシンを配置先とする．Backslash は Least Recently Used (LRU) アルゴリズムにしたがってアクセス頻度の低い複製を削除することで，決定した配置先が性能制約を満たすようにする．しかし，Backslash は通信遅延を必ずしも小さくしない．オーバレイ上での論理距離を縮めるホスティングマシンを配置先として選択するが，配置先とクライアントとの通信遅延が小さいとは限らないからである．

Stefano と Santoro [59] は，複数の CDN で計算機資源を共有する基盤として Content Distribution Grid (CDG) を提案している．CDG は CDN 間の計算機資源の提供を経済モデルで表現し，市場原理に基づく需要供給曲線によって調停を行いホスティングマシンの割り当てを動的に行う [107]．CDG は CDN が性能制約によって新たな複製を配置できなくなると，他の CDN で余剰となっているホスティングマシンを新たな配置先とする．これにより CDG は性能制約を達成できるホスティングマシン数を増加させ，性能制約を超過しないようにする．しかし，CDG は性能制約を満たすホスティングマシンの探索方法は提供しない．また，CDG は通信遅延を考慮しない．

複数のクライアントに分散してキャッシュされたファイルを分割してダウンロードさせることで，負荷分散を行いファイル配信の際のスループットを向上するシステムがある．BitTorrent [60] は，オリジナルサーバがあらかじめファイルをチャンクという単位に分割し，クライアント間でチャンクを共有する．BitTorrent ではチャンクのダウンロード先をチャンクを持つ複数のクライアントに分割して割り当てることで負荷分散を行う．CoBlitz [61] は，Consistent Hashing [108] の派生系である Highest Random Weight (HRW) アルゴリズム [109] を用いて，負荷が均一になるようにクライアントの分割ダウンロード先を割り当てる．Flashback [62] は，クライアントに複製を配置する際に離脱したクライアントのもつ複製の一部が採

表 2.3: **オンデマンド負荷軽減網方式を採る手法の定性的比較．**

	高速性	通信遅延	性能制約
FCAN [54]	◯	×	×
CoDeeN [55,56]	◯	×	×
DotSlash [57]	◯	×	×
SCDN [58]	◯	×	×
Backslash [13]	◯	×	△
CDG [59]	◯	×	△
BitTorrent [60]	◯	×	×
CoBlitz [61]	◯	×	×
Flashback [62]	◯	×	×

索不能になる問題を解決する．クライアントに複製の一部を配置する場合，クライアントが高頻度で負荷軽減網への参加と離脱を繰り返すと複製の探索成功率が低下してしまう．Flashback では確率論に基づいて近傍の複製を探索していくことで，高い確率で複製が探索できるようにする．これらのシステムはダウンロード先を複数割り当て，分割してダウンロードさせることで負荷分散を行うが，複製を配置する際に割り当て先の負荷は考慮しない．また，通信遅延を考慮した複製の配置は行わない．

オンデマンド負荷軽減網方式の特徴を表 2.3 にまとめる．いずれの方式も高速性を満たす．しかし，通信遅延を考慮した配置先の決定は行わない．また Backslash と CDG 以外の手法は，性能制約を考慮せずアドホックに複製サーバを配置する．Backslash と CDG は性能制約に従って実行中の複製サーバを停止するが，配置先の決定時に性能制約を考慮しない．そのためオンデマンド負荷軽減網方式では，複製サーバの配置先が発生した需要に対して十分な資源を残しているホスティングマシンであるとは限らない．

2.4 通信遅延考慮型配信網方式

通信遅延考慮型配信網方式は，オーバレイネットワーク上のルーティングの際に通信遅延を考慮することで，複製サーバとクライアントとの通信遅延が小さく

なるように配置先を決定する．通信遅延考慮型配信網方式はオンデマンド負荷軽減網方式と異なり，オーバレイネットワークの構造に通信遅延の大小を反映させ，オーバレイネットワーク上の経路選択の際に通信遅延の小さいものが選択されやすくなるようにする．通信遅延考慮型配信網方式はこのようなオーバレイネットワークをホスティングマシン間で構築することで，個々のホスティングマシンがクライアントとの通信遅延の大小を，クライアントが送信したリクエストの転送経路から自律的に判定できるようにする．

通信遅延考慮型配信網方式は，ホスティングマシンが自律的に複製サーバの配置先となるか判定できるようにオーバレイネットワークを動的に構築することで，高速性を満たす．また，オーバレイネットワークのルーティングテーブルを通信遅延を考慮して経路選択を行うように構築することで，クライアントとの通信遅延を小さくするようなホスティングマシンを選択できるようにする．しかし，通信遅延考慮型配信網方式による通信遅延の削減は，オーバレイネットワークの構造に制限される．この制限により通信遅延考慮型配信網方式での複製サーバの配置は，オーバレイネットワークのルーティングテーブルの制約内では最小であるにもかかわらず，制約外により通信遅延が小さくなるホスティングマシンが存在することがある．また通信遅延考慮型配信網方式は，配置先の決定に使用するオーバレイネットワークが性能制約を考慮しないため，性能制約を満たさない配置先となる場合がある．

以下では通信遅延考慮型配信網方式の例を示す．CoralCDN [12,63] は，通信遅延で区分した多階層のルーティングテーブルによって，通信遅延を考慮した配置先決定用のオーバレイネットワークを動的に構築する．CoralCDN は Distributed Sloppy Hash Table (DSHT) [110] を用いることで，分散ハッシュテーブル (DHT) のひとつである Kademlia [82] のルーティングテーブルを通信遅延で区分し多階層化する．DSHT によってホスティングマシンがもつルーティングテーブルは，設定された閾値ごとの通信遅延であるホスティングマシン群で構成される．このオーバレイネットワークを利用してホスティングマシンは周辺の需要を検出する．まず，ホスティングマシンは DSHT で構築したルーティングテーブルに基づいてクライアントのリクエストをオリジナルのサーバまで転送する．転送の際に低遅延のルーティングテーブルから参照することで，クライアント群から低遅延のホスティングマシンでより多くのリクエスト転送が発生するようにする．ホスティングマシンはリクエストの転送量が増加すると，周辺に需要があると判断し配置先となる．CoralCDN はホスティングマシンが自律的に配置先となるか判定するため

高速性を満たす．また，オーバレイネットワークの制約内において，クライアントとの通信遅延を下げるように配置先を決定する．しかし，ルーティングの制約外に通信遅延がより小さくなるホスティングマシンがあったとしても選択されない．また，CoralCDN では性能制約を考慮せずに配置先を決定する．

PAST [64] は，通信遅延を考慮した DHT である Pastry [80] を用いて，配置先を決定するためのオーバレイネットワークを動的に構築する．Pastry はオーバレイネットワークへ新規に参加する際に，通信遅延の小さいマシンを初めの接続先とする制約を設ける．これにより，通信遅延の小さいマシンから順に参照するようにルーティングテーブルを構築する．PAST は Pastry のオーバレイネットワークにおいて，クライアントのアクセス経路上にあるホスティングマシンに複製を配置する．したがって PAST は，Pastry のルーティングの制約内において通信遅延を削減するように配置先を決定するといえる．しかし，PAST はルーティングの制約外により通信遅延が小さくなるホスティングマシンが存在しても発見できない．また，PAST は配置先の決定時に性能制約を考慮しない．

SCAN [65,66] は，オーバレイネットワークの構造上において通信遅延を満たすように複製の配置先を決定する．複製の配置先を決定するために，SCAN は DHT のひとつである Tapestry [81] を用いて複製の配置先を決定する．SCAN は Tapestry が備えるアクセス先の局所化という性質を利用する．Tapestry は複製の位置を示すインデックス情報を，配信網のルートを担当するマシンに到達するまでの経路上のマシンに記録する．クライアントはルートまでの経路中で最初に到達した複製のインデックスを利用し，複製にアクセスする．このため，オーバレイ上で近傍にあるクライアントは同一の複製にアクセスするようになる．SCAN ではこの性質を利用して配置先決定用の配信木を構築する．この配信木はオリジナルのサーバをルート，ホスティングマシンをノード，クライアントをリーフとし，親子の通信遅延が小さくなるように構成する．SCAN は Tapestry のルーティングの範囲内において，クライアントと複製との通信遅延を小さくする．しかし物理ネットワークにおいて必ずしも通信遅延が小さくなるとは限らない．さらに SCAN は性能制約を満たすために，複製を通信遅延の大きい複製サーバに配置することがある．また，SCAN は性能制約を満たす配置先を選択する手段は提供しない．

CobWeb [67] は，DHT 等のオーバレイネットワーク上で動作する複製配信システムである．CobWeb は複製の配置を，複製の配置コストと複製によって低下する通信遅延の最適化問題を解くことで決定する．CobWeb では Honeycomb という分散実行可能な発見的数値アルゴリズムを用いることで，M をオリジナルサーバ

表 2.4: **通信遅延考慮型配信網方式を採る手法の定性的比較．**

	高速性	通信遅延	性能制約
CoralCDN [12,63]	◯	△	×
PAST [64]	◯	△	×
SCAN [65,66]	◯	△	×
CobWeb [67]	◯	△	×
RE [68]	◯	△	×

数，N をホスティングマシン数としたときに，計算量 $O(MlogMlogN)$ で通信遅延の最適化問題を解く．したがって CobWeb は高速性を満たす．しかし CobWeb はオーバレイネットワークでの通信遅延をモデル化するため，通信遅延の低減効果はオーバレイネットワークの構造に制限される．また，CobWeb は複製サーバの保存に使用するストレージ量といった静的に計算可能な負荷に関してはモデル化できるが，クライアントアクセスによって変動する負荷はモデル化できない．そのため CobWeb は性能制約を満たすことが難しい．

Waldvogel ら [68] は，DHT 上での複製の管理と探索の手段として Replica Enumeration (RE) を提案している．RE は配置した順で複製サーバに与えられた列挙値を利用することで，最寄りの複製サーバを二分探索で特定できる手法である．RE では複製サーバの探索手法側で通信遅延の小さい複製サーバを探索できるようにすることで，複製サーバの配置先は単純に定められる．具体的には，RE はサーバの識別名と列挙値を引数にとるハッシュ関数で得られる ID のホスティングマシンに複製サーバを配置する．RE がクライアントとの通信遅延が小さいホスティングマシンを配置先とできるかどうかは，あらかじめ指定する複製サーバの最大数に依存する．また，RE は配置先を決定する際にホスティングマシンの負荷を考慮しない．

通信遅延考慮型配信網方式の特徴を表 2.4 にまとめる．いずれの方式も高速性を満たす．一方，通信遅延は限定的に達成される．これは通信遅延考慮型配信網方式では，配置先として決定したホスティングマシンとの通信遅延が構築したオーバレイネットワークのルーティングプロトコルの制約内では最小であるにもかかわらず，制約外により通信遅延が小さくなるホスティングマシンが存在することがあるためである．また，通信遅延考慮型配信網方式はホスティングマシンの性

能制約を考慮せずに複製サーバの配置先を決定する．

2.5 まとめ

本章では，本研究の関連研究を 1) 固定配信網方式，2) オフライン計算方式，3) オンデマンド負荷軽減網方式，4) 通信遅延考慮型配信網方式の 4 種類に分類してまとめ，第 1 章で挙げた Flash Crowd による影響を軽減する複製サーバの配置先決定手法が満たすべき 3 つの課題である，高速性，通信遅延，性能制約を満たすことが難しいことを確認した．

本研究では高速性，通信遅延，性能制約の 3 つを全て満たす複製サーバの配置先決定手法 ExaPeer Server Reposition (EPSR) を提案する．次章では EPSR の詳細について述べる．

第3章　ExaPeer Server Reposition

本章では本研究の提案である，Flash Crowd による影響を軽減する複製サーバの配置先決定手法 ExaPeer Server Reposition (EPSR) について述べる．まず EPSR を設計する上で考慮した 4 つの要点について述べる．次に EPSR の概要について述べ，続いて EPSR の詳細を説明する．EPSR の詳細は，需要が発生している地域の特定部分と，配置先とするホスティングマシンの選定部分とに分割して説明する．

3.1　設計上の要点

EPSR は Flash Crowd のような急激な需要増加に対しても，その影響を軽減するように複製サーバの配置先を決定できる手法である．Flash Crowd のような急激な需要増加による影響を効果的に軽減するために，EPSR は第 1 章で議論した 3 つの課題である，高速性，通信遅延，性能制約を満たす必要がある．そのために EPSR は以下の 4 点の機能を備える．

1) 軽量な需要変動の検出

高速性を満たすために，EPSR は需要変動に対して高い応答性を備える必要がある．そのため EPSR は精度が多少低下しても，軽量な需要変動の検出機構を備えなければならない．一方でオフライン計算方式のような中央サーバによる集中処理方式は，短時間の需要変動の検出には向かない．集中処理方式は計算を行うサーバが全ホスティングマシンのアクセス記録を収集しなければならず，収集を行う特定のサーバがアーキテクチャ上のボトルネックとなってしまうからである．集中処理方式は現在の需要分布に関する大域情報を収集する必要がある．需要分布に関する大域情報とは，現在配置されている複製サーバのアクセスログから得られるクライアント情報や，ホスティングマシンの性能情報および負荷情報で構成される．Flash Crowd 発生時はクライアントアクセスの頻度が急速に増加するため，クライアント情報や負荷情報は数秒単位で頻繁に変動し，またその容量も通

常時より大きくなる．複製サーバは ExaPeer によってインターネット上に分散配置されているため，大域情報を高頻度で収集することは難しい．

2) **複製サーバへのアクセス透過性**

高速性を満たすためには，配置先を変更した複製サーバへ迅速にクライアントアクセスを振り分けられなければならない．そのために EPSR はクライアントのアクセスを新規に配置された複製サーバへ透過的に切り替える仕組みを必要とする．あるサービスへの需要が短時間で変動したとき，EPSR は数十秒という短時間で大量の複製サーバの配置先を変更する．そのため，EPSR は DNS ラウンドロビン による複製サーバの探索を使用できない．DNS データベースの更新遅延は，数十秒から数分という短時間での複製サーバの再配置に比べて長いためである．複製サーバへのアクセス透過性を提供するために，EPSR は複製サーバの新しい配置先をクライアントに通知できる仕組みを備える必要がある．

3) **トポロジを考慮した需要発生地域の検出**

需要が発生している地域のホスティングマシンを配置先として決定し通信遅延を低下させるために，EPSR は物理ネットワークの構造を考慮する必要がある．トポロジの構造を活用して，EPSR はネットワークトラフィックを削減できるような位置にあるホスティングマシンを複製サーバの配置先として選択する．複製サーバがクライアントの近傍に配置されていれば，クライアントのアクセス経路が短くなるため通信遅延を低減できる可能性が高い．さらに，もしクライアントのアクセス経路が集中する地点に複製サーバを配置できれば，クライアントリクエストを中継するルータやホストを減らしながら，配置された複製サーバは多くのクライアントを処理することができる．

4) **性能制約と負荷を考慮した複製サーバ数の決定**

性能制約を満たすために，複製サーバ数の配置先を決定するとき EPSR はホスティングマシンの性能と現在の負荷を考慮しなければならない．もしホスティングマシンの性能が低い場合や現在の負荷が高い場合は，複製サーバの配置先を増加させる必要がある．もしホスティングマシンの性能が高い場合や現在の負荷が低い場合は，複製サーバの実行コストを削減するために配置先を減少させるべきである．

また ExaPeer が複数のサービスを並行して提供できるようにするために，EPSR

はホスティングマシン上で動作する複製サーバごとの負荷を個別に考慮できるようすべきである．もしホスティングマシンが複数のサービスの高需要地域内にあった場合，そのマシンはそれぞれの複製サーバを並行して稼働する必要がある．複数の複製サーバを稼働しているとき，ホスティングマシンは過負荷にならないように複製サーバごとに負荷を計測し，複製サーバの配置先をそれぞれ決定する必要がある．

3.2 概要

EPSR は，個々のホスティングマシンが独立して需要変動を検知できる仕組みを提供する．EPSR では需要変動に関する情報の収集や解析を行う特定の中央サーバを必要としない．EPSR は需要変動に関する情報の収集範囲をホスティングマシンの近傍に限定することで，大域情報の収集に要するコストを不要とする．

EPSR はホスティングマシンやクライアントの位置関係を特定するために，物理ネットワークのトポロジを考慮したオーバレイネットワークを構築する．このオーバレイネットワークはクライアントによる複製サーバの探索にも用いる．EPSR が構築するオーバレイネットワークは以下の 2 点の特徴を備える．

- ホスティングマシンは d 次元のネットワーク座標が割り当てられる．ネットワーク座標とは，座標間のユークリッド距離がマシン間のラウンドトリップタイム (RTT) を近似するように割り当てられたユークリッド座標のことである．ネットワーク座標の詳細は第 3.3 節で後述する．
- クライアントが複製サーバを特定するために送信する**サーバ解決要求メッセージ**を，複製サーバが稼働しているホスティングマシンまでオーバレイネットワーク上で転送する．サーバ解決要求メッセージは宛先へ常に近づくように転送される．物理ネットワークのトポロジを考慮した DHT を利用すれば，この条件を満たすオーバレイネットワークを構築できる．

EPSR によってホスティングマシンは，サービスごとに需要が発生している地域の方角を認識できるようになる．EPSR が構築するオーバレイネットワークはサーバ解決要求メッセージが常に宛先へ近づくように転送することを保証するので，ホスティングマシンはサーバ解決要求メッセージが転送されてくる方角に需要の発生地域が存在すると推測することができる．大量のサーバ解決要求メッセージが

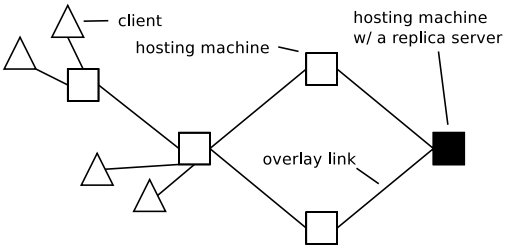
一方向から転送されてくると，EPSR はその方角にあるホスティングマシンを需要の発生地域を示す**マーカ**とする．もしマーカとなったホスティングマシンが同様に大量のサーバ解決要求メッセージが一方向から転送されてくることを認識すると，そのマーカはさらにメッセージが転送されてくる方角のホスティングマシンをマーカとする．これを繰り返すことでマーカが需要の発生地域へ近づいていき，EPSR が需要の発生地域を特定することができる．

マーカによって需要の発生地域を特定すると，EPSR はその地域の需要に見合うホスティングマシンを選び出し複製サーバの配置先とする．マーカとその近傍にあるホスティングマシンは，クライアントによる現在の負荷がマシンの性能制約を超えないように，周辺のホスティングマシンを複製サーバの配置先とする．あるホスティングマシンが負荷の増加によって性能制約の限界に達したとき，ホスティングマシンは近傍のホスティングマシンのうち，最も多くのクライアントリクエストを転送してくるものを新たな配置先とする．もし選ばれたホスティングマシンが既に性能限界に達していたときは，そのホスティングマシンはさらにサーバ解決要求メッセージが転送されてくる方角のホスティングマシンを配置先とする．これを繰り返すことで，EPSR はクライアントとの通信遅延を低下させ，かつホスティングマシンの性能制約を満たすように複製サーバの配置先を決定する．

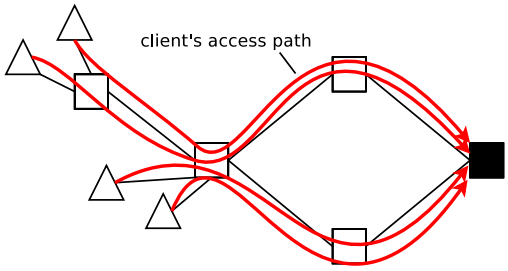
図 3.1 に EPSR の動作の概念図を示す．まず，物理ネットワークのトポロジを考慮したオーバレイネットワークを構築する（図 3.1 (a)）．クライアントは接続すべき複製サーバを探索するために，最寄りのホスティングマシンにサーバ解決要求メッセージを送信する．ホスティングマシンはオーバレイネットワーク上でサーバ解決要求メッセージを複製サーバまで転送する（図 3.1 (b)）．このとき，サーバ解決要求メッセージが多く転送されているホスティングマシンは，周辺に需要が発生しているとみなし，複製サーバの配置先とする（図 3.1 (c)）．以降に送信されたサーバ解決要求メッセージは，より近い複製サーバに到達する（図 3.1 (d)）．複製サーバを稼働するホスティングマシンの負荷が性能制約に到達した場合，さらにクライアントに近いホスティングマシンに複製サーバを配置する（図 3.1 (e)）．

3.3 ネットワーク座標

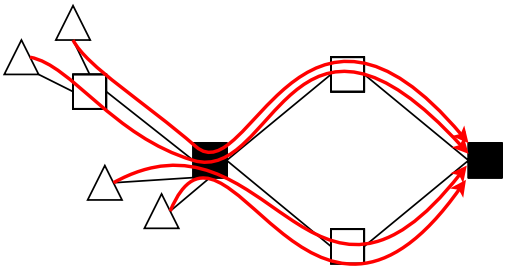
本節では，EPSR の要素技術であるネットワーク座標について説明する．ネットワーク座標はネットワークにおけるホスト間の RTT をユークリッド距離で近似す



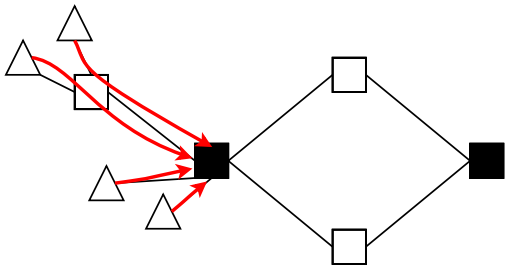
(a) 物理ネットワークのトポロジを考慮したオーバレイネットワークを構築する．



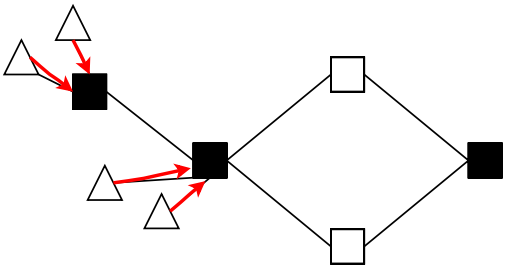
(b) ホスティングマシンはクライアントが送信したサーバ解決要求メッセージを複製サーバまで転送する．



(c) メッセージが多く転送されているホスティングマシンを複製サーバの配置先とする．



(d) 以降に送信されたサーバ解決要求メッセージは，より近い複製サーバに到達する．



(e) ホスティングマシンの負荷が性能制約に達すると，さらにクライアントに近いホスティングマシンに複製サーバを配置する．

図 3.1: **EPSR の動作概念図．**

るように計算されたユークリッド座標である．ネットワーク座標によって，インターネット上のホストは互いの座標から通信遅延を推測することができる．

ネットワーク座標を生成するシステムはネットワーク座標系と呼ばれ，様々なものが提案されている [69–77]．ネットワーク座標系は各ホストに割り当てるネットワーク座標を，座標から計算されるユークリッド距離がホスト間の RTT になるように発見的アルゴリズムを用いて決定する．ネットワーク座標系は 2 種類に大別される．ひとつは，あらかじめ用意したランドマークホストから d 次元の座標空間を構築しておき，一般ホストの座標をランドマークとの RTT とユークリッド距離の誤差を最小化する最適化問題から求める中央処理型の方法 [69–71] である．もうひとつは，ランドマークホストを用意せずに個々のホスト間の RTT とユークリッド距離の誤差を最小化する最適化問題で求める完全分散型の方法 [72–77] である．適切な次元数 d はネットワーク座標系によって異なるが，通常は 6 から 8 次元を用いる．これは RTT を距離としたインターネット空間は複雑な位置関係となっており，低次元では三角不等式を満たさない座標になり精度が低下するからである．ネットワーク座標系の精度は相対誤差が 0.2 程度と文献 [77] で報告されている．

ここでは Global Network Positioning (GNP) [69] を例に挙げて，ネットワーク座標の計算手続きを説明する．GNP はランドマークのネットワーク座標を求める手続きと，一般ホストのネットワーク座標を求める手続きの 2 段階を経る．まず，N 台のランドマークのネットワーク座標 $c^S_{L_1}, \ldots, c^S_{L_N}$ は次の目的関数 $f_{obj1}(\cdot)$ を最小化する最適化問題を解くことで求める．

$$f_{obj1}(c^S_{L_1}, \ldots, c^S_{L_N}) = \sum_{L_i, L_j \in \{L_1, \ldots, L_N\} | i>j} \varepsilon(d_{L_i L_j}, \hat{d}^S_{L_i L_j}) \tag{3.1}$$

ここで $c^S_{L_i}$ はネットワーク座標空間 S におけるランドマーク L_i のネットワーク座標，$d_{L_i L_j}$ はランドマーク L_i と L_j の RTT，$\hat{d}_{L_i L_j}$ はネットワーク座標から求めたランドマーク L_i と L_j のユークリッド距離，$\varepsilon(\cdot)$ は誤差関数である．誤差関数 $\varepsilon(\cdot)$ は例えば次のような単純な二乗誤差を用いることで定義できる．

$$\varepsilon(d_{H_1 H_2}, \hat{d}^S_{H_1 H_2}) = (d_{H_1 H_2} - \hat{d}^S_{H_1 H_2})^2 \tag{3.2}$$

この最適化問題は発見的アルゴリズムで近似解を求めることができる．GNP によるネットワーク座標は，例えば滑降シンプレックス法 [111] で求められることが知られている．

次に一般ホストのネットワーク座標を求める．ホスト H のネットワーク座標 c_H^S は次の目的関数 $f_{obj2}(\cdot)$ を最小化する最適化問題を解くことで求める．

$$f_{obj2}(c_H^S) = \sum_{L_i \in \{L_1,\ldots,L_N\}} \varepsilon(d_{HL_i}, \hat{d}_{HL_i}^S) \tag{3.3}$$

ここで $\varepsilon(\cdot)$ はランドマークのネットワーク座標を求める際と同様の誤差関数である．一般ホストのネットワーク座標の場合もランドマークのネットワーク座標の場合と同様に，発見的アルゴリズムを用いて近似解を求められる．

3.4 EPSR オーバレイネットワーク

需要が発生している地域の方角を特定するために，EPSR はオーバレイネットワークを GNP と Content-Addressable Network (CAN) [79] を組み合わせることで構築する．GNP は第 3.3 節で述べた通り，ネットワーク座標系のひとつである．CAN は DHT のひとつである．CAN は個々のホスティングマシンに仮想的な d 次元の座標を与え，また d 次元空間をゾーンと呼ばれる単位に分割する．各ゾーンはホスティングマシンが 1 台ずつ割り当てられるように分割される．ホスティングマシンは自身が管理するゾーンが座標 $(h_0(K), \ldots, h_{d-1}(K))$ を含む場合，(key K, value V) の組を自身に保存する．ここで h_i はホスティングマシン間で共有するハッシュ関数である．もし key K が存在するとき，CAN は K を管理するホスティングマシンまでの到達ホップ数の平均を $\frac{d}{4}N^{\frac{1}{d}}$ とすることを保証する．ここで N は CAN に参加しているホスティングマシンの数である．

GNP と CAN を用いることで，EPSR は物理ネットワークのトポロジを考慮したオーバレイネットワークを構築する．EPSR では，d 次元の CAN で割り当てる座標を GNP によって生成した d 次元のネットワーク座標とする．これにより，EPSR オーバレイネットワーク上で隣接するホスティングマシンは物理ネットワーク上でも近傍にあることが保証される．ExaPeer にサービスが登録されると，EPSR は**ルートマシン**と呼ばれるホスティングマシンを選び出す．ルートマシンはサービスの識別子[1]を key として，CAN のプロトコルに従って決定する．なお，EPSR は GNP と CAN に依存するものではない．例えば，EPSR は GNP の代わりに他のネットワーク座標系である SCoLE [73] や Vivaldi [75] を用いることができる．また，CAN の代わりに eCAN [112] を用いて構築することも可能である．

[1]サービス名などの一意に定められるものを使用する．

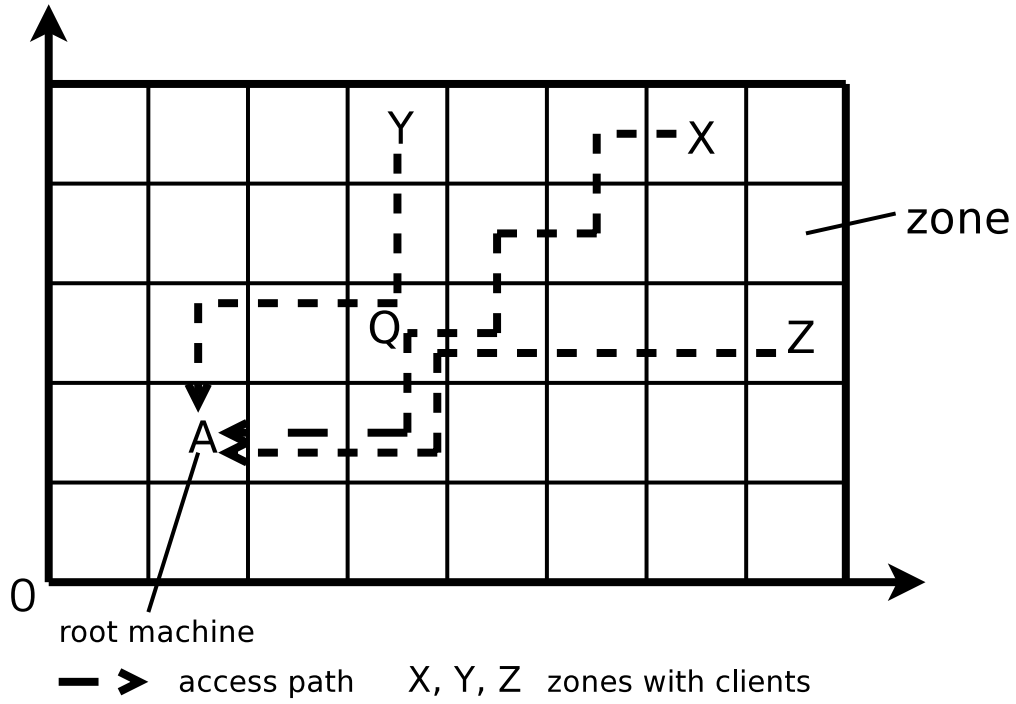


図 3.2: **APC degree に基づいたマーカの設定．**

クライアントは自身が接続すべき複製サーバを特定するために，サーバ解決要求メッセージをクライアント自身が含まれるゾーンを管理するホスティングマシンに送信する．サーバ解決要求メッセージは，メッセージの種類を示す 4 ビット値，クライアントの IP アドレス，サービス名から変換して求めた d 次元の宛先座標およびクライアントからのメッセージを最初に受信したホスティングマシンの IP アドレスを含む．サーバ解決要求メッセージは CAN のプロトコルに従い，複製サーバかルートマシンに到達するまでホスティングマシンによって転送される．

クライアントがサーバ解決要求メッセージの送信先とするホスティングマシンを特定する手順は以下の通りである．まず，クライアント X は DNS ラウンドロビンなどの手段で ExaPeer に参加している任意のホスティングマシン M を特定し，**ゾーン特定用メッセージ**を送信する．ゾーン特定用メッセージは，メッセージの種類を示す 4 ビット値，クライアントの IP アドレスと GNP 座標，およびクライアントが最初に送信したホスティングマシンの IP アドレスを含む．ホスティングマシンはクライアントの GNP 座標を宛先としてゾーン特定用メッセージを転送する．宛先座標を含むゾーンを管理するホスティングマシン N は，ホスティングマシン M を経由して自身の IP アドレスをクライアント X に通知する．

図 3.2 は EPSR が 2 次元の CAN を用いて構築したオーバレイネットワークを示す[2]．それぞれのゾーンにはひとつずつホスティングマシンがある．この図では登録したサービスのルートマシンがホスティングマシン A となっている．クライアント X は登録されているサービスを受けるために，自身が含まれるゾーンを管理するホスティングマシンにサーバ解決要求メッセージを送信する．送信されたメッセージは CAN のプロトコルに従ってルートマシン A まで転送される．クライアント Y および Z に関しても同様である．

3.5 需要発生地域の特定

需要が発生している地域を特定するために，EPSR は需要の発生地域を示す**マーカ**と呼ばれるホスティングマシンを選び出す．マーカの周辺にあるホスティングマシンを複製サーバの配置先とすれば，需要の発生している地域に複製サーバを配置することができる．本節では EPSR が需要の発生している地域の位置を特定するメカニズムについて述べる．

EPSR は**アクセス経路の集中度** (*APC degree*) に従ってマーカとなるホスティングマシンを選択する．アクセス経路とは，クライアントが送信するサーバ解決要求メッセージがサーバまで転送される際に通過したオーバレイネットワーク上の経路を指す．APC degree はホスティングマシン上を通過するアクセス経路の数である．ホスティングマシンは APC degree をサービスごとに記録し管理する．例えばホスティングマシン Q がサービス S のサーバ解決要求メッセージを転送したとき，Q はサービス S の APC degree を加算する．図 3.2 の例では，ホスティングマシン Q はクライアント X，Y，Z のサーバ解決要求メッセージの転送経路であるため，Q の APC degree は 3 となる．

APC degree を計測するために，個々のホスティングマシンはサービスごとに，他のマシンへ転送したサーバ解決要求メッセージの数を記録する．もしある地域で需要が増加していると，転送されるサーバ解決要求メッセージの数もアクセス経路に沿って増加する．APC degree が事前に設定した閾値 $upper$ に達すると，ホスティングマシンはマーカとなる．需要が減少することで APC degree が別の閾値 $lower$ を下回ると，マーカとなっているホスティングマシンは一般のホスティング

[2] ここでは説明のため 2 次元空間を用いているが，実際は第 3.3 節で述べたように精度の問題からより高い次元を用いる．また，一般にゾーンの大きさは不均一である．

マシンに戻る．図 3.2 の例で $upper$ が 3 に設定されている場合，ホスティングマシン Q は APC degree が 3 であるのでマーカとなる．

周囲のホスティングマシンの APC degree を知るために，ホスティングマシンは任意のタイミングで **APC degree 要求メッセージ**を隣接するホスティングマシンに送信する．APC degree 要求メッセージは，メッセージの種類を示す 4 ビット値を含む．APC degree 要求メッセージを受け取ったホスティングマシンは直ちに **APC degree 通知メッセージ**を送信元に送信する．APC degree 通知メッセージはメッセージの種類を示す 4 ビット値と現在の APC degree を示す整数値を含む．

EPSR は一定時間内のサーバ解決要求メッセージの転送数を数え上げるためにスライディングウィンドウを使用する．スライディングウィンドウは 1 秒間のサーバ解決要求メッセージ転送数を指定された秒数分記録する．複製サーバの配置先の数が振動することを避けるために，$upper$ と比較して $lower$ は十分に小さくなければならない．閾値 $upper$ と $lower$ の指定方針については第 5 章で議論する．

マーカとなったホスティングマシンは周囲にマーカであることを通知するために，**マーカ設定通知メッセージ**を隣接するゾーンのホスティングマシンに送信する．マーカ設定通知メッセージは，メッセージの種類を示す 4 ビット値とサービスを示す座標値を含む．マーカ設定通知メッセージを受け取ったホスティングマシンは送信元のホスティングマシンがマーカであることを記録する．また，マーカを解除するときは**マーカ解除通知メッセージ**を隣接するゾーンのホスティングマシンに送信する．マーカ解除通知メッセージは，メッセージの種類を示す 4 ビット値とサービスを示す座標値を含む．マーカ解除通知メッセージを受け取ったホスティングマシンは送信元のホスティングマシンがマーカでないことを記録する．

3.5.1 基本メカニズムの改良

EPSR の基本メカニズムは，状況によっては需要発生地域の特定とマーカの選定に失敗することがある．図 3.3 は EPSR の基本メカニズムが需要発生地域の特定に失敗する例である．図 3.3 の例では，3 台のクライアント X, Y, Z がルートマシン A にアクセスしている．注意すべきは，この図が示す例は図 3.2 の例とクライアントの配置は同じであるが，アクセス経路が図 3.2 の例に比べてより分散していることである．その結果，図 3.2 とクライアントの配置は同じであるにも関わらず，どのホスティングマシンも需要の発生を認識することができない．

EPSR の基本メカニズムが需要発生地域の特定に失敗する原因は 2 点ある．ひ

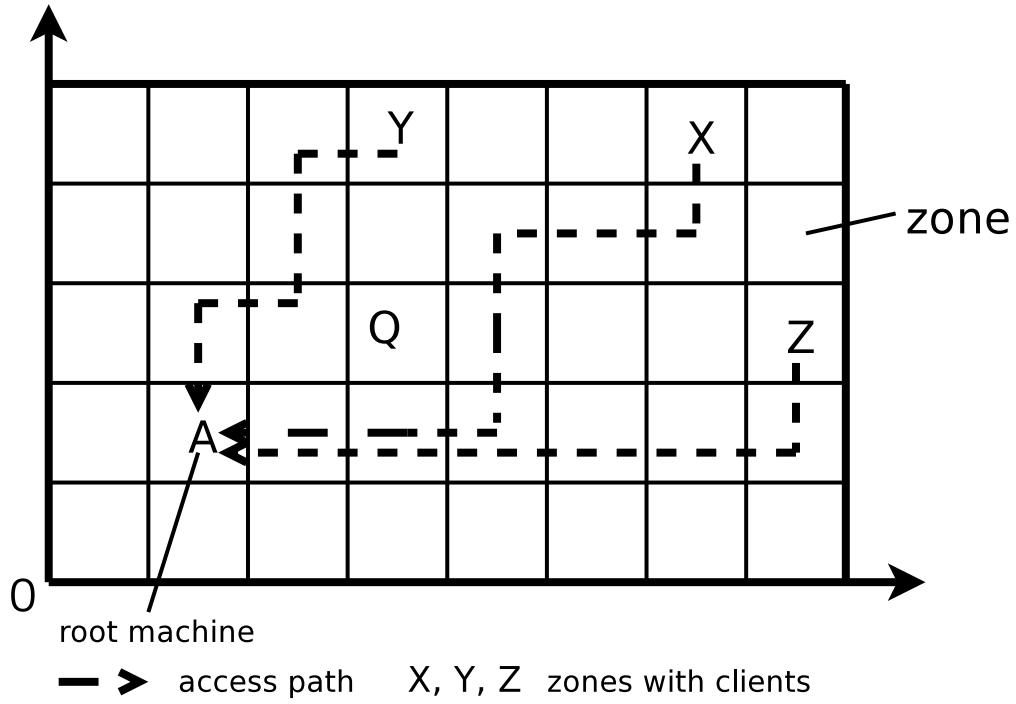


図 3.3: **基本メカニズムではマーカ設定が失敗する例．**

とつは，CAN は同じクライアントが送信したメッセージでも異なる転送経路をとることがあるためである．これは CAN が信頼性や耐故障性向上のために，転送経路の選択に一定の自由度を持たせているためである．もうひとつは，高い需要の発生している地域がルートマシンに対して広範囲かつ遠方に存在した場合である．このような場合，ルートマシンに対してアクセス経路が様々な方向に分散してしまうため，アクセス経路が特定のマシン上で集中する可能性が減少する．

EPSR はこれらの原因に対処するために 2 つの手法を用いる．第一に挙げた転送経路の揺らぎによる問題に対処するために，EPSR は *Virtual Path Expanding* を用いる．第二に挙げた需要の発生地域の分散による問題に対処するために，EPSR は *Virtual Demand Raising* を用いる．以下ではこれらの手法について説明する．

Virtual Path Expanding

Virtual Path Expanding (VPE) はアクセス経路を仮想的に拡大する．VPE はアクセス経路上にあるホスティングマシンの APC degree の増加に加えて，アクセス経路に隣接するホスティングマシンの APC degree も増加させる．これによりアクセス経路の幅を見かけ上拡大し，アクセス経路が揺らいでも適切なホスティングマ

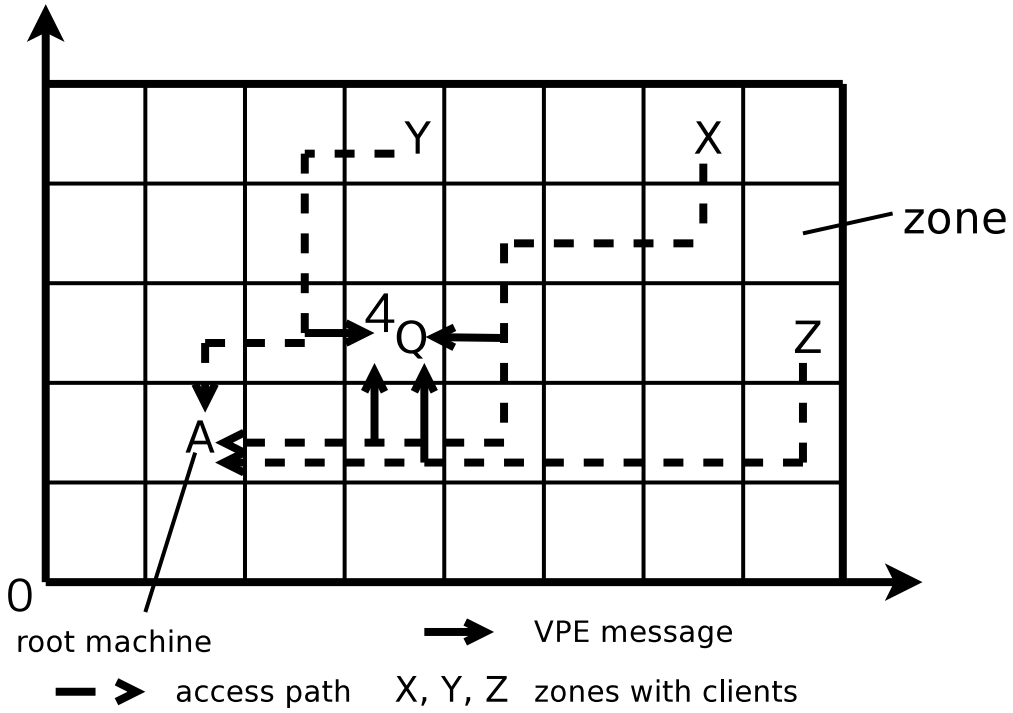


図 3.4: **Virtual Path Expanding (VPE).** VPE によってアクセス経路に隣接するホスティングマシンの APC degree が増加する．

シンの APC degree が上昇しやすくなる．図 3.4 は VPE によってアクセス経路の揺らぎによる問題が解消される様子を示す．図 3.4 の例では，クライアント X, Y, Z のアクセス経路に隣接するホスティングマシンの APC degree を VPE が増加させる．例えば，ホスティングマシン Q は全てのアクセス経路に隣接しているため，Q の APC degree は VPE によって 4 となる．これによりアクセス経路が揺らいでも，VPE によって図 3.2 と同じ条件でホスティングマシン Q がクライアント X, Y, Z による需要の発生を正しく認識できるようになる．

VPE を適用した場合，アクセス経路に隣接するホスティングマシンは**VPE メッセージ**を隣接するホスティングマシンに送信する．VPE メッセージは，メッセージの種類を示す 4 ビット値のみを含む．VPE メッセージを受け取ったホスティングマシンは該当するサービスの APC degree を 1 増加させる．この操作を複製サーバを稼働しているホスティングマシンに到達するまで繰り返す[3]．VPE メッセージは隣接するホスティングマシンのみに送信するため，各ホスティングマシンが送信する VPE メッセージの数は $O(2d)$ となる．VPE メッセージによる通信負荷の

[3]複製サーバを稼働するホスティングマシン上では VPE は働かない．

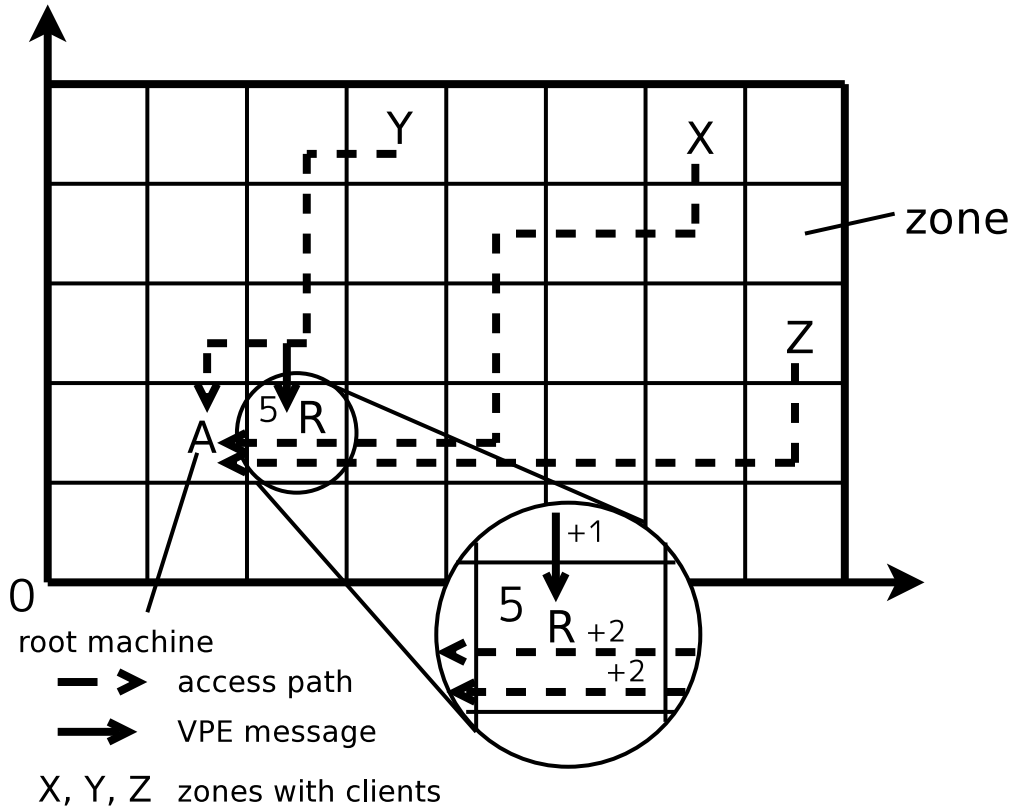


図 3.5: **Virtual Demand Raising (VDR).** 複製サーバが稼働中のホスティングマシンに隣接するホスティングマシンは，アクセス経路を 2 倍して数え上げる．

増加量は第 5 章で議論する．

Virtual Demand Raising

ルートマシンに到達するアクセス経路が多方向に分散している場合，VPE を適用しても APC degree が十分に増加せず，需要の増加した地域を効率よく特定することができない．需要が発生している地域の方向をより積極的に検知し精度よく特定するために，EPSR は Virtual Demand Raising (VDR) を用いる．VDR によって，アクセス経路上にあるホスティングマシンはよりマーカになりやすくなる．VDR はルートマシンもしくはマーカであるホスティングマシンに隣接するマシンにおける APC degree の加算量を，他のホスティングマシンの 2 倍にする．これにより，需要の発生している地域の方角にあるホスティングマシンの APC degree の増加率が他のホスティングマシンに比べて大きくなる．

図 3.5 は，VDR によって需要が発生している地域の方角にあるホスティングマシンがよりマーカになりやすくなる様子を示す．図 3.5 の例では，ホスティングマシン R はルートマシン A に隣接しており，クライアント X と Z のアクセス経路上にある．ホスティングマシン R の APC degree は VDR 適用前の 3 から 5 $(= 2 + 2 + 1)$ となる[4]．

3.6 複製サーバの配置先の決定

EPSR はマーカによって需要の発生地域を特定した後，需要の規模に応じてマーカの近傍にあるホスティングマシンを複製サーバの配置先とする．本節では EPSR が需要の発生している地域の位置を特定したのちに，需要の規模とホスティングマシンの性能制約に応じて，需要発生地域のホスティングマシンを複製サーバの配置先として選択するメカニズムについて述べる．

単純な手段として，マーカとなったホスティングマシンをそのまま複製サーバの配置先とする方法がある．しかし，この方法は性能制約を満たさない．これは，マーカは近傍に需要が発生していることを示すが，マーカの数が需要の規模を示すわけではないからである．言い換えれば，マーカの数が必要十分な複製サーバの数に達しているとは限らない．もし EPSR がマーカとなっているホスティングマシンをそのまま複製サーバの配置先としても，複製サーバの数や配置先となったホスティングマシンの選択が需要に見合ったものとなっているとは限らない．これはマーカが需要の規模に関係なく，需要の発生している地域の位置に応じて選択されるからである．第 4 章で後述するシミュレーションでは，マーカをそのまま複製サーバの配置先とした場合，複製サーバの配置先となったホスティングマシンの 25 パーセント以上が性能制約を超過して過負荷となった．このとき過負荷となったマシンの近傍のホスティングマシンは，複製サーバの配置先となるのに十分な性能を残していた．

マーカを複製サーバの配置先とする単純な方法が失敗する理由は，APC degree がホスティングマシンの負荷や性能制約を示す指標ではないことにある．APC degree は需要発生地域に近づく方向にあるホスティングマシン上で高い値を示すため，需要の発生地域の方角を示す指標として十分である．しかし，APC degree はホスティングマシンの負荷や性能制約によって値が変化することはない．例えば，マーカ

[4]ホスティングマシン R はクライアント Y のアクセス経路に隣接しているため，VPE によって 1 加算される．

となったホスティングマシンの余剰性能が十分にある場合は，そのホスティングマシンで複製サーバを稼働させることで近傍のクライアントへサービスを提供することができる．反対に性能が不足している場合は，マーカは近傍のホスティングマシンを新たな配置先として，複製サーバの数を増加させる必要がある．したがって EPSR は，現在の需要の規模に見合うだけの複製サーバが配置されているかどうか判断できる指標をホスティングマシンへ提供する必要がある．

需要の規模に見合うように正しく複製サーバの配置先数を調整できるように，EPSR は複製サーバの現在の負荷を反映する *Load indicator* を導入する． Load indicator は現在接続しているクライアントによって生じる複製サーバの負荷を示す抽象的な値である．ここでいう接続とは，クライアントが複製サーバを特定した後に，サービスを受けるためにクライアントが複製サーバと直接確立するものを指す．本論文の実験では Load indicator として同時接続数を使用した．これは同時接続数がいくつかの市販製品で負荷を示す指標として使用されているからである．例えば，負荷分散装置である LocalDirector [113] や BIG-IP [114] は負荷を示す指標として同時接続数を採用している．ただし，EPSR は同時接続数以外の値を Load indicator として使用することができる．例えば，CPU 負荷やメモリ使用量，使用ネットワーク帯域などや，それらを独立変数とする関数値を使用することができる．

Flash Crowd による影響を軽減するために，EPSR は複製サーバの配置先の調整をホスティングマシンによる自律的な調整によって実現する．EPSR はホスティングマシンに自身の Load indicator を監視させ，複製サーバの配置先を周囲に増加させるかどうか自律的に判断させる．EPSR は物理ネットワークのトポロジを考慮したオーバレイネットワークを構築しているため，物理ネットワーク上で近い位置にいるクライアントが送信するサーバ解決要求メッセージは同じホスティングマシンに到達する傾向がある．よって，EPSR は高負荷になっている複製サーバの周辺のホスティングマシンを新たな配置先とすればよい．このようにすれば，ExaPeer は複雑な手続きを必要とせずに，新たに配置された複製サーバへ負荷を分散することができる．

EPSR は以下の手続きによって複製サーバの配置先を決定する．まず，マーカとなったホスティングマシンは複製サーバの初期配置先となる．複製サーバの Load indicator が上昇すると，その複製サーバを実行しているホスティングマシンは隣接する他のホスティングマシンに新たな配置先となるよう依頼する．新たな配置先で複製サーバが起動した後は，高負荷となっているホスティングマシンは以降

のサーバ解決要求メッセージを自身で処理せず他の複製サーバへ転送する．高負荷になっているホスティングマシンは，自身に接続しているクライアントが減少して負荷が下がるまでこの転送を継続する．

ホスティングマシンは複製サーバの起動を依頼する際に**複製サーバ起動依頼メッセージ**を送信する．複製サーバ起動依頼メッセージは，メッセージの種類を示す 4 ビット値，自身の GNP 座標から隣接するゾーンの中心点への方向ベクトル，およびサービスの識別子から計算された座標値を含む．

EPSR は性能制約を満たすために，どのホスティングマシンも過負荷にならないように複製サーバの配置先を増加させる．先に述べたように，高負荷になっているホスティングマシンは隣接するホスティングマシンに複製サーバ起動依頼メッセージを送信する．もし依頼メッセージを受信したホスティングマシンが既に高負荷であり新たに複製サーバを稼働するだけの余剰性能が残っていない場合，そのホスティングマシンはさらに隣接するマシンのひとつにその依頼メッセージを転送する．依頼メッセージが循環しないように，依頼メッセージは送信元のホスティングマシンが最初に送信した隣接マシンの方向へ転送される．

クライアントが通信遅延の小さい複製サーバへ接続できるようにするために，EPSR はクライアントのサーバ解決要求メッセージが多く転送されるホスティングマシンを優先して複製サーバの配置先とする．これを実現するために，EPSR は配置先とするホスティングマシンを選択する際に APC degree を利用する．第 3.5 節で述べたように，APC degree が高いホスティングマシンは多くのサーバ解決要求メッセージを他のホスティングマシンへ転送している．またサーバ解決要求メッセージはクライアントとの通信遅延の小さい複製サーバへ転送されやすいことが，物理ネットワークのトポロジを反映したオーバレイネットワークによって保たれる．そのため，APC degree の高いホスティングマシンへ複製サーバを配置すれば，配置された複製サーバはより多くのクライアントへ低い通信遅延でサービスを提供できる．

また，APC degree の高いホスティングマシンを複製サーバの配置先とすることは，クライアントを低負荷の複製サーバへ接続しやすくする．APC degree の高いホスティングマシンは多くのクライアントのアクセス経路上にあるため，そのようなホスティングマシン上の複製サーバは多くのクライアントに発見されやすくなる．

図 3.6 および 図 3.7 は EPSR が複製サーバの配置先を決定する例を示す．この例では，すべてのホスティングマシンは同時に 3 台までのクライアントに対して

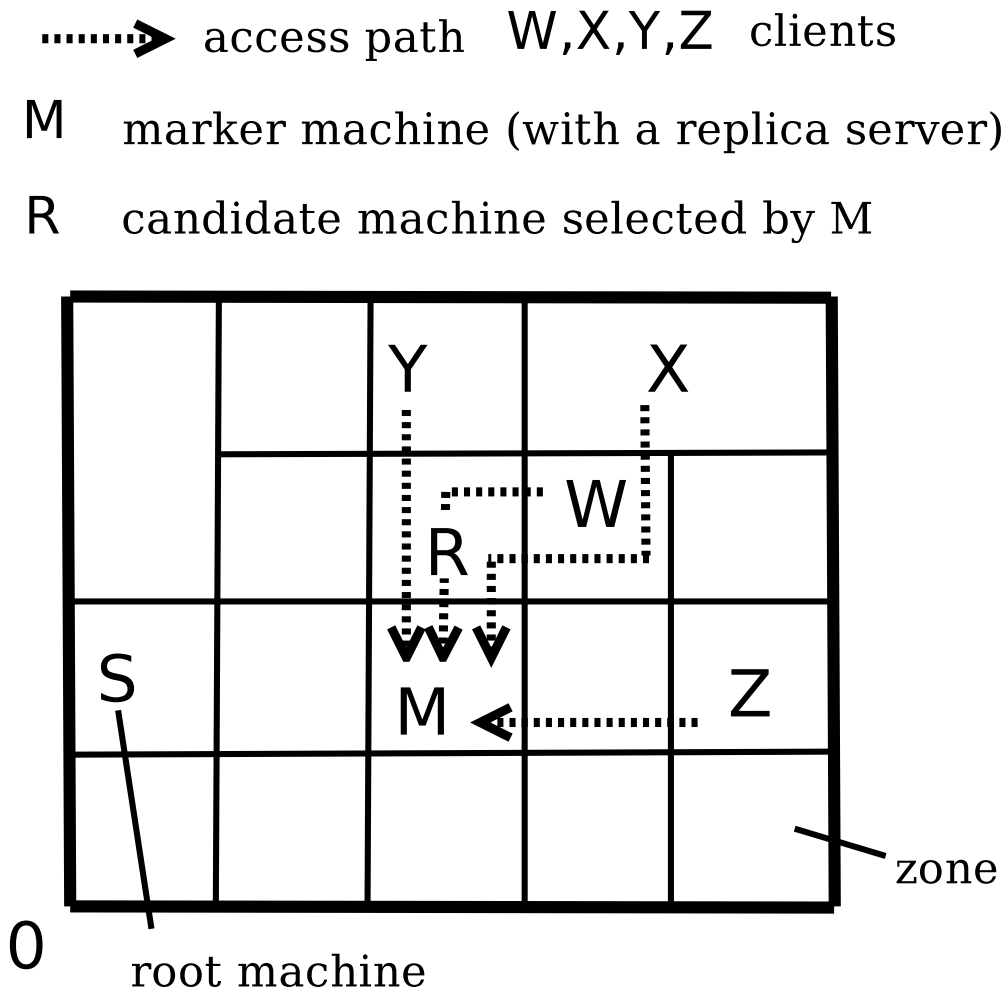


図 3.6: **マーカとなったホスティングマシンが過負荷となる例．**ホスティングマシン M の複製サーバが全てのクライアントを処理するため，性能制約を超過し過負荷となる.（ホスティングマシンの性能制約をクライアント 3 台とした場合.）

サービスが提供できるものとする．M は，クライアント W, X, Y, Z によって発生している需要の地域を示すマーカである．クライアントは 4 台であるため，M だけが複製サーバを稼働した場合，M が過負荷となる．そこで M は隣接するホスティングマシンから最も APC degree の大きいホスティングマシン R を選び出し，新たな複製サーバの配置先となるよう R に依頼する（図 3.6）．R は複製サーバを起動し，クライアント W, X, Y の接続を受け付け，サービスの提供を開始する（図 3.7）．もし依頼したホスティングマシンが既に高負荷であった場合，依頼メッセージを受信したホスティングマシンは他のホスティングマシンへメッセー

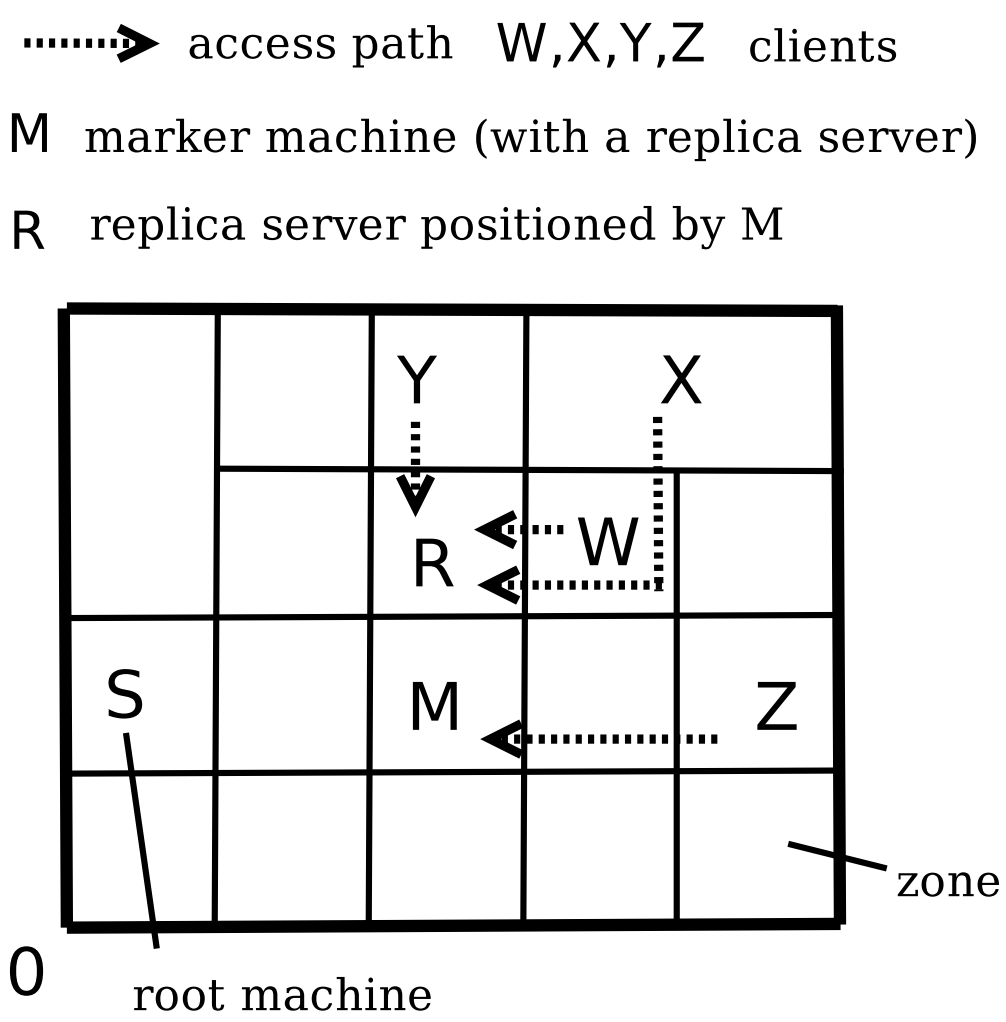


図 3.7: **APC degree と Load indicator に基づいた複製サーバ数の増加．**ホスティングマシン M は隣接するホスティングマシンの中で APC degree がもっとも高い R に複製サーバの起動を依頼する．

ジを転送する．図 3.7 では，もし R が既に高負荷であった場合，R は M から送られてきた依頼メッセージを Y にあるホスティングマシンへ転送する．

余分な複製サーバの数を減らすため，ホスティングマシンは一定時間クライアントの接続がないと複製サーバを停止する．ただし，複製サーバの配置数や位置の振動を低減するために，ホスティングマシンは複製サーバの稼働開始から一定時間はクライアントの接続数が 0 であっても停止しない．

複製サーバを停止する際，ホスティングマシンは**複製サーバ停止通知メッセージ**を隣接するゾーンのホスティングマシンに送信する．複製サーバ停止通知メッ

セージは，メッセージの種類を示す 4 ビット値とサービスの識別子から計算された座標値を含む．複製サーバ停止通知メッセージを受け取ったホスティングマシンは該当する複製サーバがそのゾーンで起動していないことを認識する．

3.6.1 基本メカニズムの改良

EPSR の基本メカニズムは多くの場合で複製サーバの適切な配置先を決定できるが，一方で配置先の決定に失敗する場合がある．EPSR が複製サーバの配置先の決定に失敗する場合は，1) マーカの複製サーバと他の複製サーバで負荷分散をした場合，2) 短時間に多数のクライアントが一斉にサーバ解決要求メッセージを送信した場合，3) ホスティングマシンが複数の需要が発生している地域にある場合の 3 つがある．以下ではこの 3 つの場合について，EPSR が複製サーバの配置先を適切に決定できるようにする手法を説明する．

Dummy Query

EPSR は需要が発生している地域を示すマーカの周辺から複製サーバの配置先を選択する．複製サーバが起動した後は，クライアントが新たに起動した複製サーバに振り分けられるため，元々存在したマーカへのクライアントのサーバ解決要求メッセージの転送は減少する．このことから，マーカの APC degree が減少するために，ホスティングマシンが周辺の需要を実際よりも低く見積もってしまうことがある．このような状況になると，EPSR は需要が消失したと誤認するため，実際は高負荷であるにもかかわらず複製サーバの配置先が減少する恐れがある．このことはまた，複製サーバ数の振動を大きくする．

図 3.8 はこのような状況となる例を示す．この例では，マーカであるホスティングマシン M 上の複製サーバの負荷を分散するために，M によってホスティングマシン R が新たな複製サーバの配置先として決定されている．R で複製サーバが起動すると，R はクライアント V, W, X の，M はクライアント Y, Z の接続を受けることになる．このような状況になると M の APC degree が減少するため，M は周辺の需要が減少したと誤認しマーカから通常の状態に移行してしまう．実際は周辺に高い需要が存在するため，期待された動作とは異なる．

マーカによる需要減少の誤認の問題を解決するために，EPSR では複製サーバがクライアントと接続する際に，その複製サーバを稼働するホスティングマシンは

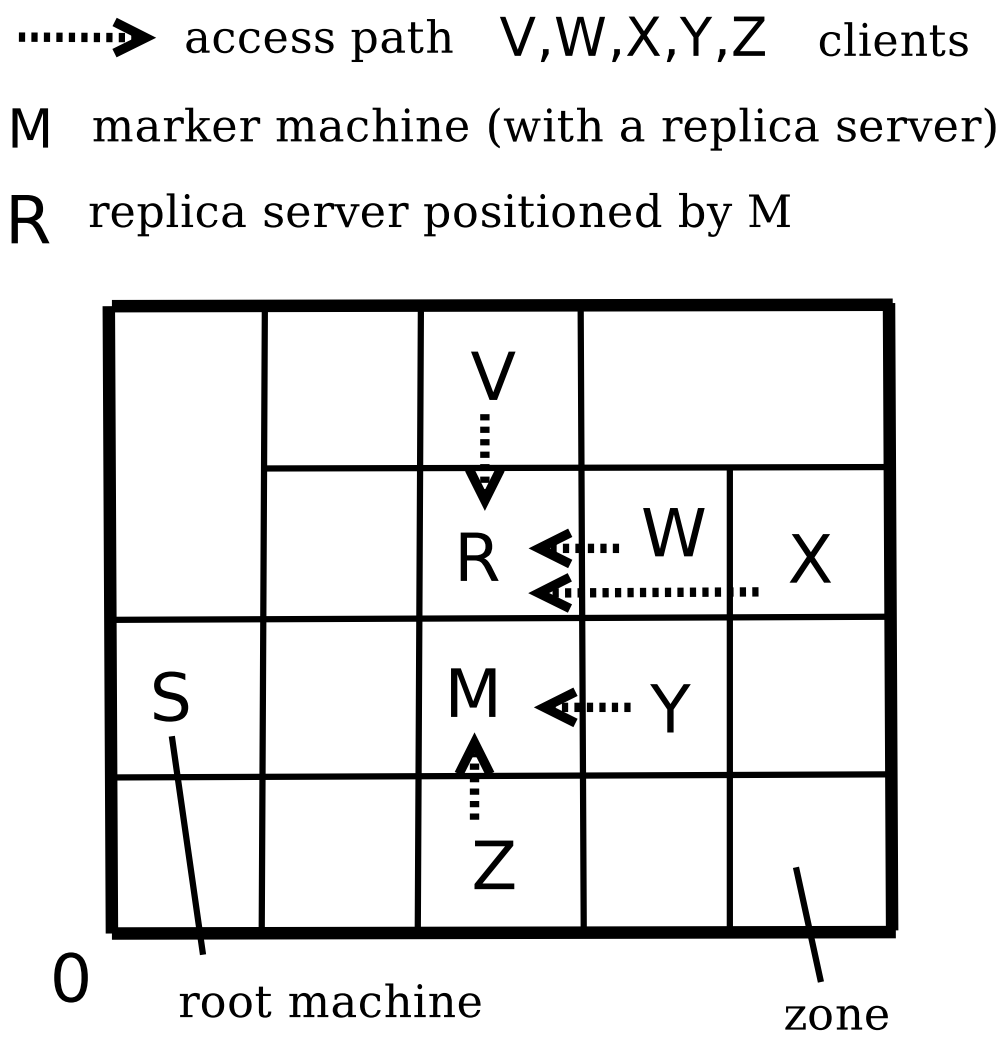


図 3.8: **複製サーバ数が誤って減少しうる例．**ホスティングマシン R が M の代わりにクライアント V，W および X を引き受けるため，M の APC degree が減少する．このため M は需要があるにもかかわらず複製サーバを停止しうる．M が複製サーバを停止すると，もしクライアント Y および Z が再びサービスを要求した場合，R に接続され R が過負荷となる．

マーカに向けて *Dummy Query* を送信する．Dummy Query はメッセージの種類を示す 4 ビット値と，その複製サーバが提供するサービスの識別子から計算される宛先座標を含む．Dummy Query は CAN のプロトコルに従って転送される．この際，Dummy Query は転送経路上にあるホスティングマシンの APC degree を，クライアントのリクエストと同様に増加させる[5]．Dummy Query は APC degree を増

[5]サーバ解決要求メッセージと同様に，Dummy Query も Virtual Path Expanding や Virtual Demand

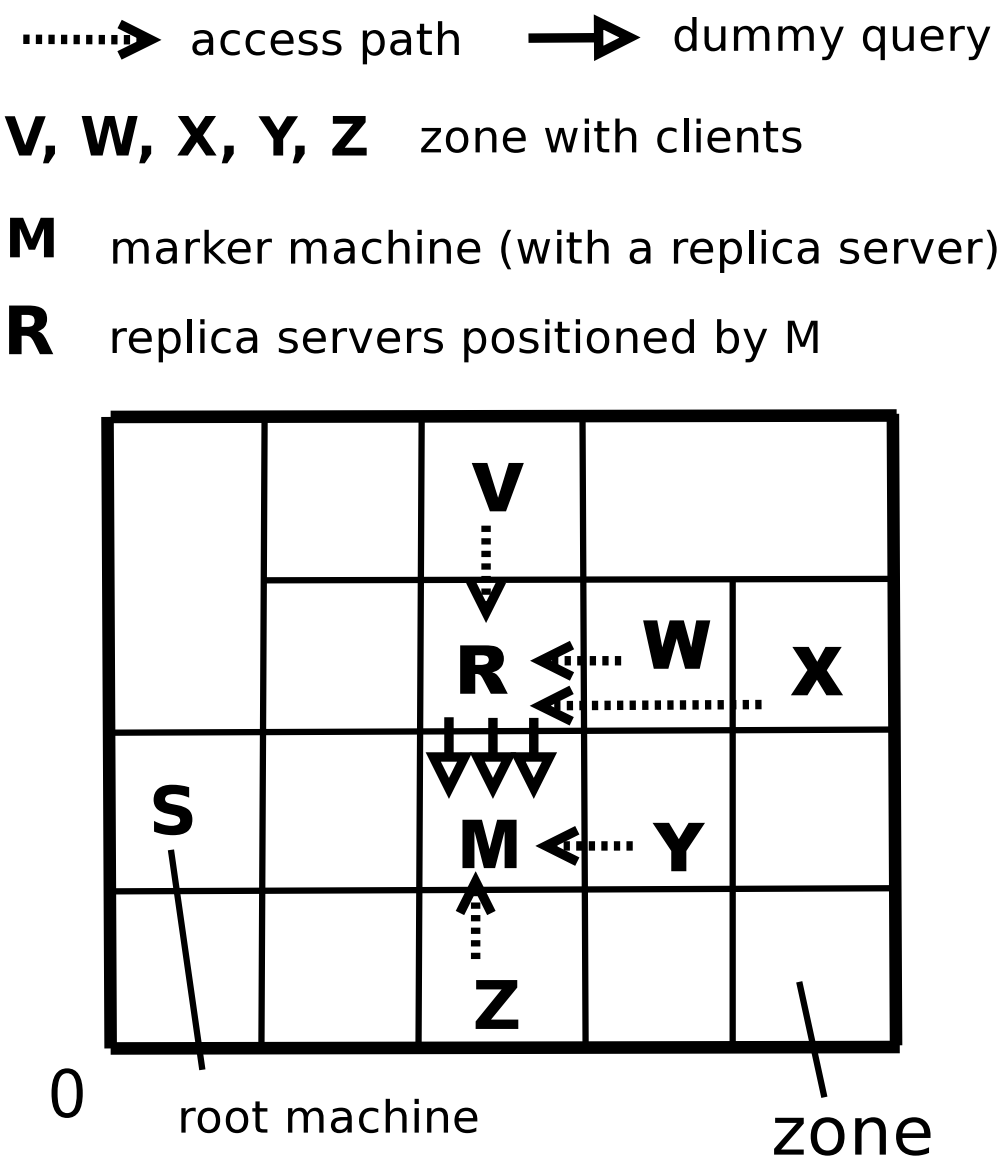


図 3.9: **Dummy Query によって複製サーバ数が需要の規模に適した数に保たれる例．**ホスティングマシン M の APC degree が見かけ上高くなるため，M は R に複製サーバが起動しても引き続きマーカとなり複製サーバを稼働する．これにより複製サーバ数が正しく保たれる．

加させる以外のことは行わない．

図 3.9 は Dummy Query を有効にした際の EPSR の動作を示す．図 3.9 の例では，ホスティングマシン R がマーカであるホスティングマシン M に対して Dummy

Raising は有効である．

Query を送信している．クライアントは R および M 上の複製サーバに振り分けられ負荷分散が行われているが，Dummy Query によってマーカ M の APC degree は複製サーバ R が存在しない場合と同じだけ上昇する．Dummy Query によってホスティングマシン M は複製サーバ R による負荷分散に関わらずマーカとして存在し続けられる．これにより，EPSR はクライアント V, W, X, Y および Z によって発生している需要を正しく認識できる．この状態では図 3.8 の例のような，複製サーバの配置先の振動は起こらない．

Dummy Connection

多数のクライアントが一斉に同じサービスに対してサーバ解決要求メッセージを送信した場合，EPSR の基本メカニズムではホスティングマシンが性能制約以上のクライアントを誤って受信し過負荷になってしまうことがある．これは複製サーバの負荷上昇には遅延があるためである．クライアントが複製サーバを発見した後，クライアントがサービス取得のために複製サーバに接続するまではその複製サーバの負荷は上昇しない．そのためクライアントが同時に同じサービスに対してサーバ解決要求メッセージを送信した場合，ホスティングマシンは性能制約を超えるにも関わらずその要求メッセージを許容してしまう．これはサーバ解決要求メッセージを許容するか否かを判断する時点ではクライアントは複製サーバに接続しておらず，ホスティングマシンの余剰性能が残っているからである．この場合許容したクライアントが複製サーバへ接続するとホスティングマシンが過負荷となってしまう．

この問題を解決するために EPSR は，ホスティングマシンがクライアントのサーバ解決要求メッセージを許容する際に，*Dummy Connection* を複製サーバに対して確立する．Dummy Connection はクライアントが接続を確立した後に生じると予測される仮想的な負荷を複製サーバに対して与え，複製サーバの負荷を仮想的に上昇させる．これにより，ホスティングマシンは以降に受信するサーバ解決要求メッセージを許容すべきか否かを正しく判断できるようになる．Dummy Connection はクライアントが複製サーバに対して実際に接続を確立するか一定時間が経過すると削除される．Dummy Connection とクライアントとの対応づけはクライアントの IP アドレスを用いて行う．

高負荷な複製サーバの優先配置

ホスティングマシンが複数の需要が発生している地域にある場合，複数の複製サーバを実行することがある．ホスティングマシンが複数の複製サーバを実行し過負荷になったとき，配置先を増加させるべき複製サーバを選ぶ必要がある．最も高負荷な複製サーバの配置先を増加させることが直感的な手法である．しかしEPSR の場合，高負荷な複製サーバの配置先を変更することが，多くのクライアントで複製サーバとの通信遅延を増加させる恐れがある．これは，EPSR ではクライアントが通信遅延の小さい複製サーバに接続するため，高負荷な複製サーバは近傍に多くのクライアントが存在する可能性が高いからである．

そこで EPSR は，通信遅延の低下を期待して**低負荷な**複製サーバの配置先を増やす．EPSR が構築するオーバレイネットワークは通信遅延の小さい複製サーバへクライアントを接続させるため，高負荷な複製サーバは低負荷な複製サーバよりも多くのクライアントが近傍に存在することが期待できる．よって，高負荷な複製サーバにより多くのクライアントを接続させるために，EPSR は周辺のホスティングマシンを配置先として低負荷な複製サーバの数を増やし，高負荷な複製サーバにより多くのハードウェア資源を割り当てるために低負荷な複製サーバを終了する．高負荷な複製サーバは新たに割り当てられたハードウェア資源を用いてより多くのクライアントへサービスを提供する．またこのようにすることで，低負荷な複製サーバの配置先は元のホスティングマシンよりも通信遅延の小さいホスティングマシンが配置先となる可能性がある．

高負荷な複製サーバの優先配置は，複製サーバの提供するサービスが次の 2 つの条件を備えている必要がある．第一に，複製サーバの生じる負荷がクライアント数に応じて増加する種類である場合である．第二に，クライアント 1 接続当たりに生じる負荷が，サービス間で極端に異なることがない場合である．EPSR はこの条件を満たすためにサービスを負荷の特徴に応じて分類して管理する．高負荷な複製サーバの優先配置は同じ種類に分類されたサービス間で行う．

3.7 まとめ

本章では，Flash Crowd による影響を軽減する複製サーバの配置先決定手法Exa-Peer Server Reposition (EPSR) の詳細について述べた．急激に需要が増加しても，EPSR は自動で需要の発生地域の特定と需要の規模の見積もりを行い，複製サー

表 3.1: **EPSR の構成要素と課題の対応．**

	高速性	通信遅延	性能制約
ホスティングマシンによる自律的な需要発生地域の特定	○	○	
物理ネットワークのトポロジを考慮したオーバレイネットワーク		○	
Virtual Path Expanding (VPE)		○	
Virtual Demand Raising (VDR)		○	
ホスティングマシンによる複製サーバの配置先の自律選択	○	○	○
Dummy Query		○	○
Dummy Connection			○
高負荷な複製サーバの優先配置		○	○

バの配置先として需要の発生した地域にある適切な候補マシンを選出する．また EPSR はマシンが過負荷にならないように，需要の変動に応じて配置先候補となるマシンの再割り当てを需要の発生地域ごとに継続して行う．

EPSR は全サーバのアクセスログや性能情報および負荷情報のような大域情報を使用せず，Peer-to-Peer 技術を用いることで急激な需要の増加に対応する．個々のマシンは自律的に需要の変動を検出し，必要に応じて複製サーバの配置先候補となる．個々のマシンでの需要変動の検出と需要の発生地域の特定を実現するために，EPSR は物理ネットワークのトポロジを考慮したオーバレイネットワークを，ネットワーク座標系と分散ハッシュテーブルを組み合わせて構築する．個々のマシンは，オーバレイ上で隣接するマシンとのみメッセージを交換することでクライアントのアクセス経路を分析し，サービスへの需要が発生している地域を特定する．マシンはアクセス経路の分析から複製サーバの実行による効果を見積もり，配置先の候補となるかどうか決定する．個々のマシンは複製サーバの負荷とマシンの許容負荷を元に，マシンが過負荷にならないように隣接するマシンから追加の配置先候補を独立して選択する．

EPSR の構成要素と達成に貢献した課題の対応を表 3.1 に示す．ホスティングマシンによる自律的な需要発生地域の特定は高速性の実現と通信遅延の低下に貢献

する．GNP と CAN を組み合わせて実現した，物理ネットワークのトポロジを考慮したオーバレイネットワークは通信遅延の低下に貢献する．VPE と VDR はオーバレイネットワーク上でホスティングマシンが行う需要発生地域の特定の精度向上に貢献する．ホスティングマシンによる複製サーバの配置先の自律選択は高速性の実現と性能制約の達成に貢献する．Dummy Query と高負荷な複製サーバの優先配置は通信遅延の低下と性能制約の達成に貢献する．Dummy Connection は性能制約の達成に貢献する．

次章では本章で述べた EPSR の効果を確認するために行った実験について述べる．シミュレーションの中で Flash Crowd を再現し，EPSR による複製サーバ配置先の決定の効果を確認する．

第4章　実験

本章では，EPSR の効果を確認するために行った実験結果について述べ，EPSR の効果を明らかにする．実験ではシミュレーションで Flash Crowd が発生している状況を再現し，需要の急激な増加に対して EPSR が適切な配置先を決定することを確認する．具体的には，EPSR が配置先の決定に要する時間，配置先のホスティングマシンとクライアント間の通信遅延，配置先となるホスティングマシンの負荷を測定し，効果を確認する．

4.1　実験方法

EPSR の効果を評価するために，オーバレイ構築ツールキット Overlay Weaver [85, 86] 上に EPSR を実装した．Overlay Weaver は構造化オーバレイネットワークの容易な実装と評価を実現するフレームワークである．異なる需要変動をシミュレートすることで EPSR の評価実験を行った．評価実験では以下の 5 つの指標について計測を行った．

- **経過時間．**これは EPSR が需要の発生地域の特定と配置先の決定に要した時間を示す．Flash Crowd による需要増加に対して短時間であれば，EPSR が高速性を備えていることが示される．
- **複製サーバの配置先の数．**これは需要の規模の変動に応じて EPSR が配置先の数をどの程度調整できているかを示す．配置先の数が需要の変動に合わせて増減していれば，EPSR が需要変動に応じて複製サーバの配置先の数を調整できていることが示される．
- **クライアントから複製サーバの配置先までのオーバレイネットワーク上のホップ数．**これは EPSR がオーバレイネットワーク上においてどの程度クライアントの近傍にあるホスティングマシンを複製サーバの配置先として選択できているかを示す．ホップ数が少ないほど，EPSR がオーバレイネットワー

ク上においてクライアントの近傍にあるホスティングマシンを配置先として選択できていることを示す.

- **クライアントから複製サーバの配置先までのラウンドトリップタイム** (*RTT*)**.** これは EPSR がクライアントに対してどの程度通信遅延の小さいホスティングマシンを複製サーバの配置先として選択できているかを示す．RTT が小さいほど，EPSR がクライアントに対して通信遅延の小さいホスティングマシンを配置先として選択できていることを示す.

- **同時接続数．**これは複製サーバおよびそれを稼働しているホスティングマシンの負荷を示す．同時接続数がホスティングマシンの性能制約以下であれば，EPSR が性能制約に従って複製サーバの配置先を選択できているといえる．また，複製サーバを稼働しているホスティングマシンの同時接続数の集合における最低値が大きく分散が小さいほど，無駄な複製サーバが少ないといえる.

より現実に即した条件下で EPSR の挙動を確認するために 2 種類のデータセットを用意した．用意したデータセットは，インターネットを模した仮想トポロジから得たものとインターネットでの実測によるものである．インターネットの実測データは King 計測手法 [115] によって計測されたものである．実験ではこれら 2 つのデータセットを用いて擬似的なインターネットのトポロジを構成し，そのトポロジ上で EPSR を動作させた.

用意した 2 つのデータセット *TS* および *Meridian* の詳細は次の通りである.

1. *TS* **データセット**: 2 階層のルーティングドメインを持つ Transit-Stub インターネットモデルである [87]．Transit-Stub モデルは Transit ドメインが下位にある Stub ドメインを接続する構造でインターネットをモデル化する．Transit-Stub モデルを生成するために GT-ITM トポロジ生成器を利用した [116]．Transit-Stub モデルによるトポロジの生成パラメータは以下の通りである．Transit ドメイン数は 228 とした．それぞれの Transit ドメインに 5 つの Transit ノードをおき，それぞれの Transit ノードに対して 4 つの Stub ドメインを割り当てた．それぞれの Stub ドメインに 2 つのノードを割り当てた．実験では Transit-Stub モデルで生成した約 10,000 ノードのうち，GNP によって計算されたネットワーク座標を一意に割り当てることができた約 3,000 ノードをホスティングマシンとして使用した.

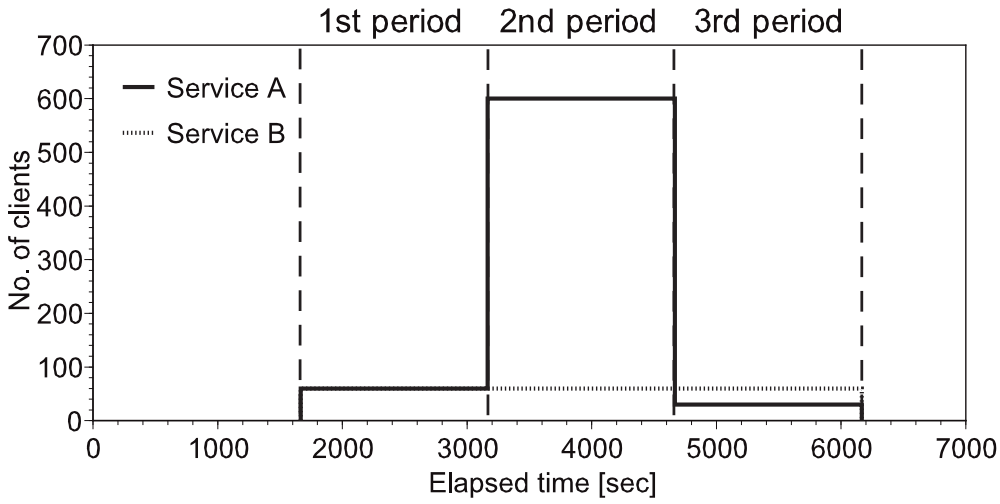


図 4.1: **サービス A およびサービス B のクライアント数の推移．**

2. *Meridian* **データセット**: Meridian プロジェクト [117] で使用されたインターネットの実測データである．一意な IP アドレスをもつ 2,500 の DNS サーバ間の RTT から成る．組み合わせの数は約 625 万である．このデータセットは 2004 年 5 月 5 日から 13 日の間に収集された．実験ではこのデータセットが含む 2,500 台のうち，GNP によって計算されたネットワーク座標を一意に割り当てることができた約 2,200 台をホスティングマシンとして使用した．

なお，本実験で使用するネットワーク座標の次元数は，文献 [73] によるネットワーク座標系の精度の評価に基づき 6 とした．

4.2 EPSR による複製サーバの配置先決定の評価

EPSR は Flash Crowd のように急激に需要が変動しても，需要の発生している地域を特定し，ホスティングマシンの性能制約を満たすように複製サーバの配置先を決定する．この EPSR の特徴を確認するために，EPSR の複製サーバ配置先決定部の評価を行った．評価はシミュレーションによって行い，ネットワークのモデルには Transit-Stub モデルを，通信遅延のデータセットには TS データセットを使用した．

EPSR の複製サーバ配置先決定部の特徴を確認するために，局所的に高い需要が発生している地域をシミュレーション上で再現し実験を行った．クライアントは TS モデルで，互いの RTT が 4000 ミリ秒以下になるように設定した．クライアン

トは 60 秒ごとにサービスに対するリクエストメッセージを送信する．EPSR は複数のサービスが同時に提供されている場合でも需要変動に応じて複製サーバの配置先を決定できる．これを確認するために，本シミュレーションでは 2 つのサービス A，B が同時に提供されているように設定した．図 4.2 はシミュレーション中のクライアント数を示す．図 4.2 が示すように，サービス A のクライアント数は区間毎に増減するが，サービス B のクライアント数は一定とした．サービス A とサービス B のクライアントは同じ地域から選択した．

EPSR が性能制約を考慮して複製サーバの配置先を決定できていることを確認するために，以下の方式との比較を行った．

- *Root_only*: 複製サーバを配置しない．
- *Marker_only*: マーカとなったホスティングマシンのみを複製サーバの配置先とする．
- *Globule*: Globule [49–51] で用いている HotSpot アルゴリズム [52,53] に基づいてマーカを選択する．HotSpot アルゴリズムは全複製サーバのアクセス記録やホスティングマシンの位置情報のような大域情報に依存するため，長期の需要変動のみに対応する．一方で大域情報に基づいて計算するためマーカ選択の精度は高く，最適解に最も近いとみなすことができる．これより，EPSR の測定結果が GLOBULE のそれと近い場合，EPSR は各ホスティングマシンがもつ局所情報のみで最適解に近いマーカ配置が得られることを示す．HotSpot アルゴリズムが本シミュレーションで発生させる短期の需要変動を対象としていないため，HotSpot アルゴリズムによるマーカ選択はオフラインで行った．

なお，Globule が複製サーバの配置先としたホスティングマシンの数は，サービス A を 90 台，サービス B を 10 台とした．これはシミュレーション中で Flash Crowd が発生したと想定した第 2 区間の後半において EPSR が決定した配置先の数に合わせたものである．

本シミュレーションにおける設定値を表 4.1 に示す．APC degree を管理するスライディングウィンドウは 300 秒分とし，各ホスティングマシンは 30 秒間隔で APC degree を確認するように設定した．マーカとなる閾値 $upper$ は 300，マーカから一般のホスティングマシンに戻る閾値 $lower$ は 200 とした．ホスティングマシン 1 台当たりの性能制約は，クライアントの接続数を基準とし，同時接続数 30

表 4.1: **EPSR による複製サーバの配置先決定の評価で使用した各種パラメータ.**

サービス数	2
クライアントのリクエスト間隔	60 秒
スライディングウィンドウのサイズ	300 秒分
APC degree の確認間隔	30 秒
マーカとなる閾値 $upper$	300
マーカから戻る閾値 $lower$	200
ホスティングマシンの性能制約	同時接続数 30 以下

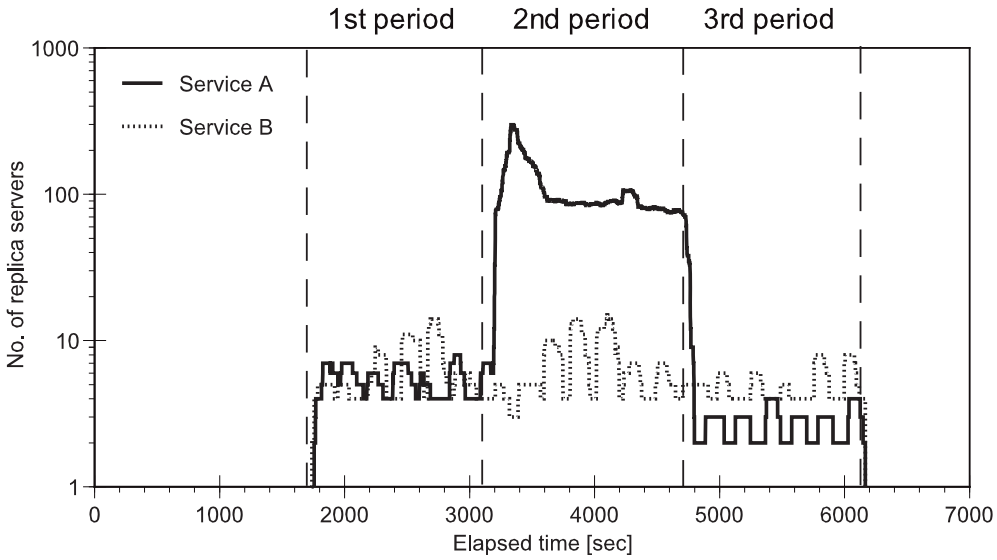


図 4.2: **EPSR による複製サーバの配置先候補点数の推移.**

とした．すなわち，ホスティングマシンは最大で 30 クライアントまでを，マシン上で稼働する複製サーバは許容できるものとした．また，複製サーバが与える負荷はサービス A とサービス B で等しいとした.

4.2.1 複製サーバの配置先数の推移と決定に要した時間

図 4.2 は EPSR が決定した複製サーバの配置先数の変動を示す．x 軸は経過時間を，y 軸は複製サーバの配置先数を示す．y 軸は対数である．シミュレーション結果は EPSR が需要変動に応じて複製サーバの配置先数を調整できていることを示している．サービス A の需要が変動に応じて，サービス A の複製サーバの配置先

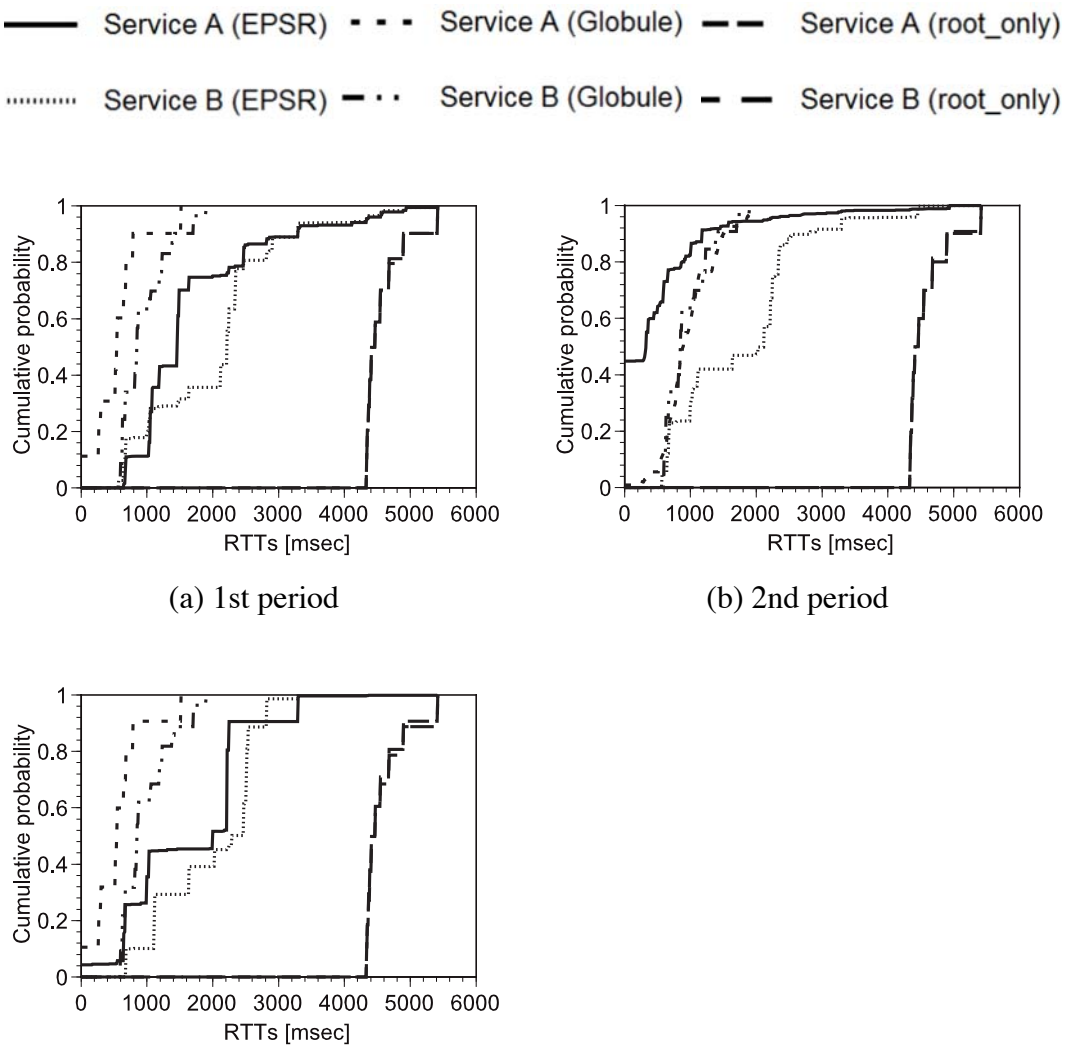
数が増減した．シミュレーションの第 2 区間では，図 4.1 が示すようにクライアント数が第 1 区間の 10 倍となっている．この需要に対応するために，EPSR は複製サーバの配置先数の増加を 25 秒後に開始した．これは文献 [10] で報告されている Flash Crowd がピークに達するまでに要した時間 40 秒より高速である．その後，EPSR はサービス A の複製サーバの配置先数を，166 秒間で最初の 7 から 299 まで増加した．さらに 300 秒経過後，EPSR は負荷の少ない複製サーバを終了し，最終的に複製サーバの配置先数を 90 とした．第 3 区間では，サービス A の複製サーバの配置先数はサービス B のものより少ない．これは第 3 区間ではサービス A のクライアント数がサービス B のものよりも少ないからである．

さらに図 4.2 は，EPSR がサービスごとに独立して複製サーバの配置先数を調整できていることを示す．第 1 区間から Flash Crowd の発生した第 2 区間にかけてサービス A のクライアント数が 60 から 600 と 10 倍に増加したとき，サービス A の複製サーバの配置先数が 7 から 299 に増加した．これは EPSR がサービス A の需要増加を検出したことを示す．ただし，同区間においてサービス B の複製サーバの配置先数も他区間と比較して若干増加している．これはサービス A のクライアント数の増加によって，サービス B の複製サーバの配置先であるホスティングマシンの負荷が増加したためである．サービス A のクライアント数の増加に伴って，サービス A の複製サーバとサービス B の複製サーバの配置先へ同時になるホスティングマシンが増加する．このときホスティングマシンの性能制約を超過する場合，EPSR は高需要の複製サーバを優先する．これにより，低需要であるサービス B の複製サーバの配置先が増加する．

4.2.2 クライアントと複製サーバの配置先間の通信遅延

EPSR が需要の発生している地域にあるホスティングマシンから複製サーバの配置先を選択できていることを確認するために，クライアントと複製サーバの配置先間の RTT を EPSR，Root_only，Globule で比較した．図 4.3 (a)，(b) および (c) はそれぞれ第 1 区間，第 2 区間，第 3 区間の RTT の分布を示す累積頻度グラフである．

図 4.3 は，EPSR が需要の発生した地域に複製サーバの配置先を設定していることを示す．特に Flash Crowd が発生している第 2 区間においては，EPSR が Globule よりも多くのクライアントの RTT を小さくした．EPSR ではクライアントと複製サーバの配置先との RTT の 90 パーセンタイル値が 1180 ミリ秒であったのに対し



(a) 1st period

(b) 2nd period

(c) 3rd period

図 4.3: **配置先候補点との RTT の累積頻度グラフ.**

て，Globule の 90 パーセンタイル値は 1500 ミリ秒であった．これは EPSR はホスティングマシンが過負荷にならないように需要の発生している地域内で積極的に複製サーバの配置先を増加させるからである．EPSR では需要の発生している地域内の多くのホスティングマシンがサービス A の複製サーバの配置先となったため，44 パーセントのクライアントのリクエストが最寄りのホスティングマシン上の複製サーバで処理された．一方，Globule は第 2 区間では RTT が大きいクライアントが他区間と比較して増加した．第 1 区間の RTT の 90 パーセンタイル値は 790 ミリ秒であるのに対して，第 2 区間の RTT の 90 パーセンタイル値が 1500 ミ

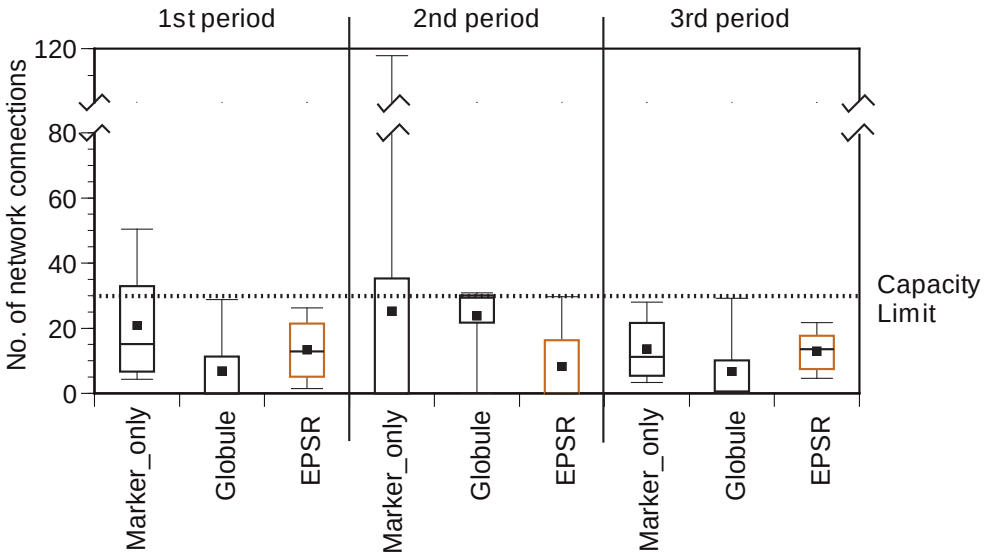


図 4.4: **同時接続数の分布．**

リ秒であった．これは Globule ではホスティングマシンが高負荷になると，クライアントのリクエストを遠方にある複製サーバを稼働中の他のホスティングマシンへ転送するからである．

また EPSR は通常時においてもクライアントの RTT を小さくするホスティングマシンを複製サーバの配置先としている．第 1 区間において，EPSR での RTT の 90 パーセンタイル値は，サービス A，B ともに 3300 ミリ秒であった．一方で Root_only の RTT の 90 パーセンタイル値はサービス A，B ともに 4900 ミリ秒であった．第 3 区間では，サービス A の RTT の 90 パーセンタイル値が 2250 ミリ秒，サービス B の RTT の 90 パーセンタイル値が 2540 ミリ秒であった．一方で Root_only の RTT はサービス A，B ともに 4900 ミリ秒であった．Globule では，第 1 区間および第 3 区間におけるクライアントの RTT の 90 パーセンタイル値は EPSR の場合の約 4 分の 1 であった．これは Globule が大域情報を用い，最適配置の近似解をオフラインで算出しているからである．

4.2.3 性能制約に対するホスティングマシンの負荷

EPSR はホスティングマシンが過負荷状態にならないように複製サーバの配置先を決定できていることを確認するために，負荷の指標とした同時接続数の分布を各方式間で比較した．図 4.4 は各区間における複製サーバの配置先であるホスティ

ングマシンの同時接続数の分布を方式ごとに示した箱ひげ図である．図 4.4 が示すように，EPSR はすべてのホスティングマシンの同時接続数が性能制約である 30 以下となっている．これは EPSR が過負荷とならないように複製サーバの配置先数を調整できていることを示している．一方で，Marker_only は性能制約を満たした複製サーバの配置先数の調整が行えていない．特に過負荷となっているホスティングマシンの数はクライアント数が多いほど増加した．Flash Crowd の発生を再現した第 2 区間では，25 パーセント以上のホスティングマシンが性能制約を超過した．

また EPSR は余剰な複製サーバの配置先を減少させている．EPSR では，Flash Crowd の発生していない第 1 区間および第 3 区間において，接続数が 0 となる複製サーバが存在しないように複製サーバの配置先を決定できている．これにより EPSR は他と比べて同時接続数の最低値が大きく，分散が小さくなっている．これは EPSR が需要の減少に合わせて複製サーバの接続数が 0 となるようなホスティングマシンを配置先から除外する操作を行っているからである．第 2 区間では低負荷となるホスティングマシンの数が他区間と比較して増加している．これは EPSR が急激な需要の増加に対して，ホスティングマシンの性能制約を超えないように急速に複製サーバの配置先を増加させるからである．一方で，Globule は第 1 区間および第 2 区間において余剰な複製サーバの配置先が EPSR と比較して多い．これは Globule が複製サーバの配置先を，需要のピーク時である第 2 区間に合わせて決定しているためである．

4.3 EPSR 各部の評価

EPSR を構成する各要素の，複製サーバの配置先決定に対する効果を確かめるために，各部に対する評価を行った．以下では実施した実験結果について述べる．

4.3.1 需要発生地域特定部の評価

第 3.5 節で述べた，需要が発生している地域を特定する部分の効果を確認する実験について述べる．EPSR は需要の規模の増減や発生地域の移動が生じても，変化に追従して需要が発生している地域を特定する．また EPSR は需要が Flash Crowd のように急激に変動しても，数十秒で需要が発生している地域を特定する．発生している需要の規模が拡大した場合，EPSR はより多くのホスティングマシンをマー

カとする．この結果，マーカとクライアントとの RTT やオーバレイネットワーク上のホップ数が減少する．また需要の規模が縮小した場合，EPSR は余剰なマーカを消去する．EPSR はマーカ数を需要の規模に応じて調整するため，マーカの数が減少しても RTT やホップ数はルートマシンにアクセスするよりも低いままとなる．

EPSR の特徴を評価するために 2 通りの条件を用意して実験を行った．ひとつは，需要の規模が増減する場合である．もうひとつは，需要の発生している地域が移動する場合である．さらに，需要の発生地域の特定部分に利用した要素技術の効果についても評価を行った．以下，実験の詳細について述べる．

アクセス頻度の変動に対する評価

アクセス頻度の変動に合わせて EPSR がマーカの数を適切に増減できているか確認するために，クライアントのアクセス頻度を増減させて EPSR が割り当てたマーカの変化を確認するシミュレーションを行った．シミュレーションでは TS データセットを用い，クライアントのアクセス頻度を一定間隔で変動させた．図 4.5 はシミュレーションで発生させたクライアントのアクセス頻度を示す．クライアントは Transit-Stub モデルの終端ノードの近傍に存在すると仮定した．Transit-Stub モデルの終端ノードとは，Transit-Stub モデルのトポロジにおいて，1 ノードとしか他のノードとは接続していないノードを指す．このようなノードをクライアントが属するローカルエリアネットワークのゲートウェイと本シミュレーションでは見立てた．本シミュレーションではクライアントのアクセスを発生させる際，終端ノードをランダムに 1 つ選択し，そのノードへクライアントからのアクセスがあったとみなす．なお，終端ノードの数は 1,815 であった．

シミュレーションでは比較のために，マーカを割り当てなかった場合のホップ数および RTT の計測を行った．本節ではこの条件を *ROOT_ONLY* と呼ぶ．これはすなわち，ルートマシンのみがサービスを提供した場合の性能を示す．

シミュレーションで使用したパラメータを表 4.2 に示す．提供しているサービスの数は 1 とした．APC degree を管理するスライディングウィンドウは 300 秒分とし，各ホスティングマシンは 60 秒間隔で APC degree を確認するように設定した．これは文献 [10] による，過去発生した Flash Crowd は 40 秒から 15 分でクライアント数の上昇がピークに達したという報告を参考にした．各ホスティングマシンがマーカとなる閾値 $upper$ は 300，マーカから一般のホスティングマシンへ戻る閾値 $lower$ は 150 とした．

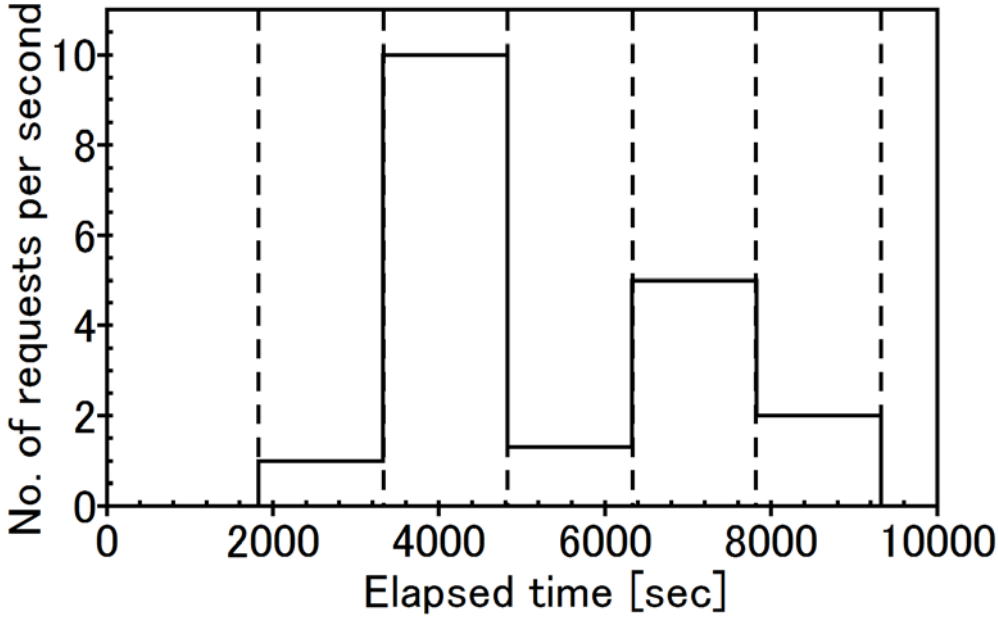


図 4.5: **発生させたリクエスト数の推移．**

表 4.2: **アクセス頻度を変動させた際の需要発生地域の特定部に対する評価で使用した各種パラメータ．**

サービス数	1
スライディングウィンドウのサイズ	300 秒分
APC degree の確認間隔	60 秒
マーカとなる閾値 $upper$	300
マーカから戻る閾値 $lower$	150

図 4.6 は EPSR が割り当てたマーカ数を示す．縦の破線はアクセス頻度が変化した時点を示す．シミュレーション結果は EPSR がアクセス頻度の変動に合わせてマーカ数を動的に増減させたことを示している．アクセス頻度が増加すると EPSR はマーカ数を増加させ，アクセス頻度が減少すると EPSR はマーカ数を減少させた．これは，クライアントのアクセス数が突然増減しても，EPSR がアクセス数の増減に合わせてマーカ数を調整できていることを示す．例えば図 4.6 において，3,200 秒経過時点のマーカ数と 6,400 秒経過時点のマーカ数を比較すると，前者のマーカ数が 160 程度まで増加したのに対して，後者は 40 程度まで上昇した．これはそれぞれのアクセス頻度の大小関係に対応している．マーカ数の増減はアクセス

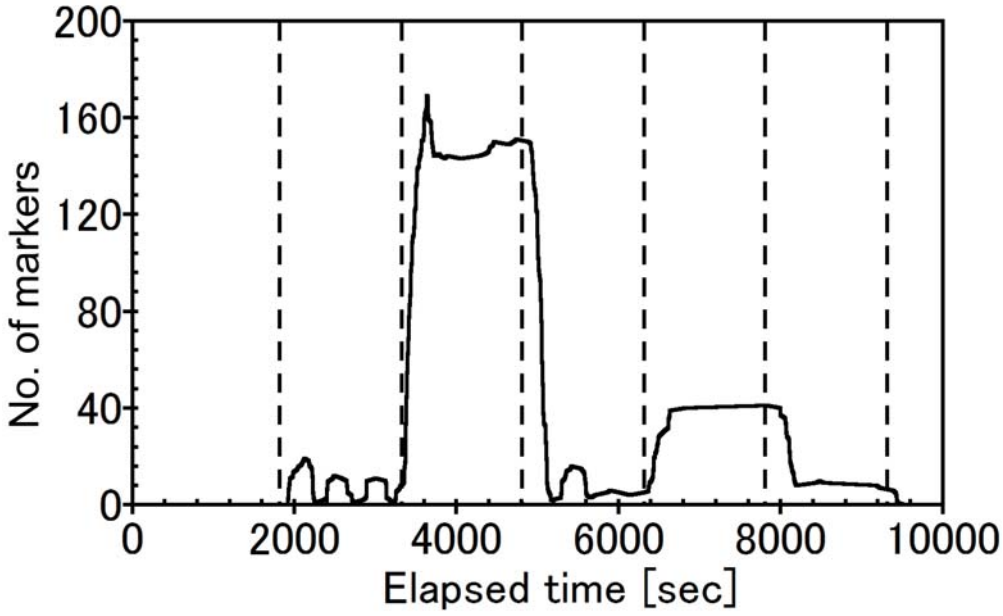


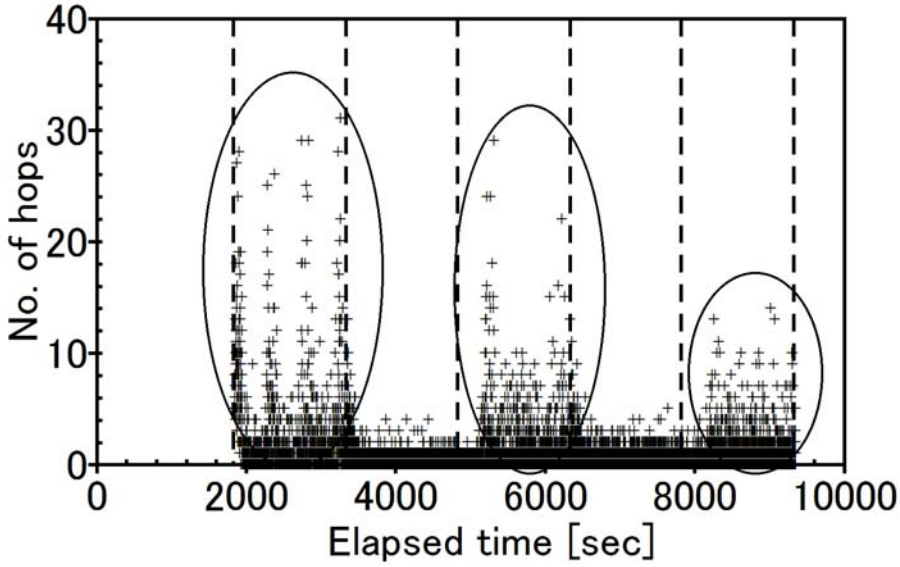
図 4.6: **EPSR によって設定されたマーカ数の推移．**

頻度が変動してから 10 秒程度で開始した．その後マーカ数が安定するまでに要した時間は最大で 340 秒であった．APC degree の確認間隔を短くすることで EPSR の応答速度を高め，より迅速にマーカ数を調整できるようになる．

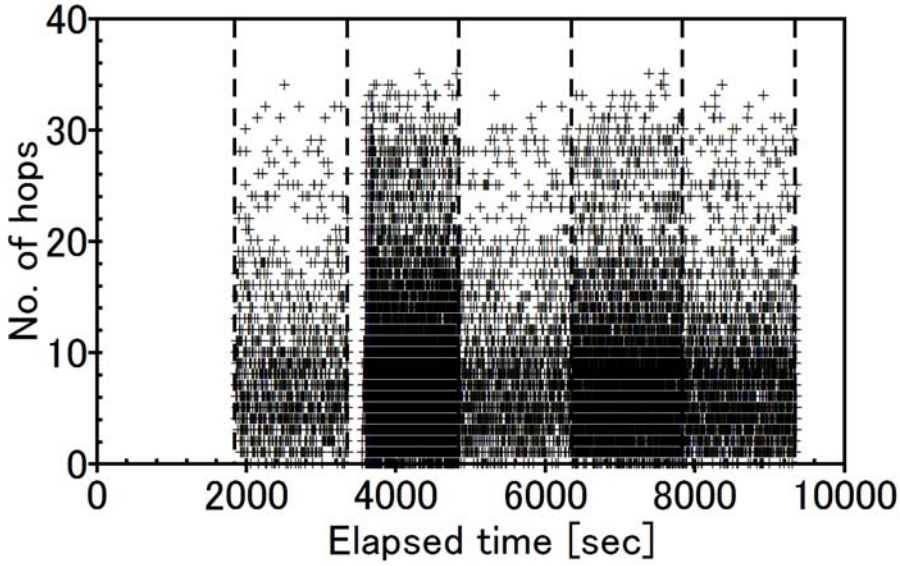
図 4.7 はクライアントと最寄りのマーカ間のオーバレイネットワークにおけるホップ数を示す．図 4.7 (a) は EPSR の場合を，図 4.7 (b) は ROOT_ONLY における測定結果を示す[1]．縦の破線はアクセス頻度が変化した時点を示す．ROOT_ONLY と比較してホップ数が小さいほど，EPSR がオーバレイネットワーク上においてクライアントの近傍にあるホスティングマシンをマーカとして選択できているといえる．図 4.7 (b) が示す結果と比較して，EPSR が選択したマーカとクライアントとの平均ホップ数がルートマシンとの平均ホップ数に比べて低下していることが分かる．EPSR の場合，アクセス頻度が高い期間では平均ホップ数が 4 程度であった．アクセス頻度が低くなると，図 4.7 (a) の楕円で囲まれた区間が示すように，平均ホップ数は大きくなる．これはアクセス頻度の低下によってクライアントの集中する地点が減少したことを EPSR が正しく検出しているからである．

図 4.8 はクライアントと最寄りのマーカ間の RTT を示す．図 4.8 (a) は EPSR の場合を，図 4.8 (b) は ROOT_ONLY における測定結果を示す[2]．ROOT_ONLY と比

[1]ROOT_ONLY ではマーカがないため，ルートマシンとのホップ数になる．
[2]ROOT_ONLY ではマーカがないため，ルートマシンとの RTT になる．



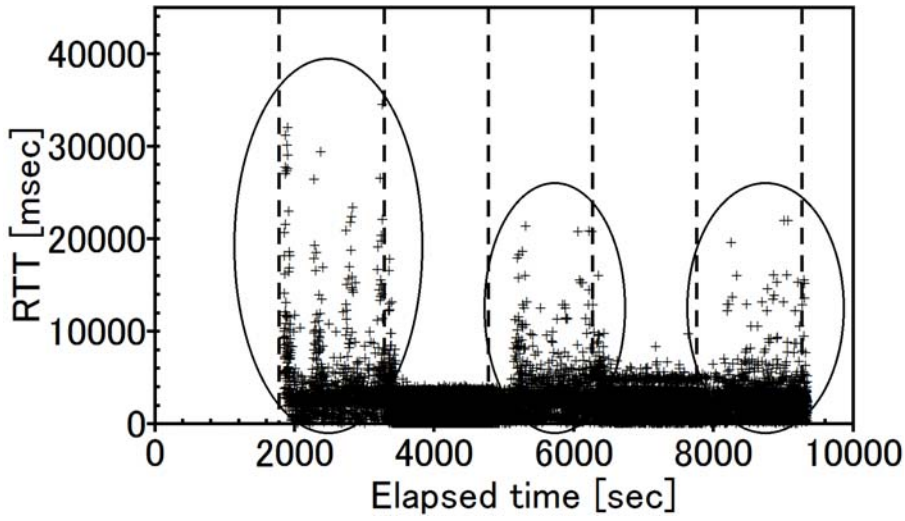
(a) EPSR



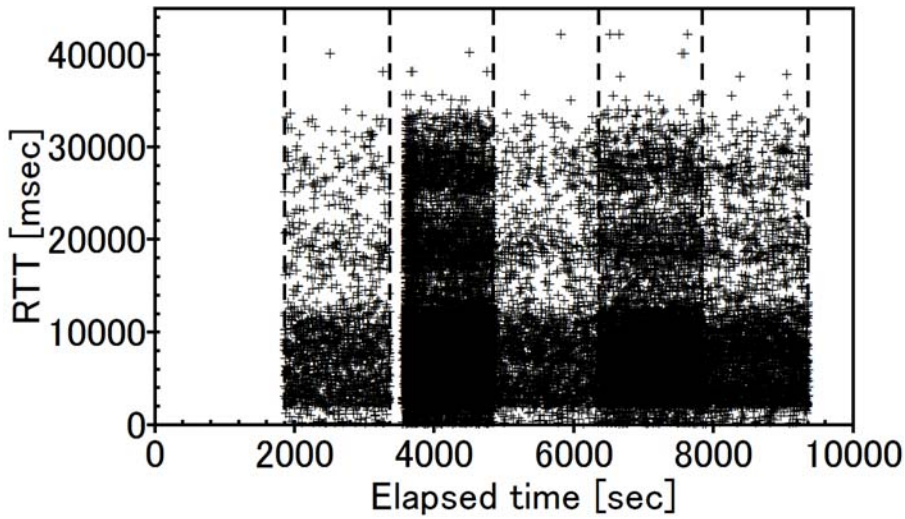
(b) ROOT_ONLY

図 4.7: **EPSR と ROOT_ONLY におけるホップ数の比較．**

較して RTT が小さいほど，EPSR が物理ネットワーク上においてクライアントの近傍にあるホスティングマシンをマーカとして選択できているといえる．図 4.8 (b) が示す結果と比較して，EPSR が選択したマーカとクライアントとの平均 RTT がルートマシンとの平均 RTT に比べて低下していることが分かる．アクセス頻度が低くなると，図 4.8 (a) の楕円で囲まれた区間が示すように，平均 RTT は大きくな



(a) EPSR



(b) ROOT_ONLY

図 4.8: **EPSR と ROOT_ONLY における ラウンドトリップ時間 (RTT) の比較．**

る．これはホップ数の場合と同様に，アクセス頻度の低下によってクライアントの集中する地点が減少したことを EPSR が正しく検出しているからである．

EPSR によって選択されたマーカが他の手法によって選択されたものよりも需要の発生地域の範囲を高精度で特定できていることを確認するために，以下の 3 つの手法との比較実験を行った．

- *RND*: 全てのホスティングマシンからマーカをランダムに選択する．

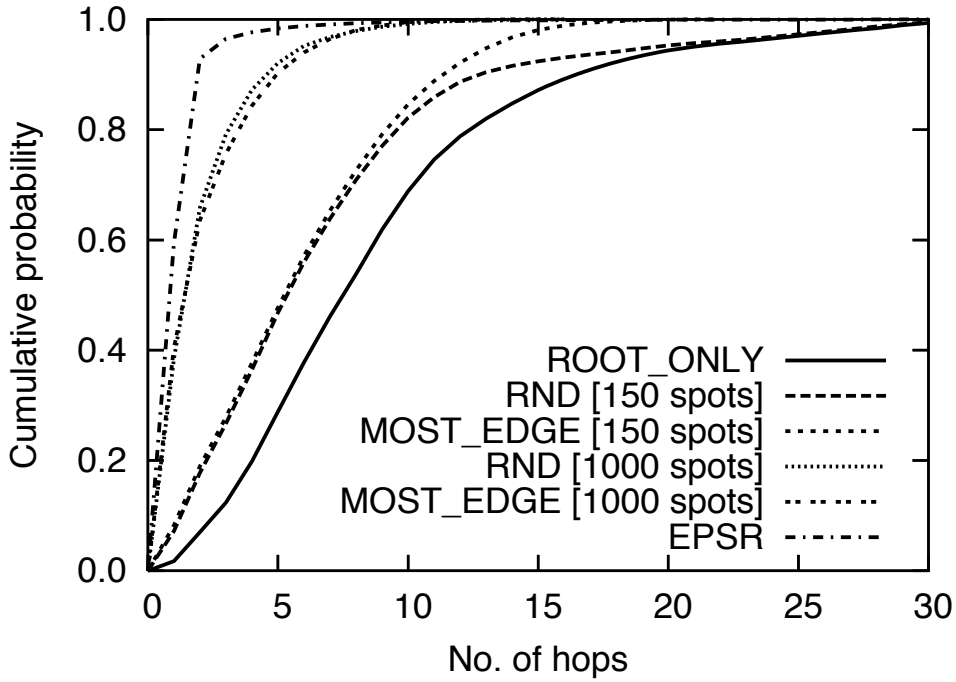


図 4.9: **ホップ数の累積頻度グラフ．**

- *MOST_EDGE*: Transit-Stub トポロジにおいて終端ノードと多く接続しているホスティングマシンをマーカとして選択する．

- *GLOBULE*: Globule [49–51] で用いている HotSpot アルゴリズム [52,53] に基づいて配置先を決定する．最適解に最も近い配置が得られる．第 4.2 節と同様に，HotSpot アルゴリズムによる配置先の決定はオフラインで行った．なお，Globule のオーバレイネットワークと EPSR で使用したものとは構造が異なるため，ホップ数の測定は行っていない．

また比較のため，マーカを選択しない ROOT_ONLY の場合の計測も行った．EPSR が最大で 150 台のホスティングマシンをマーカとして選択したため，比較のために用意した RND，MOST_EDGE，GLOBULE も同じく 150 のマーカを配置するよう設定した．

図 4.9 はホップ数の分布を示す累積頻度グラフである．EPSR は RND や MOST_EDGE と比較して，ホップ数が小さくなるようなホスティングマシンをマーカとして選んでいる．EPSR が決定したマーカとクライアントとのホップ数の 90 パーセンタイル値は 2 ホップであった．一方 RND の 90 パーセンタイル値は 13 ホップ，MOST_EDGE の 90 パーセンタイル値は 12 ホップであった．また，

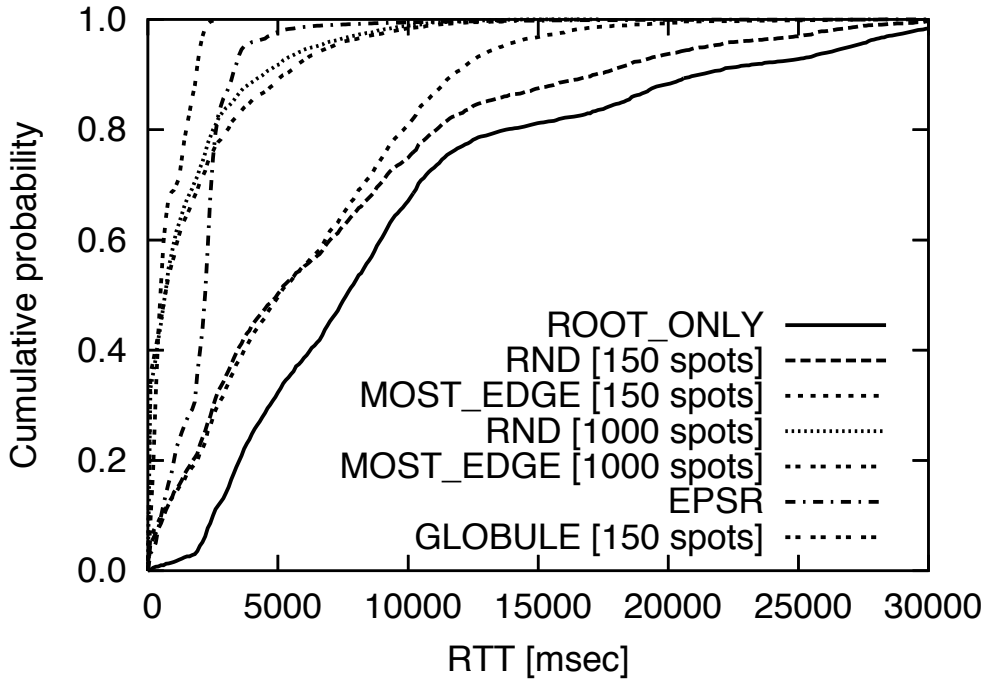


図 4.10: **RTT の累積頻度グラフ．**

ROOT_ONLY の 90 パーセンタイル値は 17 ホップであった．

図 4.10 は RTT の分布を示す累積頻度グラフである．ホップ数の場合と同様に，EPSR は RND や MOST_EDGE と比較して RTT が小さくなるようなホスティングマシンをマーカとして選んでいる．EPSR が決定したマーカとクライアントとの RTT の 90 パーセンタイル値は 3,300 ミリ秒であった．一方 RND の 90 パーセンタイル値は 17,200 ミリ秒，MOST_EDGE の 90 パーセンタイル値は 11,800 ミリ秒であった．また，ROOT_ONLY の 90 パーセンタイル値は 21,300 ミリ秒であった．

比較のため，RND と MOST_EDGE が決定するマーカの数を 150 から 1000 に増加させた際のホップ数と RTT も合わせて計測した．図 4.10 が示すように，RTT の 80 パーセンタイル値は EPSR よりも小さく，良い結果を示した．これは RND と MOST_EDGE が決定したマーカの数が EPSR の約 6.7 倍であるため，通信遅延の小さいホスティングマシンを選び出す確率が高くなったためである．しかしながら，RTT の 90 パーセンタイル値では EPSR が RND と MOST_EDGE を逆転している．また，図 4.9 が示す通り，ホップ数では EPSR の方が RND と MOST_EDGE よりも終始小さい値を示した．

EPSR は GLOBULE に近い性能を示した．GLOBULE が決定したマーカは最適解に近いため，この結果は EPSR が最適解に近づくようにマーカを決定できてい

ることを示す．GLOBULE は EPSR よりも良いマーカを決定できているが，これは GLOBULE があらかじめ収集した大域情報から集中計算を行うことで近似解を導き出しているためである．

需要発生地域の移動に対する評価

EPSR は需要の発生地域が移動しても，移動した先を特定することができる．需要が移動しても EPSR が移動先を特定できることを確認するために，本節では EPSR が需要発生地域の移動に合わせてマーカを決定することをシミュレーションにより示す．需要の発生地域はシミュレーションで使用するそれぞれのトポロジに対してクライアントのクラスタを設定することで設定した．クライアントのクラスタの詳細は次の通りである．

- CLS(TS): TS データセットの終端ノードからクライアントのゲートウェイを選択する[3]．需要の発生地域の中心点となる終端ノードを，それぞれの RTT が 5,000 ミリ秒以上となるように 3 つ選び出す．中心点として選び出した終端ノードごとに，中心点からの RTT が 2,000 ミリ秒以内となる終端ノードを 10 選び出し，これをクライアントのゲートウェイとする．

- CLS(Meridian): Meridian データセットの全てのノードから選択する．需要の発生地域の中心点となるノードを，それぞれの RTT が 250 ミリ秒以上となるように 3 つ選び出す．中心点として選び出したノードごとに，中心点からの RTT が 100 ミリ秒以内となるノードを 10 選び出し，これをクライアントのゲートウェイとする．

CLS(TS) と CLS(Meridian) の生成パラメータが異なるのは，CLS(TS) と CLS(Meridian) の平均 RTT が異なるためである．これは TS データセットが基礎とするトポロジと Meridian データセットのそれとの違いに起因する．CLS(TS) は Transit-Stub モデルを，CLS(Meridian) は実測時のインターネットを基礎としている．

シミュレーションで使用したパラメータを表 4.3 に示す．シミュレーションでは，クライアントのクラスタは 5000 秒ごとに切り替えた．クライアントは 500 ミリ秒ごとに一台選択され，サービスへの要求メッセージを送信した．提供しているサー

[3]終端ノードはクライアントのゲートウェイであるが，ここではゲートウェイとクライアントを同一のものとして考える．

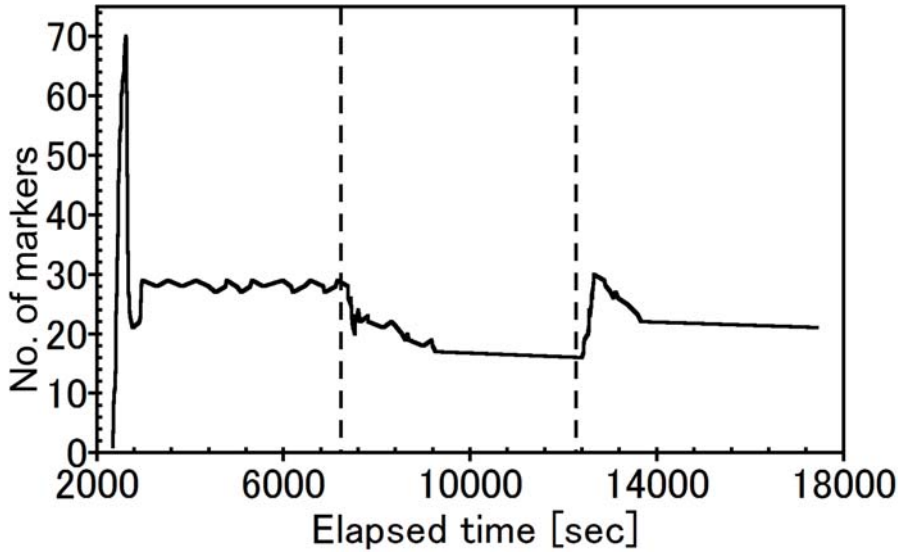
表 4.3: **需要発生地域を移動させた際の需要発生地域の特定部に対する評価で使用した各種パラメータ.**

サービス数	1
クライアントのクラスタの切り替え間隔	5000 秒
クライアントの選択間隔	500 ミリ秒
スライディングウィンドウのサイズ	300 秒分
APC degree の確認間隔	60 秒
マーカとなる閾値 $upper$	300
マーカから戻る閾値 $lower$	200

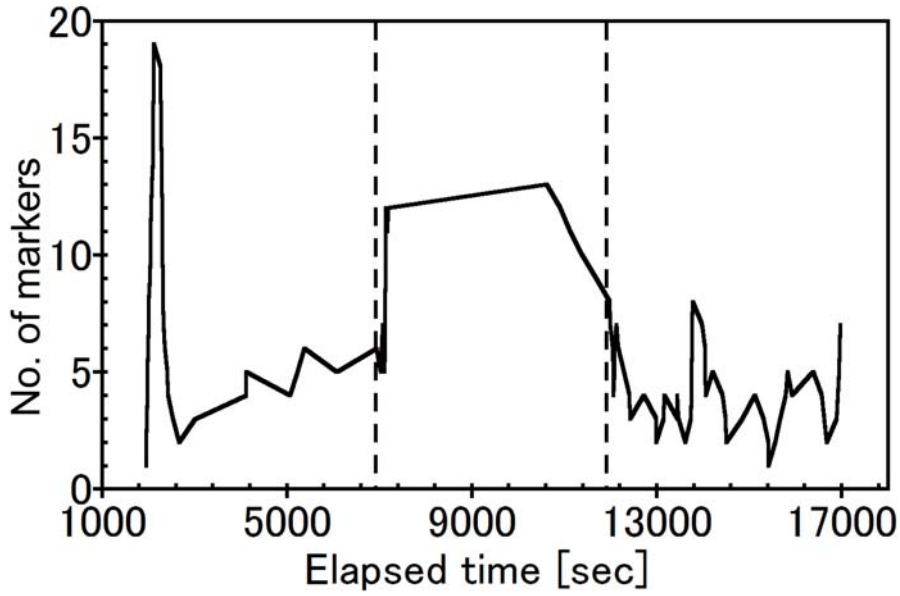
ビスの数は 1 とした．APC degree を管理するスライディングウィンドウは 300 秒分とし，各ホスティングマシンは 60 秒間隔で APC degree を確認するように設定した．マーカとなる閾値 $upper$ は 300，マーカから一般のホスティングマシンに戻る閾値 $lower$ は 200 とした．

図 4.11 は本シミュレーションにおける EPSR が決定したマーカ数の推移を示す．縦の破線はクライアントのクラスタを変更した時点を示す．図 4.11 (a) は CLS(TS) の場合を，図 4.11 (b) は CLS(Meridian) における測定結果を示す．いずれの場合も，EPSR はクライアントのクラスタが変更された際にマーカ数を増減させている．クライアントのクラスタが変更されると，すぐにマーカ数は大きく増減している．これは EPSR がクライアントのアクセス経路に合わせて一時的にマーカ数を増減させているためである．その後，EPSR はクライアントから遠い不要なマーカを迅速に除去し，需要発生地域の特定の精度を高める．

図 4.12 はクライアントと最寄りのマーカ間のオーバレイネットワークにおけるホップ数を示す．縦の破線はクライアントのクラスタを変更した時点を示す．図 4.12 (a) は CLS(TS) の場合を，図 4.12 (b) は CLS(Meridian) における測定結果を示す．EPSR が選択したマーカとクライアントとの平均ホップ数は，クライアントのクラスタの変更に追従して小さくなるように変動している．もし EPSR がクライアントの地理的分布の移動に追従できていなければ，クライアントのクラスタが変更された時点でホップ数が大きく上昇するはずである．シミュレーションでは CLS(Meridian) の 2 番目の区間を除いて，EPSR はホップ数が小さくなるようにマーカを選択した．CLS(Meridian) の 2 番目の区間は他に比べてホップ数が高



(a) CLS(TS)

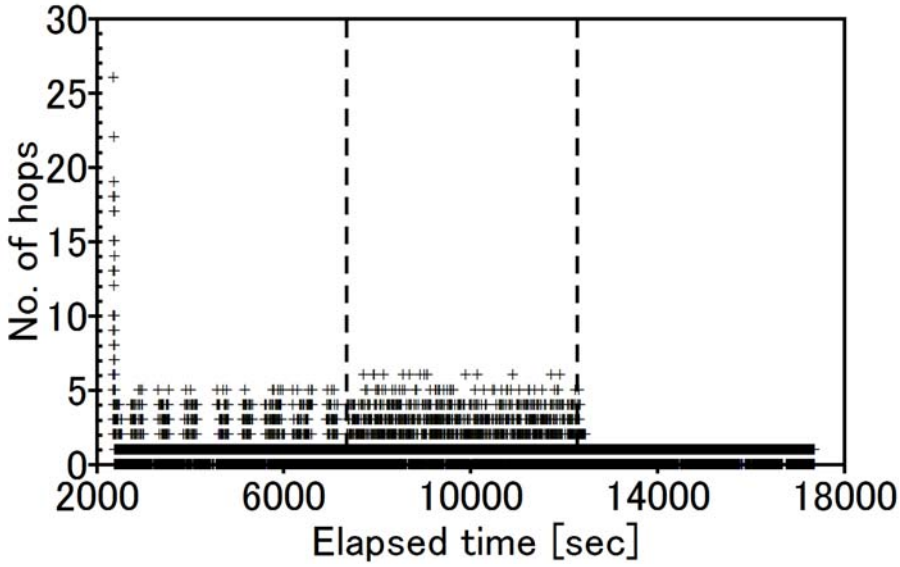


(b) CLS(Meridian)

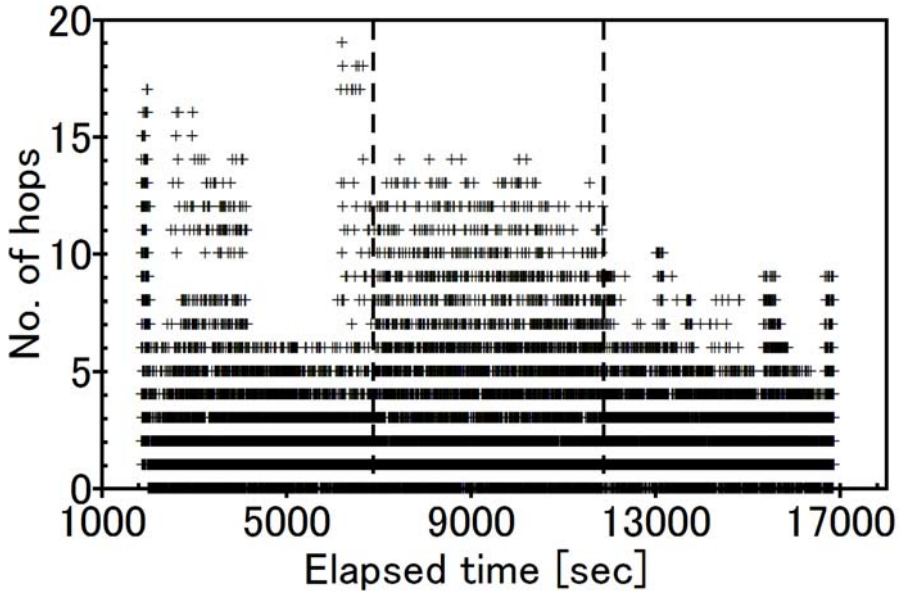
図 4.11: **需要発生地域が移動する状況における EPSR が設定したマーカ数の推移．**

い値を示した．これは他の区間に比べて，この区間の需要の発生地域にあるホスティングマシンが少なかったためである．これによりアクセス経路が他に比べて長くなり，ホップ数が大きな値を示した．

図 4.13 はクライアントと最寄りのマーカ間の RTT を示す．縦の破線はクライア



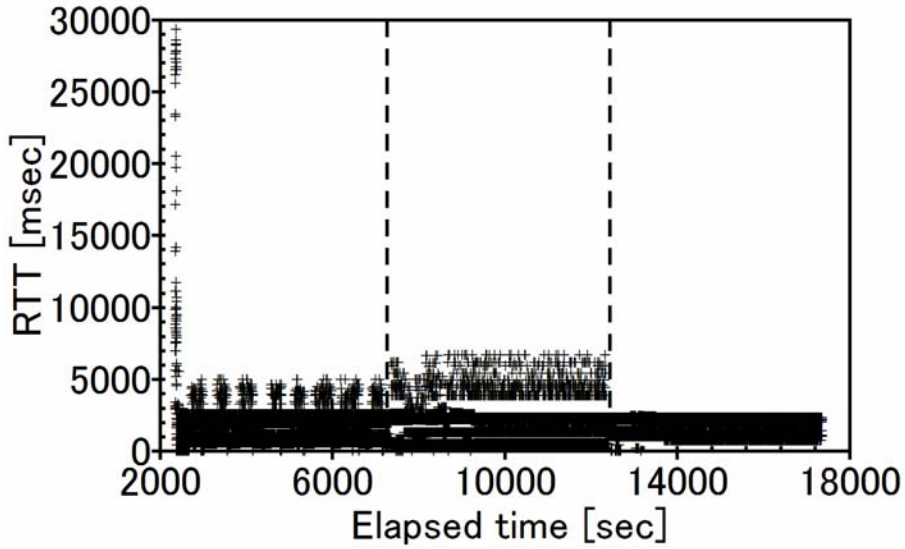
(a) CLS(TS)



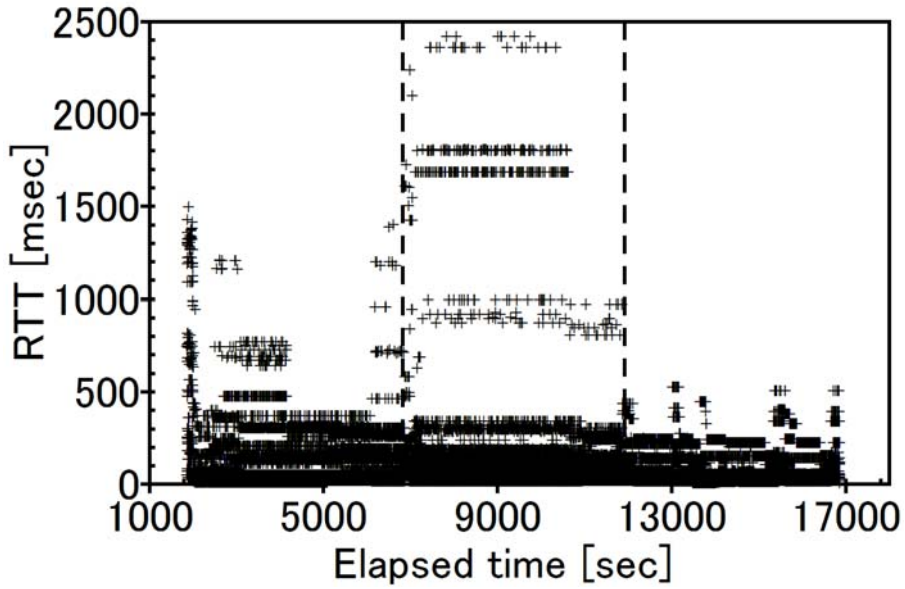
(b) CLS(Meridian)

図 4.12: **需要発生地域が移動する状況で EPSR を使用した際のホップ数の推移．**

ントのクラスタを変更した時点を示す．図 4.13 (a) は CLS(TS) の場合を，図 4.13 (b) は CLS(Meridian) における測定結果を示す．EPSR が選択したマーカとクライアントとの平均 RTT は，クライアントのクラスタの変更に追従して小さくなるように変動している．これは EPSR が物理ネットワークトポロジの観点からも，ク



(a) CLS(TS)



(b) CLS(Meridian)

図 4.13: **需要発生地域が移動する状況で EPSR を使用した際の RTT の推移．**

ライアントの地理的分布の移動に応じてマーカを決定できていることを示す．もし EPSR がクライアントの分布の移動に追従できていなければ，クライアントのクラスタが変更された時点で RTT が大きく上昇するはずである．シミュレーションでは CLS(Meridian) の 2 番目の区間を除いて，EPSR は RTT が小さくなるようにマーカを選択した．CLS(Meridian) の 2 番目の区間は他に比べて高い RTT を示

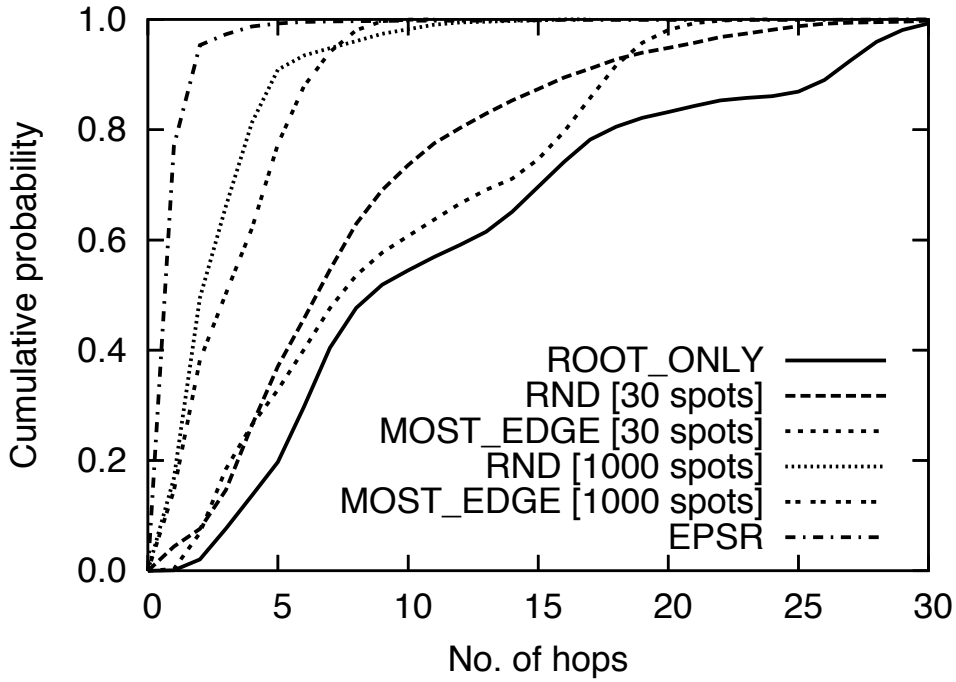


図 4.14: **ホップ数の累積頻度グラフ（CLS(TS) の場合）．**

した．これはホップ数の場合と同様に，他の区間に比べてこの区間の需要の発生地域にあるホスティングマシンが少なかったためである．

EPSR によって選択されたマーカが他の手法によって選択されたものよりも需要の発生地域の移動を高精度で特定できていることを確認するために，第 4.3.1 節と同様に RND，MOST_EDGE および ROOT_ONLY との比較実験を行った．RND および MOST_EDGE は CLS(TS) で 30 台，CLS(Meridian) で 6 台のホスティングマシンをマーカとして選択した．これは同条件下で EPSR が CLS(TS) の場合に最大で 30 台のホスティングマシンを，CLS(Meridian) の場合に最大で 6 台のホスティングマシンをマーカとして決定したためである．なお，CLS(Meridian) は Meridian データセットのネットワークトポロジの詳細が不明であるため，MOST_EDGE は実施していない．

図 4.14 は CLS(TS) を用いたシミュレーションにおいて最初に選択したクライアントクラスタにおけるホップ数の分布を示す累積頻度グラフである．EPSR は RND，MOST_EDGE と比較して，ホップ数が小さくなるようなホスティングマシンをマーカとして選んでいる．CLS(TS) における EPSR が決定したマーカとクライアントとのホップ数の 90 パーセンタイル値は 2 ホップであった．一方 RND の

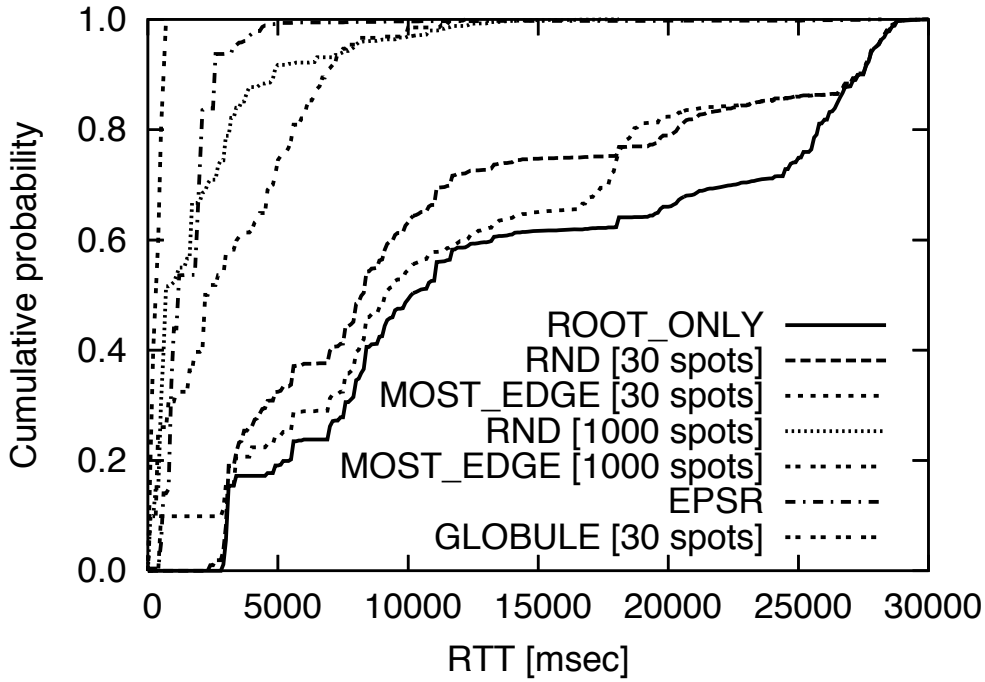


図 4.15: **RTT の累積頻度グラフ（CLS(TS) の場合）．**

90 パーセンタイル値は 17 ホップ，MOST_EDGE の 90 パーセンタイル値は 18 ホップであった．また，ROOT_ONLY の 90 パーセンタイル値は 27 ホップであった．

図 4.15 は CLS(TS) を用いたシミュレーションにおいて最初に選択したクライアントクラスタにおける RTT の分布を示す累積頻度グラフである．ホップ数の場合と同様に，EPSR は RND，MOST_EDGE と比較して RTT が小さくなるようなホスティングマシンをマーカとして選んでいる．EPSR が決定したマーカとクライアントとの RTT の 90 パーセンタイル値は 2,600 ミリ秒であった．一方 RND および MOST_EDGE の 90 パーセンタイル値は 27,300 ミリ秒であった．また，ROOT_ONLY の 90 パーセンタイル値は 27,400 ミリ秒であった．

図 4.16 および図 4.17 は CLS(Meridian) を用いたシミュレーションにおいて最初に選択したクライアントクラスタにおけるホップ数と RTT の分布を示す累積頻度グラフである．CLS(TS) の場合と同様に，EPSR が決定したマーカは RND や ROOT_ONLY よりもクライアントに近いホスティングマシンとなっている．

比較のため，RND と MOST_EDGE が決定するマーカの数を CLS(TS) の場合で 30 から 1000 に，CLS(Meridian) の場合で 6 から 500 台に増加させた際のホップ数と RTT も合わせて計測した．測定結果は図 4.14，図 4.15，図 4.16 および図 4.17

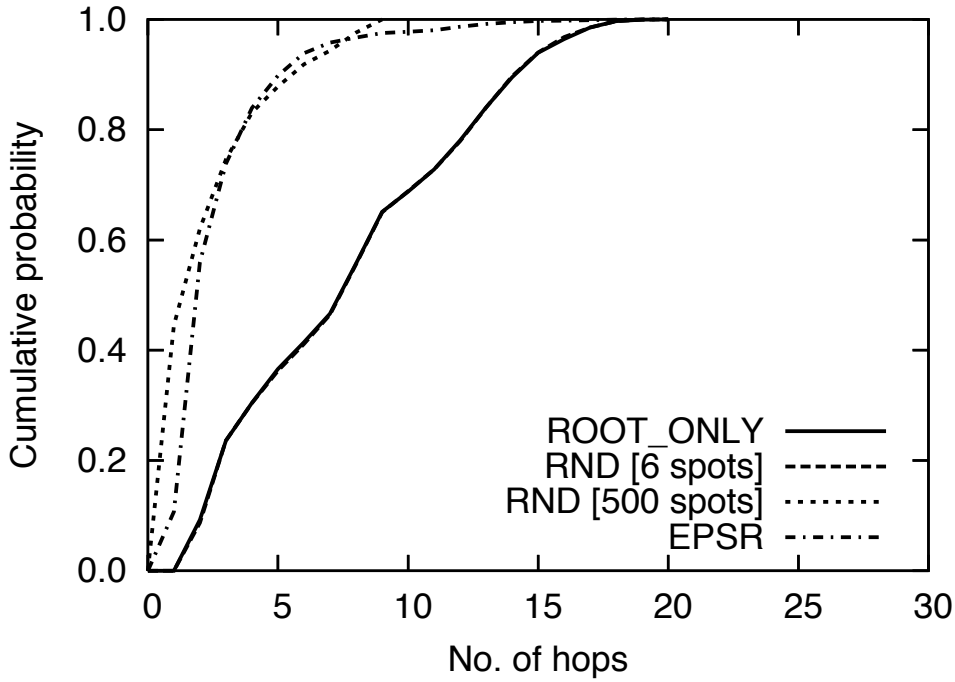


図 4.16: **ホップ数の累積頻度グラフ（CLS(Meridian) の場合）.**

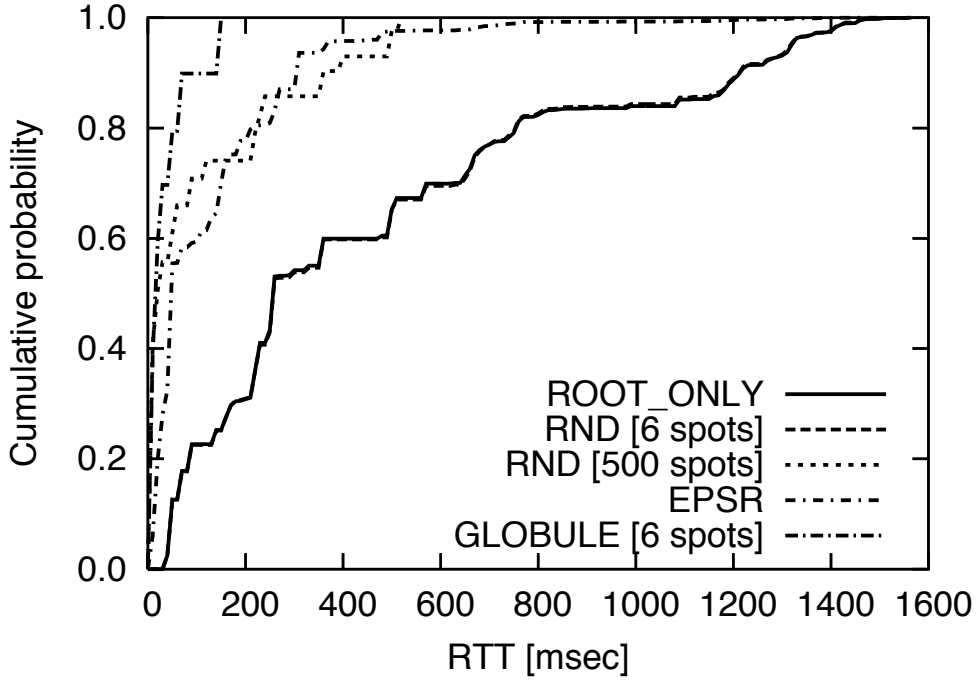


図 4.17: **RTT の累積頻度グラフ（CLS(Meridian) の場合）.**

に示す．CLS(TS) の場合で約 33 倍，CLS(Meridian) の場合で約 83 倍のマーカを設定しているにもかかわらず，RND および MOST_EDGE のホップ数および RTT の 90 パーセンタイル値は EPSR のホップ数および RTT と同等かそれ以上であった．

EPSR が適切にマーカを決定できているか確認するために，最適解に近いマーカの選択を行う GLOBULE との比較を第 4.3.1 節と同様に行った．EPSR が設定したマーカ数の最大値だけ GLOBULE にマーカを決定させた．すなわち，CLS(TS) で 30 台，CLS(Meridian) で 6 台のマーカを決定した．図 4.15 および図 4.17 が示すように，EPSR は RND や MOST_EDGE よりも GLOBULE に近い性能を示した．

本節のシミュレーション結果は第 4.3.1 節のものと比較して，より EPSR の性能が良いという結論を導き出している．これは第 4.3.1 節のシミュレーションの条件と比較して，RND や MOST_EDGE が良いマーカを設定できる確率が低いためである．本節のシミュレーションはクライアントを一部の地域に限定している．そのため，ホップ数や RTT が小さくなるようなホスティングマシンの数は第 4.3.1 節のシミュレーションと比較して少なくなる．これにより，RND や MOST_EDGE のようなアドホックな手法がホップ数や RTT が小さくなるホスティングマシンを選択できる確率が低くなる．

4.3.2 ネットワーク座標系，VPE および VDR の効果

EPSR が需要発生地域の特定に利用している要素技術の効果を確認するために，EPSR を 5 つの異なる設定条件下でシミュレーションを行った．シミュレーションではホップ数と RTT の計測を行った．需要発生地域の特定に利用している要素技術とは，ネットワーク座標系，Virtual Path Expanding (VPE)，Virtual Demand Rasing (VDR) の 3 つである．用意した設定条件は次の通りである．

- all-off: 上記 3 つの要素技術を全て無効にしたもの．
- GNP: ネットワーク座標系によるオーバレイネットワークの構築を有効にしたもの．
- VPE & GNP: GNP に加えて，VPE を有効にしたもの．
- VDR & GNP: GNP に加えて，VDR を有効にしたもの．
- all-on: 上記 3 つの要素技術を全て有効にしたもの．

表 4.4: **ネットワーク座標系，VPE および VDR の効果の評価で使用した各種パラメータ．**

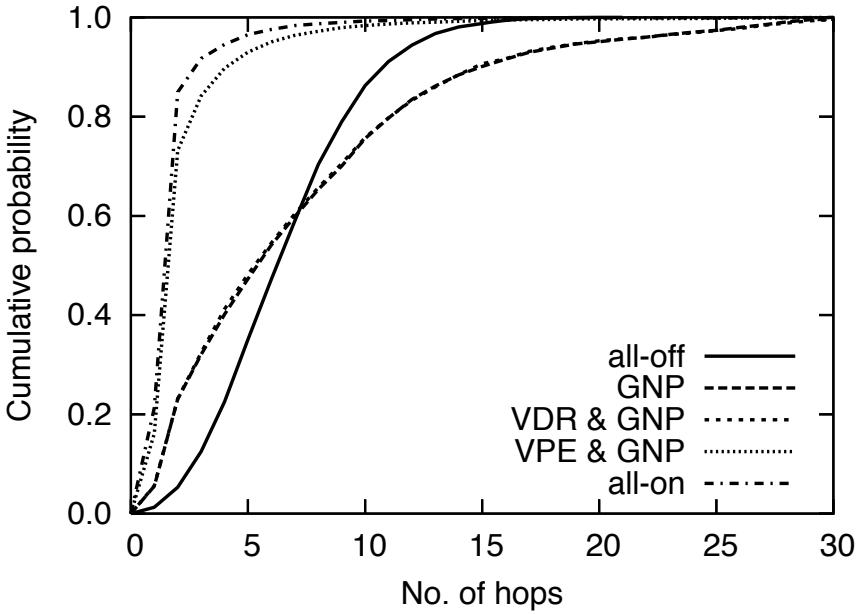
サービス数	1
クライアントの選択間隔	500 ミリ秒
スライディングウィンドウのサイズ	300 秒分
APC degree の確認間隔	60 秒
マーカとなる閾値 $upper$	200
マーカから戻る閾値 $lower$	150

本節のシミュレーションでは，TS データセットを用い，500 ミリ秒ごとにランダムに選択した終端ノードからクライアントのリクエストを送信した．

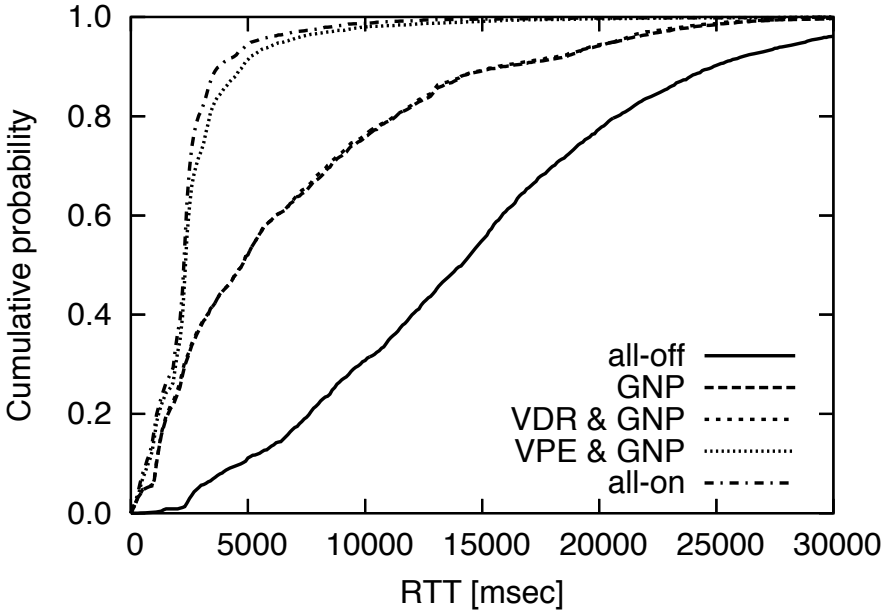
シミュレーションで使用したパラメータを表 4.4 に示す．提供しているサービスの数は 1 とした．APC degree を管理するスライディングウィンドウは 300 秒分とし，各ホスティングマシンは 60 秒間隔で APC degree を確認するように設定した．マーカとなる閾値 $upper$ は 200，マーカから一般のホスティングマシンに戻る閾値 $lower$ は 150 とした．

図 4.18 は各条件下でのホップ数と RTT の分布を示す累積頻度グラフである．ホップ数においては条件 all-off が条件 GNP および条件 VDR & GNP よりも 90 パーセンタイル値で小さい値を示しているが，RTT においては条件 all-off は他のすべての条件よりも大きい値となった．このような結果となる主原因は，条件 all-off では物理ネットワーク上での位置関係がオーバレイネットワーク上で考慮されていないことにある．条件 all-off では CAN のオーバレイネットワーク構築にネットワーク座標系を使用していない．このため，CAN 上で隣接するホスティングマシンが物理ネットワーク上では遠方にあることがある．これよりホップ数は小さいが RTT が大きくなるようなホスティングマシンが存在することになる．一方でネットワーク座標系を用いた場合，構築したオーバレイネットワーク上の位置関係が物理ネットワーク上での RTT の大小関係を反映する．このことから条件 GNP の RTT の分布は条件 all-off よりも良い値を示す．

条件 VPE & GNP は条件 GNP よりもホップ数および RTT が小さくなった．一方で条件 VDR & GNP は条件 GNP のそれとほとんど変わらない結果であった．これは VDR が VPE による効果を促進するものであるためである．EPSR では VPE



(a) ホップ数



(b) RTT

図 4.18: **VPE，VDR および GNP の効果．**

がクライアントのアクセス経路の検知を容易にし，そして VDR がマーカを設定すべき方向の特定を容易にする．これらの特性から，VDR と VPE を組み合わせる

表 4.5: **Dummy Query および Dummy Connection の効果の評価で使用した各種パラメータ.**

サービス数	8
クライアントの選択間隔	500 ミリ秒
スライディングウィンドウのサイズ	300 秒分
APC degree の確認間隔	30 秒
マーカとなる閾値 $upper$	300
マーカから戻る閾値 $lower$	150
ホスティングマシンの性能制約	同時接続数 50 以下

ことでマーカ決定の精度がより高まる．条件 all-on では 3 つ全ての技術が有効となり，クライアントとのホップ数や RTT が条件 VPE & GNP と比較してさらに減少した．

4.3.3 Dummy Query および Dummy Connection の効果

Dummy Query および Dummy Connection の複製サーバ配置先の決定に対する効果を確認するために，Dummy Query および Dummy Connection の有無による EPSR の性能の違いを調査した．本節のシミュレーションは第 4.2 節の条件を基本としている．しかし，Dummy Query および Dummy Connection は需要の規模が大きいほど効果が高いため，Dummy Query および Dummy Connection の効果をより正確に知るために条件を一部変更した．本シミュレーションのパラメータを表 4.5 に示す．本シミュレーションでは 8 のサービスが同時に稼働していることを想定した．マーカとなる閾値 $upper$ は 300，マーカから一般のホスティングマシンに戻る閾値 $lower$ は 150 とした．ホスティングマシン 1 台当たりの性能制約は，クライアントの同時接続数 50 とした．

Dummy Query

第 3.6.1 節で述べたように，Dummy Query は複製サーバの配置先を需要の発生している地域内に止めることに貢献し，需要の発生している地域外へ配置先が移動することを防ぐ．これは Dummy Query によってホスティングマシンは周辺需要

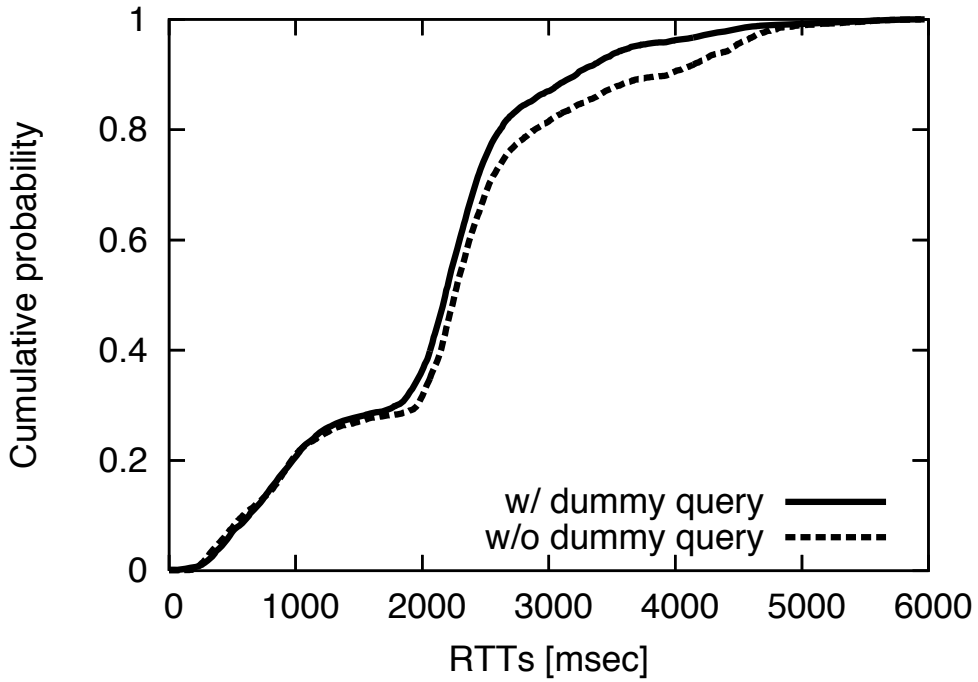


図 4.19: **Dummy Query を有効および無効にした際の RTT の累積頻度グラフ.**

の低下の誤検出を回避できるようになるからである．図 4.19 はクライアントと複製サーバの配置先となったホスティングマシン間の RTT の分布を示す累積頻度グラフである．図 4.19 は Dummy Query を有効にしたものと無効にしたものをそれぞれ記している[4]．Dummy Query を有効にした際の RTT の 90 パーセンタイル値は 3220 ミリ秒であった．一方で Dummy Query を無効にした際の RTT の 90 パーセンタイル値は 3950 ミリ秒であった．この結果は Dummy Query がクライアントの遠方にあるホスティングマシンを複製サーバの配置先とすることを防ぐことを示す.

Dummy Connection

Dummy Connection は複製サーバのクライアント数をホスティングマシンの性能制約以下に止めることに貢献する．これは Dummy Connection が，クライアントアクセス数の急増時にホスティングマシンがクライアントによる負荷を誤って過小評価することを防ぐからである．図 4.20 はホスティングマシンにおけるクライアントの同時接続数を示す累積頻度グラフである．図 4.20 は Dummy Connection

[4]両者とも Dummy Connection は有効にしている.

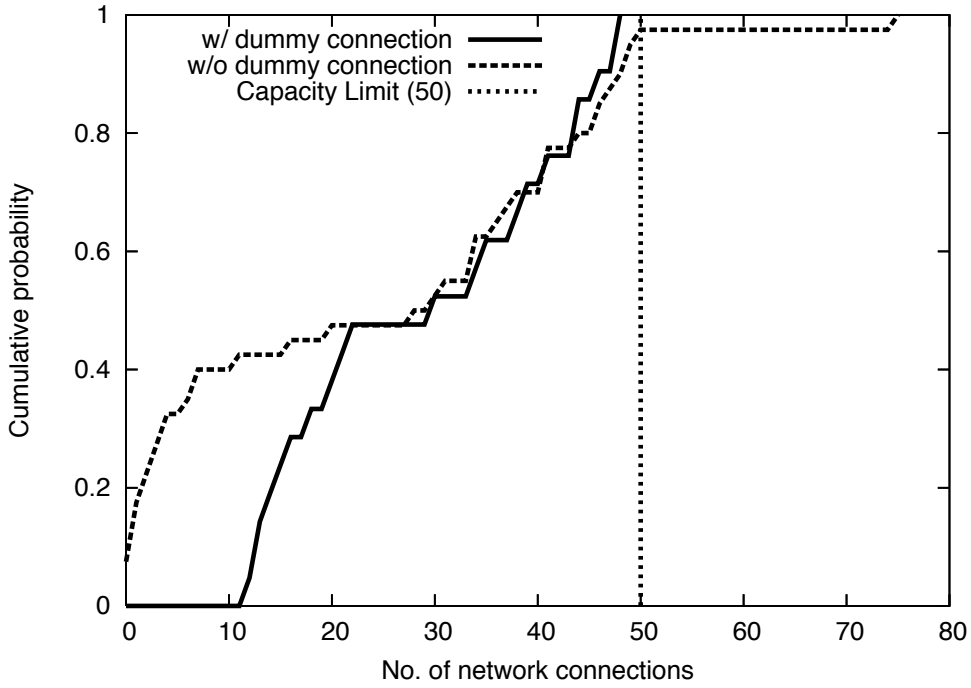


図 4.20: **Dummy Connection を有効および無効にした際の同時接続数の累積頻度グラフ.**（複製サーバを稼働していないホスティングマシンを除く.）

を有効にしたものと無効にしたものをそれぞれ記している[5]．Dummy Connection を有効にした際，EPSR はすべてのホスティングマシンについてクライアントの同時接続数を性能制約である 50 以下に抑えている．一方 Dummy Connection を無効にした際は，配置先となったホスティングマシンのうち 2.5 パーセントが性能制約以上の同時接続を確立した.

さらに図 4.20 は Dummy Connection が，同時接続数の少ない配置先ホスティングマシンを減少させていることを示している．Dummy Connection を無効にした際は 40 パーセントの配置先ホスティングマシンが同時接続数 7 以下であり，また約 8 パーセントの配置先ホスティングマシンが同時接続数 0 であった．一方，Dummy Connection を有効にした際はすべての配置先ホスティングマシンが同時接続数 10 以上となった.

この同時接続数の少ない配置先ホスティングマシンの減少は Dummy Connection の副次的効果である．大量のクライアントリクエストが短時間に送信される状況において，Dummy Connection は複製サーバへの均一な負荷分散を支援する．Dummy

[5]両者とも Dummy Query は有効にしている.

Connection が有効であると，EPSR は大量に発生したクライアントリクエストの処理を開始する前に，リクエストの処理で発生する負荷を予測して複製サーバの配置先を決定できる．また，クライアントリクエストは新たに複製サーバが配置されるホスティングマシンに転送される．これによりクライアントの接続が確立する際は，Dummy Connection による仮想負荷で追加に配置された複製サーバへクライアントが分散されることになる．一方 Dummy Connection が無効であると，多くのクライアントリクエストは一部の配置先ホスティングマシンに集中し，そのホスティングマシンは過負荷となる．また大量のクライアントリクエストによって追加された配置先ホスティングマシンはリクエストの一部のみ処理することになる．これは配置先が決定される前に，既に配置された一部の複製サーバがクライアントリクエストの処理を開始してしまうからである．

4.4 まとめ

本章では EPSR の効果を確認するために行ったシミュレーションについて述べた．効果測定の指標として，複製サーバの配置先数，オーバレイネットワーク上のホップ数，RTT，同時接続数の 4 つについて計測した．またインターネットを想定したシミュレーションを行うために，インターネットモデルのひとつである Transit-Stub モデルによるデータセットと，RTT の実測によるデータセットを用いた．シミュレーションの結果，EPSR が Flash Crowd による影響を軽減するように複製サーバの配置先を決定することを確認した．シミュレーションでは，Flash Crowd 発生時において EPSR が約 3000 台のホスティングマシン上で Flash Crowd の検出と発生地域の特定を行い，配置先の選択を開始するまでの時間は 25 秒であった．これは文献 [10] で報告されている Flash Crowd がピークに達するまでに要した時間 40 秒より高速である．その後 EPSR は不要な複製サーバの配置先を除去し，300 秒で配置先数が安定することを確認した．その際，他方式と比較して EPSR がクライアントとの通信遅延が小さくなるように配置先を決定することと，配置先ホスティングマシンの性能制約を超過しないことを確認した．また，EPSR が使用している要素技術の効果も合わせて確認した．

第5章　議論

本章では，EPSR についての定性的な議論を行う．論点は EPSR の適用範囲，EPSR のパラメータの決定方法，スケーラビリティ，サービス間での公平性，セキュリティの 5 点である．EPSR の適用範囲については，EPSR が適用可能なサービスの条件について述べる．EPSR のパラメータの決定方針については，EPSR が需要発生地域を特定する際に使用するパラメータの決定方針について議論する．スケーラビリティという観点では，ホスティングマシン数やサービス数に対して EPSR がスケーラブルであることを確認する．サービス間での公平性については，EPSR を適用した際に需要の高いサービスと需要の低いサービスとの間で公平性が保たれているか議論を行う．セキュリティについては，EPSR が前提とするホスティングマシン群の条件についてと，DoS 攻撃に対する耐性について述べる．

5.1　EPSR の適用範囲

EPSR は以下の 2 つの特徴をもつサーバを対象とする．1 点目に EPSR は，平常時は需要の少ない単体のサーバに適用可能である．このようなサーバは規模が比較的小さく，複製サーバの起動に必要なファイルをホスティングマシンにあらかじめ配備しておくことが容易である．そのため EPSR を適用することが容易になる．

この対象設定は妥当であると考えられる．平常時の負荷が小さいサーバは負荷分散用の設備を設置していないことが多い．そのため，負荷分散用の設備を設置している規模の大きいサービスと比較して Flash Crowd による影響が大きい．負荷分散用の設備とは，複製サーバ稼働用のハードウェア，余剰ネットワーク，ロードバランサなどである [19]．平常時の需要が単一のサーバで十分対応可能であるならば，このような負荷分散用の設備は不要である．一方，Flash Crowd 発生時は需要が平常時の数倍から数十倍に増加するため，負荷分散用の設備が必要となる．Flash Crowd の発生を予測することは難しいため，負荷分散用の設備を Flash Crowd の発生に合わせて設置することは難しい．もし EPSR がない場合，Flash Crowd に対

応するために負荷分散用の設備を常時設置しておく必要がある．これには余剰なコストが発生する．一方，負荷分散用の設備を設置していない場合は Flash Crowd が発生すると最悪の場合サービスが停止してしまう．

2 点目に EPSR は，クライアントとの通信時に構成要素が変更されないサーバに適用可能である．例えばフロントエンドサーバなどが対象となる．クライアントとの通信時に構成要素が変更されなければ，複製サーバの構成要素を更新することがあっても適用可能である．ただし，その更新間隔が Flash Crowd の発生時間よりも十分長い場合に適用可能である．Flash Crowd の発生時間は数分間から数時間程度であるので [10,12]，想定する更新間隔は最短で数時間程度である．例えば一日間隔のバッチ処理でデータの更新を行うカタログサイトや，数時間単位で気象情報を更新する天気予報サイトなどのサーバが対象となる．更新周期が Flash Crowd の発生時間よりも長期間であれば，複製サーバが読み込み専用のデータベースを備えていてもよい．

この適用範囲の設定は Flash Crowd 対策としては妥当であると考えられる．Flash Crowd 対策における問題は，Flash Crowd による負荷を分散するために急激に増加した複製サーバの一貫性を保たなければならない場合である．更新の問題への単純な対策として，Flash Crowd 発生中は更新を遅延するという方法がある．Flash Crowd の発生時間は一般に数分から数時間程度であり，永続的に続くことはない．一方で Flash Crowd による負荷の急増はサーバの停止を引き起こす．これより，Flash Crowd 発生中の更新の問題よりも，Flash Crowd によるサーバの異常停止によるサービス中断の問題の方が利用者に与える影響は大きいといえる．そのため，更新を遅延してでも EPSR を適用した方が有用であると考えられる．また，クライアントとの通信時に構成要素の更新を行わず，かつ更新間隔が Flash Crowd の発生時間よりも十分長いサーバであれば，Flash Crowd 発生時の更新を回避できる可能性が高い．もし Flash Crowd 発生中に緊急に更新しなければならない場合は，例えば Peer-to-Peer ネットワーク上で効率よく更新通知を伝播する手法 [118] などを併用することで解決を図ることができる．

一方で，クライアントとの通信時に構成要素を更新するサーバや，Flash Crowd の発生時間と比較して更新周期が短いサーバに無変更で EPSR を適用することは難しい．例えば Facebook などのソーシャル・ネットワーキング・サービス (SNS) や Twitter などのコミュニケーション・サービスは，クライアントのアクセス時にサーバ側のデータベースを更新することが多い．こうしたサービスでは Flash Crowd 発生時に頻繁な更新が発生する．そのため複製サーバがもつデータベースの更新が

頻繁に発生し，そのたびに一貫性をとらなければならなくなる．

頻繁な更新を伴うデータベースを含むサーバに対して EPSR を適用するには，データベースを読み込み頻度が高く更新頻度が低い部分と，読み込み頻度よりも更新頻度の方が高い部分とに分割する必要がある．読み込み頻度が高く更新頻度が低い部分は複製サーバに含め再配置の対象とする．更新頻度の高い部分はキーバリューストア [31–35] や分散ストレージ [119–121] などの外部共有媒体に配置することを想定する．キーバリューストアや分散ストレージは同時アクセス数に対するスケーラビリティを備えているため，Flash Crowd に対する負荷分散を行う際にボトルネックとはならない．また，キーバリューストアや分散ストレージはデータアクセス時の入出力による負荷を分散する目的であるため，これらとコンテンツの動的生成時における CPU 負荷やメモリ負荷およびコンテンツ提供時のネットワーク負荷の分散を目的とする EPSR は補完し合う関係にある．なおデータベースの分割は，例えば GlobeTP [122] のようなデータベースクエリの負荷分散技術などを併用することによって可能である．また，更新頻度の高いファイルを使用するサーバに対しても同様に外部共有媒体の使用を想定する．

なお，EPSR は複製サーバを構成するファイル群がホスティングマシンにあらかじめ保存されていることを適用の条件とするため，ホスティングマシンのストレージ容量を消費する．しかし，ストレージは比較的安価に容量を増加できるためこの仮定は現実的であると考えられる．このため EPSR の性能制約にストレージ容量は含めない．一方で CPU やメモリ，ネットワークの増強は比較的高価であるため，これらを EPSR の性能制約に含める．

5.2 EPSR のパラメータの決定方針

EPSR は APC degree の増減を監視することで需要発生地域を特定する．APC degree が閾値 $upper$ を超えるとホスティングマシンは近傍で需要が発生していると判断し，閾値 $lower$ を下回ると需要が消滅したと判断する．閾値 $upper$ と $lower$ の決定方法は様々な要因によって決定することができる．ここではホスティングマシンの配置の分布に基づく方針と，サービスへの課金に基づく方針の 2 種類について述べる．

ホスティングマシンの配置の分布による方針では，EPSR が構築するネットワーク座標に基づいたオーバレイネットワークにおけるホスティングマシンの配置密

度に基づいて閾値を決定する．ネットワーク座標空間においてホスティングマシンの配置の密度が高い部分はアクセス経路の分散が大きいため，APC degree が上昇しにくい．よって閾値 $upper$ と $lower$ は低く設定するべきである．反対にホスティングマシンの配置の密度が低い部分はアクセス経路が集中しやすいため，APC degree が上昇しやすい．よって閾値 $upper$ と $lower$ は高く設定するべきである．

サービスへの課金に基づく方針では，サービスに対して課金した金額の大小に応じて閾値を決定する．課金額の大きいサービスはクライアントとの通信遅延を小さくするものとし，需要の発生地域をより細かい単位で特定するようにする．よって課金額の大きいサービスでは閾値 $upper$ と $lower$ を低く設定する．一方課金額の小さいサービスはクライアントの通信遅延が大きくても良いものとし，需要の発生地域を粗い単位で特定するようにする．よって課金額の小さいサービスでは閾値 $upper$ と $lower$ を高く設定する．

また，閾値に APC degree の値を直接用いるのではなく，APC degree の変化量を用いることも可能である．APC degree が過去のパターンに比べて有意に上昇している場合，近傍の需要が増加中であるとみなすことができる．反対に APC degree が有意に減少している場合，近傍の需要が減少中であるとみなすことができる．APC degree の変化は，例えば機械学習によるパターン認識によって上昇傾向であるか減少傾向であるか判断できる．APC degree の変化量を用いる場合でも，APC degree の値を直接用いる場合と同様にホスティングマシンの配置の分布や課金に基づく判定を組み合わせることが可能である．

5.3 スケーラビリティ

5.3.1 ホスティングマシン数に対するスケーラビリティ

EPSR はホスティングマシンの数に対してスケーラブルである．これは EPSR の動作に要する通信トラフィックはホスティングマシン数 N とネットワーク座標系の次元数 d に対して $O(N^{\frac{1}{d}})$ で増加するためである．ここではホスティングマシンの増加によって転送数が増加するサーバ解決要求メッセージ，VPE メッセージ，複製サーバ起動依頼メッセージ，Dummy Query の 4 種類のメッセージについて，それぞれのメッセージの転送量を導出する過程について述べる．

サーバ解決要求メッセージの転送量

クライアントがサーバ解決要求メッセージを 1 つ送信すると，N 台のホスティングマシンで d 次元の CAN を構成する ExaPeer では，ルートであるホスティングマシンに辿り着くまでのサーバ解決要求メッセージの転送回数は文献 [79] より

$$O\left(\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)\right)$$

になる．次にメッセージの送信 1 回当たりのメッセージの総量について考える．クライアントの送信するサーバ解決要求メッセージに含まれる情報は第 3 章で述べたように，メッセージの種類を表す 4 ビット値，クライアントの IP アドレス，サービス名から計算した宛先座標，クライアントが最初に送信したホスティングマシンの IP アドレスである．IP アドレスは IPv4 の場合 4 バイト，座標は軸あたり 64 ビットの整数値で表すとすると d 次元の場合 $8d$ バイトになる．これらから，サーバ解決要求メッセージ 1 つは $(9+8d)$ バイトとなる．よって，1 回当たりのサーバ解決要求メッセージの総量は

$$(9+8d)\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)$$

バイトである．ここで文献 [73] によるネットワーク座標系の精度の報告に従い $d=6$ と設定すると，メッセージの総量は

$$85.5\left(N^{\frac{1}{6}}\right)$$

バイトになる．

VPE メッセージ

クライアントがサーバ解決要求メッセージを 1 つ送信すると，転送したホスティングマシンによって VPE メッセージが送信される． N 台のホスティングマシンで d 次元の CAN を構成する ExaPeer では，ルートであるホスティングマシンにサーバ解決要求メッセージが到達するまでに送信される VPE メッセージの数は文献 [79] より

$$O\left((2d-2)\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)\right)$$

になる．次にサーバ解決要求メッセージの送信 1 回当たりの VPE メッセージの総量について考える．VPE メッセージに含まれる情報は第 3 章で述べたように，メッ

セージの種類を表す 4 ビット値のみを含む．これより VPE メッセージ 1 つは 1 バイトとなる．よって，クライアントがサーバ解決要求メッセージを 1 つ送信すると，VPE メッセージの総量は $d = 6$ のとき

$$15\left(N^{\frac{1}{6}}\right)$$

バイトとなる．

複製サーバ起動依頼メッセージ

複製サーバ起動依頼メッセージはサーバ解決要求メッセージと同様に，ルートマシンを宛先として転送される．よって転送回数はサーバ解決要求メッセージと同様に

$$O\left(\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)\right)$$

となる．次に複製サーバ起動依頼メッセージの送信 1 回当たりの総量について考える．複製サーバ起動依頼メッセージに含まれる情報は第3 章で述べたように，メッセージの種類を表す 4 ビット値，サービス名から変換した宛先座標，自身の GNP 座標から送信先ゾーンの中心座標への方向ベクトルを含む．方向ベクトルは座標と同じ情報量となるので，d 次元の場合 $8d$ バイトとなる．よって複製サーバ起動依頼メッセージ 1 つは $(1 + 16d)$ バイトとなる．これより複製サーバの起動依頼 1 回当たりの複製サーバ起動依頼メッセージの総量は

$$(1 + 16d)\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)$$

バイトである．サーバ解決要求メッセージと同様に $d = 6$ と設定すると，メッセージの総量は

$$145.5\left(N^{\frac{1}{6}}\right)$$

バイトとなる．

Dummy Query

Dummy Query はサーバ解決要求メッセージと同様に，ルートマシンを宛先として転送される．よって転送回数はサーバ解決要求メッセージと同様に

$$O\left(\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)\right)$$

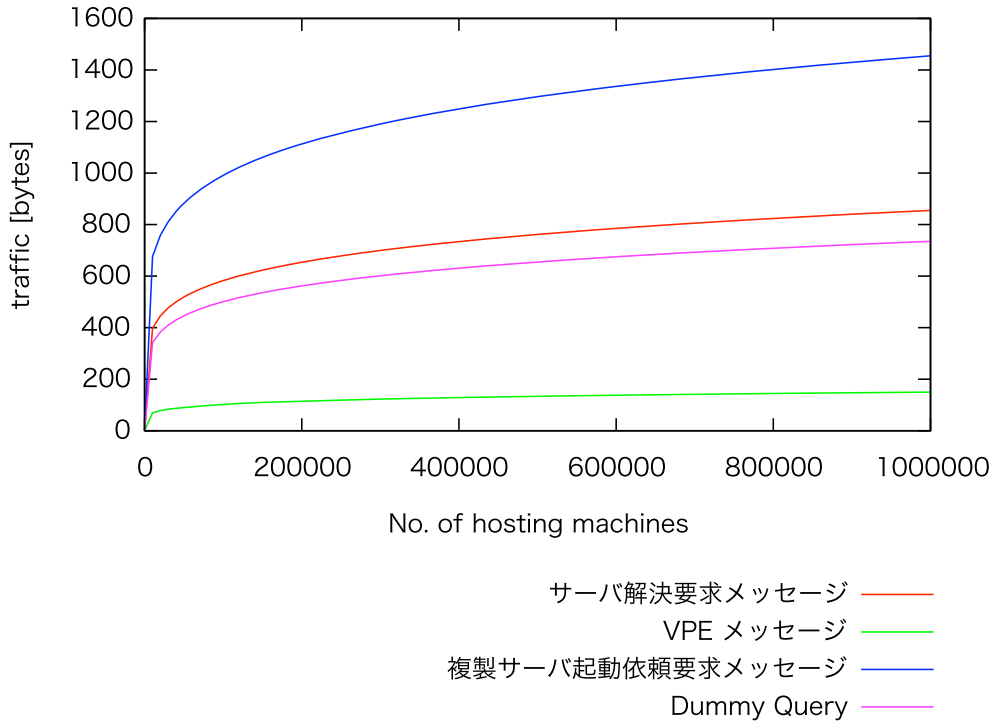


図 5.1: **各メッセージのトラフィック量とホスティングマシン数との関係．**

となる．次に Dummy Query の送信 1 回当たりの総量について考える．Dummy Query に含まれる情報は第 3 章で述べたように，メッセージの種類を示す 4 ビット値とサービス名から変換した宛先座標を含む．よって Dummy Query 1 つは $(1 + 8d)$ バイトとなる．これより Dummy Query の送信 1 回当たりの総量は

$$(1 + 8d)\left(\frac{d}{4}\right)\left(N^{\frac{1}{d}}\right)$$

バイトである．サーバ解決要求メッセージと同様に $d = 6$ と設定すると，メッセージの総量は

$$73.5\left(N^{\frac{1}{6}}\right)$$

バイトとなる．

図 5.1 に，ホスティングマシン数とサーバ解決要求メッセージ，VPE メッセージ，複製サーバ起動依頼メッセージおよび Dummy Query の 1 回当たりの送信における総量との関係を示す．いずれのメッセージの総量もホスティングマシン数 N に対して $O(N^{\frac{1}{d}})$ である．例えば $N = 1,000,000$ のとき，サーバ解決要求メッセージ，VPE メッセージ，複製サーバ起動依頼メッセージおよび Dummy Query

の 1 回当たりの送信における総量はそれぞれ 855 バイト，150 バイト，1455 バイトおよび 735 バイトとなる[1].

5.3.2 サービス数に対するスケーラビリティ

また，EPSR はサービス数の増加に対してスケーラビリティの高い設計である．ホスティングマシンが提供中のサービスごとに管理する情報は，提供するサービス数の増加に伴って増加する．しかし，EPSR はサービスごとの管理情報を APC degree のスライディングウィンドウという単純な整数値の配列のみとしているため，サービス数の増加による影響は小さい．例えばそれぞれのホスティングマシンが管理する APC degree のスライディングウィンドウが 300 秒分であった場合，サービスごとの管理情報の大きさは高々 1.2 キロバイトとなる．

5.4 サービス間での公平性

EPSR は需要の高いサービスに対して複製サーバの配置先数を増加させる．これは EPSR が需要の高いサービスを優先し，需要の低いサービスの品質を落としているように見える．しかし EPSR は需要の低いサービスの品質を落とすのではなく，反対に Flash Crowd によって需要が増加したサービスの影響から需要の低いサービスを保護するように EPSR は動作しているといえる．EPSR は需要の高い地域にあるホスティングマシンを複製サーバの配置先とする．需要の発生源に近い位置に配置された複製サーバによって，ホスティングマシンの負荷が減少する．その結果，他の需要の高いサービスによる需要の低いサービスの応答性の低下が軽減される．

5.5 セキュリティ

一般的な公開型 Peer-to-Peer (P2P) システムと異なり，EPSR は信頼できるホスティングマシン上で動作することを想定する．一般的な公開型 P2P システムでは，任意のユーザが認証なしにマシンをオーバレイネットワークへ追加することがで

[1]なお，実際の通信ではメッセージのヘッダ等のオーバヘッドを含むため，通信量はこれより大きい．

きる．一方 EPSR ではオーバレイネットワークは非公開である．EPSR のオーバレイネットワークにホスティングマシンを追加する場合，例えばチャレンジ／レスポンス方式などによって認証を行う必要がある．よって，EPSR は不正なホスティングマシンへの対策は行わない．もちろん ExaPeer の実装によっては脆弱性が存在しホスティングマシンがクラックされる可能性はある．これは本研究の対象外である．

EPSR は Denial of Service (DoS) 攻撃に対して耐性がある．これは DoS 攻撃による負荷上昇は Flash Crowd と同質であるためである [10]．EPSR は短時間に発生する急激な需要の上昇による影響を軽減するため，DoS 攻撃による急激な負荷上昇も Flash Crowd と同様の対応によって影響を回避できる．ただし，攻撃者が ExaPeer を介さず直接ホスティングマシンにアクセスすることが可能であった場合，EPSR による負荷分散は行われないため DoS 攻撃の影響を受けてしまう．さらに DoS 攻撃への耐性を高めるためには，例えば dFence [123] のような DoS 攻撃対策をホスティングマシンへ導入する必要がある．

5.6 まとめ

本章では EPSR の定性的な議論を行った．EPSR の適用範囲，EPSR のパラメータの決定方針，スケーラビリティ，サービス間での公平性，セキュリティの 5 点について論じた．

EPSR が適用可能な条件は，1) 平常時は単一のサーバで提供可能な規模の小さいサービスであること，2) 複製サーバの構成要素の構成頻度が Flash Crowd の発生時間に比べて十分長いサービスであること，の 2 点である．規模の小さいサービスは平常時は負荷が低いため負荷分散を行わないことが多く，Flash Crowd 発生時の影響が大きい．一方で Flash Crowd の発生時間は数十分から数時間程度であるため，複製サーバの構成要素の更新頻度が数時間から数日単位であれば適用可能である．また，更新頻度の高い要素は既存手法を用いて分割することができる．以上からこの制限は Flash Crowd 対策としては妥当であると考えられる．

EPSR が需要発生地域の特定に用いるパラメータは様々な条件から決定できる．例えばホスティングマシンの配置の分布密度に基づく方針や，ホスティングするサービスへの課金に基づく方針がある．また，本論文で用いた APC degree の値を直接用いる方法に対して，APC degree の変化量を用いる方法もある．

EPSR はホスティングマシン数に対してスケーラブルである．EPSR はホスティングマシン間でクライアントのリクエストを転送する仕組みであるため，クライアントのリクエスト当たりの通信量がホスティングマシンの数に応じて増加する．EPSR は通信量を低く抑えるために，転送するメッセージのサイズを小さくしている．100 万台のホスティングマシンがあった場合，EPSR が使用するメッセージの総量は 1 送信当たり高々 1455 バイトである．また，EPSR はホスティングするサービス数に対してもスケーラブルである．ホスティングマシンがサービス 1 つ当たりに管理する情報量は高々1.2 キロバイトである．

EPSR は需要の高低に関わらず，ホスティング中のサービスの質が向上するように複製サーバの配置先を決定する．あるサービスの需要が増加すると，EPSR は需要の高いサービスに対して複製サーバの配置先を増加する．これは需要の高い複製サーバによってホスティングマシンが過負荷になることを回避するのと同時に，需要の低い複製サーバが過負荷なホスティングマシン上で動作することで応答性やスループットが悪化することを防ぐ．これにより，需要の高いサービスだけでなく，需要の低いサービスに対しても品質が悪化することを EPSR は防ぐ．

セキュリティについて，EPSR は認証によって信頼できるホスティングマシン上で動作することを仮定する．また EPSR は潜在的に DoS 攻撃に対して耐性を備える．これは DoS 攻撃による負荷の上昇が Flash Crowd と本質的に同一視できるからである．そのため EPSR による負荷分散によって DoS 攻撃が発生した際のサービス停止を回避することができる．EPSR によって構築したオーバレイネットワークを用いず直接ホスティングマシンに対して DoS 攻撃を実行された場合は，既存の DoS 攻撃対策手法を応用することが可能である．

第6章　結論

6.1　本研究のまとめ

インターネットの利用者数は年々増加傾向にあり，コンテンツの配信にかかる負荷が増加し続けている．それに伴って，安定したサービスの提供を阻害する Flash Crowd と呼ばれる現象がインターネット上で発生するようになった．Flash Crowd が発生すると，インターネットサーバのアクセス数が数分のうちに通常時の数倍から数十倍に達し，サーバは過負荷状態になる．Flash Crowd の発生要因は多岐に渡るため，Flash Crowd の発生を予測することは困難である．

大量のクライアントアクセスを処理する一般的な方法として，複製サーバの配置による負荷分散が知られている．しかし，Flash Crowd 対策用に複製サーバを常時稼働させておくためには，Flash Crowd の規模をあらかじめ予測しておかなければならない．また，Flash Crowd の生じていない平常時には余剰サーバが多くなり，サーバの運用コストが増大するという問題がある．

本論文では，Flash Crowd による影響を軽減する複製サーバの配置先決定手法 ExaPeer Server Reposition (EPSR) を提案した．EPSR は複製サーバの実行用としてあらかじめ世界中に設置された共有ホスティングマシン群で動作する．EPSR はこのホスティングマシン群に対して，サービスの需要に応じて複製サーバの配置先を動的に決定する機能を提供する．これにより，Flash Crowd の規模の予測を必要とせずに Flash Crowd への対策が行える．また，需要変動に応じた動的な複製サーバの配置ができるため，余剰サーバの運用コストが生じない．

Flash Crowd の影響を軽減するように複製サーバの配置先を決定するには，次の 3 つの課題があった．第一に，Flash Crowd によるアクセス増加がピークに達するまでの数分以内に配置先を決定しなければならない（高速性）．第二に，通信負荷の分散とサービス遅延の低減のために，配置先とクライアントとの通信遅延が小さくなるように配置先を決定するべきである（通信遅延）．第三に，複製サーバによってホスティングマシンが過負荷にならないように，複製サーバの配置先を決

定しなければならない（性能制約）.

第 2 章で述べたように，複製サーバの配置先を決定する手法は，1) 固定配信網方式，2) オフライン計算方式，3) オンデマンド負荷軽減網方式，4) 通信遅延考慮型配信網方式の 4 方式に大別される．しかし，いずれの方式も先に挙げた 3 つの課題を全て解決することは難しい．固定配信網方式は静的に構成したネットワーク上で複製サーバの配置先を決定するものであり，高速性は満たすが，通信遅延および性能制約を満たすことが難しい．オフライン計算方式はクライアントの分布情報とホスティングマシンの設置場所の情報から，複製サーバの配置先の決定を最適化問題としてモデル化して解くもので，通信遅延と性能制約は満たすが高速性を満たすことが難しい．オンデマンド負荷軽減網方式はクライアントの増加に応じて負荷分散用にオーバレイネットワークをホスティングマシン間で構築するもので，高速性は満たすが性能制約への考慮は限定的であり，また通信遅延は満たすことが難しい．通信遅延考慮型配信網方式は通信遅延を考慮した負荷分散用オーバレイネットワークを構築することで高速性を満たしながら通信遅延を限定的に削減するが，性能制約は満たすことが難しい.

EPSR は複製サーバの配置に関する 3 つの課題の全てを解決する．EPSR は大域情報を必要としない Peer-to-Peer 技術を用いることで急激な需要の増加に対応する．個々のマシンは自律的に需要の変動を検出し，必要に応じて複製サーバの配置先候補となる．個々のマシンでの需要変動の検出と需要の発生地域の特定を実現するために，EPSR は物理ネットワークのトポロジを考慮したオーバレイネットワークを，ネットワーク座標系と分散ハッシュテーブルを組み合わせて構築する．個々のマシンは，オーバレイ上で隣接するマシンとのみメッセージを交換することでクライアントのアクセス経路を分析し，サービスへの需要が増加もしくは減少している地域を特定する．マシンはアクセス経路の分析から複製サーバの実行による効果を見積もり，配置先の候補となるかどうか決定する．この際，個々のマシンは複製サーバの負荷とマシンの許容負荷を元に，マシンが過負荷にならないように隣接するマシンから追加の配置先候補を独立して選択する.

シミュレーションを行い，EPSR が Flash Crowd に対して有効であることを確認した．シミュレーションでは，約 3000 台のホスティングマシンに対して EPSR が 25 秒で Flash Crowd の検出と発生地域の特定を行い，配置先の選択を開始することを確認した．また複製サーバの配置先数が 300 秒で安定することを確認した．その際，EPSR は他方式と比較してよりクライアントとの通信遅延が小さくなるようなホスティングマシンを配置先として選出した．またシミュレーション結果は，

EPSR が他方式よりも少ない複製サーバで，どのホスティングマシンも過負荷にならないように配置先が決定できることを示した．

6.2 今後の展望

EPSR によって Flash Crowd のような急激なクライアント数の増加に対しても，負荷分散に効果的な複製サーバの配置先を決定することが可能になった．以下では，今後の研究の方向として EPSR の応用例と今後の課題について述べる．まず応用例として，Flash Crowd への耐性の高いマッシュアップサービス提供システム，通信遅延に関する Service Level Agreement (SLA) を考慮した web 配信システムの実現支援，地理的情報に依存するサービスの提供支援システムについて述べる．最後に，今後の複製サーバ配置先決定手法が目指すべき指針について述べる．

6.2.1 EPSR の応用

マッシュアップサービス提供システム

EPSR は Flash Crowd への耐性の高いマッシュアップサービス提供システムの構築に用いることができる．マッシュアップサービスとは，地図や動画といった複数の小規模な web サービスを組み合わせてひとつの大規模な web サービスを提供するサービスのことである．マッシュアップによって提供される web サービスは複数の web サービスの性能に依存することになる．よって Flash Crowd への耐性を高めるためには，マッシュアップに使用する全ての web サービスが Flash Crowd への耐性を備えている必要がある．特にマッシュアップに使用される web サービスは複数の web サイトからアクセスされるためクライアントのアクセスが集中しやすく，また負荷の変動が予測しにくい．

EPSR を利用してマッシュアップに使用する個々の web サービスを提供することで，マッシュアップサービスの Flash Crowd への耐性を高めることができる．EPSR によってマッシュアップを使用するサービスおよびマッシュアップの構成要素となるサービスの複製サーバの配置先を，需要の変動に応じて決定することができる．これはマッシュアップを使用する側の複製サーバとマッシュアップサービスを提供する側の複製サーバの両方を EPSR によって配置先を決定することで実現できる．マッシュアップを使用する側のサービスは，クライアントの分布に応じて

複製サーバの数と位置を決定する．マッシュアップサービスを提供する側のサービスは，EPSR によって決定されたマッシュアップを使用する複製サーバの分布に応じて複製サーバの数と位置を決定する．これにより Flash Crowd による負荷をマッシュアップを使用するサービスだけでなく，マッシュアップを提供するサービスに対しても分散することができる．

通信遅延に関する Service Level Agreement を考慮した web 配信システムの実現支援

EPSR は通信遅延に関する Service Level Agreement (SLA) を考慮した web 配信システムに応用することができる．通信遅延に関する SLA を考慮した web 配信システムは例えば，通信遅延の平均値や最悪値を保証しなければならない．EPSR はこうした SLA の実現を支援することができる．EPSR は性能制約を満たすために，ホスティングマシンの性能限界をパラメータとして与えている．これを SLA の条件に置き換えることで，EPSR は SLA を考慮して複製サーバの配置先を決定するようになる．例えば，クライアントとの通信遅延の平均値をパラメータとして設定すれば，複製サーバの配置が通信遅延の平均値に対する SLA を考慮したものとなる．これは複製サーバに接続しているクライアントの通信遅延が設定値を超過する場合，EPSR は APC degree の大小比較によって，通信遅延を小さくするような近傍のホスティングマシンを新たな複製サーバの配置先とできるからである．通信遅延の最悪値に関しても，最悪値を EPSR のパラメータとして設定し，配置先の決定時に参照すればよい．

地理的位置情報に依存するサービスの提供支援システム

EPSR は地理的情報に依存するようなサービスの提供システムの Flash Crowd 耐性を高めることができる．地理的位置情報に依存するサービスとは，例えばクライアントを Global Positioning System (GPS) による測位情報で分類し，分類された区分に合わせて広告などを提供するサービスを指す．このようなサービスの場合，一部の地域に大勢の人が集中した場合，その地区にサービスを提供するサーバが Flash Crowd を受け過負荷となる恐れがある．EPSR を用いることで地理的に依存する種類の Flash Crowd にも対応できる．まず，EPSR のオーバレイネットワークを，ネットワーク座標によるネットワークトポロジに基づくものから，地理的位

置情報に基づくものに変更する．これにより EPSR はクライアントの地理的位置情報に基づいて複製サーバの配置先を決定することができる．また，オーバレイネットワーク上の位置関係は地理的位置関係を反映させたものとなるため，特定の地域向けのサービスを提供する複製サーバを EPSR の仕組みを用いて，その地域の需要に合わせて増減することが可能になる．

6.2.2 今後の複製サーバ配置先決定手法

EPSR によって Flash Crowd による影響を軽減するように複製サーバの配置先を決定できることを示した．今後は Flash Crowd 以外の要因に対しても，効果的な複製サーバの配置先を決定する手法について研究していく必要がある．例えば，稼働率を保証する種類の SLA を実現する場合，サービス全体の稼働率を考慮して配置先を決定する必要がある．これにはまず，現在配置している複製サーバの稼働率からサービス全体の稼働率を計算する必要がある．具体的には，複製サーバのソフトウェア品質によって計算される稼働率や配置先となったホスティングマシンのハードウェア品質によって計算される稼働率を収集する仕組みが必要となる．また，配置先を決定した際に稼働率を再計算する仕組みも必要となる．スケーラビリティを考慮すると，EPSR のような特定のサーバを用意する必要のない分散型アーキテクチャで実現することが望ましい．例えば Gossip プロトコル [124] のような分散プロトコルを用いて実現する研究を行う必要があると考えられる．

謝辞

本論文は著者が慶應義塾大学大学院理工学研究科開放環境科学専攻の後期博士課程に在籍中の研究成果をまとめたものです．本研究を行うにあたって，また本論文をまとめるにあたって，多くの方々からご指導とご協力を賜りました．お世話になった全ての方々へこの場を借りて御礼申し上げます．

まず，本論文の主査であり，著者の指導教員である慶應義塾大学理工学部情報工学科 河野健二准教授に深く感謝いたします．河野健二准教授には著者が電気通信大学学部 4 年生時から，慶應義塾大学大学院修士課程，慶應義塾大学大学院後期博士課程，そして所定単位取得満期退学後本論文をまとめる 1 年間まで，7 年間もの長きにわたって，研究活動の意義から，論文のまとめ方，口頭発表におけるプレゼンテーションの要点，そして研究活動の面白さをご教示いただきました．その親身なご指導は大変ありがたいものでした．今現在研究活動に従事できているのは，ひとえに河野健二准教授と出会えたからであると思っております．さらに国内外有数の学会へ挑戦し，参加する機会を与えていただきました．また高度な研究活動を可能とする，国内有数の潤沢な研究設備を利用させていただきました．心より感謝いたします．

次に，本論文の副査を担当していただいた，慶應義塾大学理工学部情報工学科 寺岡文男教授，西宏章准教授，重野寛准教授に感謝いたします．副査の皆様には貴重なお時間を割いていただき，本論文を丁寧に査読していただきました．副査の皆様との有意義な議論によって，本論文の完成度が向上したと実感しております．深く感謝いたします．

電気通信大学情報理工学部情報・通信工学科 岩崎英哉教授には，著者が電気通信大学電気通信学部情報工学科ならびに電気通信大学大学院電気通信学研究科情報工学専攻に在籍していた際，大変お世話になりました．岩崎英哉教授には初めて研究活動というものに触れる大切な時期に，研究を進める上で重要な助言をいくつもいただきました．心より感謝いたします．

筑波大学システム情報工学研究科コンピュータサイエンス専攻 杉木章義助教に

感謝いたします．杉木章義助教には著者が電気通信大学在籍時から，著者の研究活動に対し多くの有意義な助言をいただきました．深く感謝いたします．

慶應義塾大学理工学部情報工学科 山田浩史特別研究助教には，著者が電気通信大学在籍時から，共同研究を通して活発な議論をさせていただきました．共に共著論文を執筆したことは著者にとって大変有意義な経験です．また，山田浩史特別研究助教の常に前向きな姿勢に多くの活力をいただきました．深く感謝いたします．

慶應義塾大学理工学部情報工学科 河野研究室ならびに電気通信大学情報理工学部情報・通信工学科 岩崎研究室の皆様に感謝いたします．特に共著論文を執筆した島田明男氏，小島俊範氏，早川愛氏，石川豊氏，堀江光氏には，共に刺激しあうことで，著者の成長を助けていただきました．優秀な後輩に囲まれて過ごすことができたのは大変幸運であったと感じております．深く感謝いたします．また，慶應義塾大学大学院へ共に入学した嶋村誠氏，花岡美幸氏，宮地大輝氏，山口聖司氏には，勝手分からぬ新しい学舎での講義受講や研究活動の際に，多くの助けをいただきました．感謝いたします．

本論文の研究を進めるにあたって使用した実験機材の一部，および本研究の原著論文の発表にあたっては，文部科学省科学研究費補助金「特定領域研究」や，科学技術振興機構 CREST による支援をいただきました．また，慶應義塾先端科学技術研究センターの KLL 後期博士課程研究助成金から本研究に対する支援をいただきました．さらに，日本学生支援機構奨学金や慶應義塾大学大学院奨学金は，著者の研究活動の大きな支えとなりました．ここに感謝いたします．

最後に，経済的支援を惜しまず，著者の博士課程進学を支持し，現在まで暖かく見守っていただきました両親と祖母，そして妹に心より感謝いたします．

論文目録

定期刊行誌掲載論文

- Masato Asahara, Kenji Kono, Toshinori Kojima and Ai Hayakawa, "P2P-Based Approach to Finding Replica Server Locations for Alleviating Flash Crowds", *IEICE Transactions on Information and Systems*, Vol.E93-D, No.11, pp.3027–3037, November 2010.
- Masato Asahara, Akio Shimada, Hiroshi Yamada and Kenji Kono, "Strategy for Selecting Replica Server Spots on the Basis of Demand Fluctuations", *IPSJ Transactions on Advanced Computing System*, Vol.1, No.1, pp.160–173, June 2008.（IPSJ Digital Courier 船井若手奨励賞）

国際会議論文

- *Masato Asahara, Akio Shimada, Hiroshi Yamada and Kenji Kono, "Finding Candidate Spots for Replica Servers based on Demand Fluctuation", In *Proceedings of the 13th IEEE International Conference on Parallel and Distributed Systems* (*ICPADS '07*), 10 pages, December 2007.

国内学会発表

- *浅原 理人，河野 健二, "オーバレイネットワークにおける論理リンクの通信遅延変動に関する一考察"，情報処理学会 OS 研究会報告 (2009-OS-112), No.16，9 pages，August 2009.
- *浅原 理人，島田 明男，山田 浩史，河野 健二，"需要変動に応じた動的再配置が可能なミラーサーバ基盤"，情報処理学会 OS 研究会報告 (2007-OS-105), pp.123–130，April 2007.

- *浅原 理人，島田 明男，山田 浩史，河野 健二，“需要変動に応じたコンテンツの再配置を行う Content Delivery Network”，情報処理学会 OS 研究会報告 (2006-OS-102)，pp.85–92，May 2006.

参考文献

[1] Jon Postel. Internet Protocol. RFC 791 (Standard), September 1981.

[2] National Science Foundation. A Brief History of NSF and the Internet. `http://www.nsf.gov/news/news_summ.jsp?cntn_id=103050`, 2003.

[3] Internet World Stats. World Internet Usage and Population Statistics. `http://www.internetworldstats.com/stats.htm`, 2010.

[4] Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2009–2014. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, 2010.

[5] Stephen Adler. The Slashdot Effect, An Analysis of Three Internet Publications. `http://linuxgazette.net/issue38/adler1.html`, February 1999.

[6] Scott Lorenz. Is your Web site ready for the flash crowd? `http://www.serverworldmagazine.com/sunserver/2000/11/flash.shtml`, 2000.

[7] Paul Bar-ford and David Plonka. Characteristics of Network Traffic Flow Anomalies. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement* (*IMW '01*), pp. 69–73, November 2001.

[8] Venkata N. Padmanabhan and Kunwadee Sripanidkulchai. The Case for Cooperative Networking. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), pp. 178–190, March 2002.

[9] John Dilley, Bruce M. Maggs, Jay Parikh, Harald Prokop, Ramesh K. Sitaraman, and William E. Weihl. Globally Distributed Content Delivery. *IEEE Internet Computing*, Vol. 6, No. 5, pp. 50–58, October 2002.

[10] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *Proceedings of the 11th ACM International World Wide Web Conference* (*WWW '02*), pp. 293–304, May 2002.

[11] Hani Jamjoom and Kang G. Shin. Persistent Dropping: An Efficient Control of Traffic Aggregates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '03*), pp. 287–298, August 2003.

[12] Michael J. Freedman. Experiences with CoralCDN: A Five-Year Operational View. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '10*), pp. 95–110, April 2010.

[13] Tyron Stading, Petros Maniatis, and Mary Baker. Peer-to-Peer Caching Schemes to Address Flash Crowds. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), pp. 203–213, March 2002.

[14] Geeknet, Inc. Slashdot FAQ. `http://slashdot.org/faq/`.

[15] Wikipedia, the free encyclopedia. Slashdot effect. `http://en.wikipedia.org/wiki/Slashdot_effect`.

[16] Geeknet, Inc. Slashdot. `http://slashdot.org/`.

[17] Facebook, Inc. Facebook. `http://www.facebook.com/`.

[18] Twitter, Inc. Twitter. `http://twitter.com/`.

[19] Jeremy Elson and Jon Howell. Handling Flash Crowds from your Garage. In *Proceedings of the USENIX Annual Technical Conference* (*USENIX '08*), pp. 171–184, June 2008.

[20] Cal Henderson. *Building Scalable Web Sites*. O'Reilly, 2006.

[21] Geoff C. Berry, Jeffrey S. Chase, Geoff A. Cohen, Landon P. Cox, and Amin Vahdat. Toward Automatic State Management for Dynamic Web Services. In *Proceedings of the 1999 Netstore Symposium*, 12 pages, October 1999.

[22] Eric Jul, Henry Levy, Norman Hutchinson, and Andrew Black. Fine-grained mobility in the Emerald system. *ACM Transactions on Computer Systems*, Vol. 6, No. 1, pp. 109–133, February 1988.

[23] Jim Challenger, Arun Iyengar, Karen Witting, Cameron Ferstat, and Paul Reed. A Publishing System for Efficiently Creating Dynamic Web Content. In *Proceedings of the 19th IEEE INFOCOM* (*INFOCOM 2000*), Vol. 2, pp. 844–853, March 2000.

[24] Haifeng Yu and Amin Vahdat. Design and Evaluation of a Conit-Based Continuous Consistency Model for Replicated Services. *ACM Transactions on Computer Systems*, Vol. 20, No. 3, pp. 239–282, August 2002.

[25] Steven D. Gribble, Eric A. Brewer, Joseph M. Hellerstein, and David Culler. Scalable, Distributed Data Structures for Internet Service Construction. In *Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI 2000*), pp. 319–332, October 2000.

[26] Germán Goldszmidt and Guerney Hunt. Scaling Internet Services by Dynamic Allocation of Connections. In *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management* (*IM '99*), pp. 171–184, May 1999.

[27] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, and Philip S. Yu. The State of the Art in Locally Distributed Web-Server Systems. *ACM Computing Surveys*, Vol. 34, No. 2, pp. 263–311, June 2002.

[28] Amazon.com. Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.

[29] Simson Garfinkel and Harold Abelson. *Architects of the Information Society: 35 Years of the Laboratory for Computer Science at MIT*. MIT Press, 1999.

[30] Rovert P. Goldberg. Survey of virtual machine research. *IEEE Computer Magazine*, Vol. 7, No. 6, pp. 34–45, 1974.

[31] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Guavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles* (*SOSP '07*), pp. 205–220, October 2007.

[32] Roxana Geambasu, Amit Levy, Tadayoshi Kohno, Arvind Krishnamurthy, and Henry M. Levy. Comet: An active distributed key/value store. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '10*), 14 pages, October 2010.

[33] Amazon S3. `http://aws.amazon.com/s3/`.

[34] Apache Cassandra. `http://cassandra.apache.org/`.

[35] Project Voldemort. `http://project-voldemort.com/`.

[36] Michael Rabinovich and Amit Aggarwal. RaDaR: A Scalable Architecture for a Global Web Hosting Service. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 31, No. 11–16, pp. 1545–1561, May 1999.

[37] Pablo Rodriguez and Sandeep Sibal. SPREAD: Scalable Platform for Reliable and Efficient Automated Distribution. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 33, No. 1–6, pp. 33–49, June 2000.

[38] Xueyan Tang and Samuel T. Chanson. Coordinated En-Route Web Caching. *IEEE Transactions on Computers*, Vol. 51, No. 6, pp. 595–607, June 2002.

[39] Keqiu Li, Hong Shen, Francis Y.L. Chin, and Si Qing Zheng. Optimal Methods for Coordinated Enroute Web Caching for Tree Networks. *ACM Transactions on Internet Technology*, Vol. 5, No. 3, pp. 480–507, August 2005.

[40] Akamai. Akamai CDN. `http://www.akamai.com/`.

[41] Jan Coppens, Tim Wauters, Filip De Turck, Bart Dhoedt, and Piet Demeester. Design and Performance of a Self-Organizing Adaptive Content Distribution Network. In *Proceedings of the 2006 IEEE/IFIP Network Operations and Management Symposium* (*NOMS '06*), pp. 534–545, April 2006.

[42] Jaison Paul Mulerikkal and Ibrahim Khalil. An Architecture for Distributed Content Delivery Network. In *Proceedings of the 15th IEEE International Conference on Networks* (*ICON '07*), pp. 359–364, November 2007.

[43] Samee Ullah Khan and Ishfaq Ahmad. Comparison and Analysis of Ten Static Heuristics-based Internet Data Replication Techniques. *Journal of Parallel and Distributed Computing*, Vol. 68, No. 2, pp. 113–136, February 2008.

[44] Savio S. H. Tse. Approximate Algorithms for Document Placement in Distributed Web Servers. In *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms, and Networks* (*ISPAN '04*), pp. 61–66, May 2004.

[45] Xueyan Tang and Jianliang Xu. On Replica Placement for QoS-aware Content Distribution. In *Proceedings of the 23rd IEEE INFOCOM* (*INFOCOM '04*), Vol. 2, pp. 806–815, March 2004.

[46] Magnus Karlsson and Christos Karamanolis. Choosing Replica Placement Heuristics for Wide-Area Systems. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems* (*ICDCS '04*), pp. 350–359, March 2004.

[47] Xueyan Tang and Jianliang Xu. QoS-Aware Replica Placement for Content Distribution. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 10, pp. 921–932, 2005.

[48] Savio S. H. Tse. Approximate Algorithms for Document Placement in Distributed Web Servers. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 6, pp. 489–496, June 2005. (Extended and improved version of [44]).

[49] Guillaume Pierre, Ihor Kuz, Maarten van Steen, and Andrew S. Tanenbaum. Differentiated Strategies for Replicating Web Documents. *Computer Communications*, Vol. 24, No. 2, pp. 232–240, February 2001.

[50] Guillaume Pierre and Maarten van Steen. Globule: A Platform for Self-Replicating Web Documents. In *Proceedings of the 6th International Conference on Protocols for Multimedia Systems* (*PROMS '01*), Vol. 2213 of *Lecture Notes in Computer Science*, pp. 1–11. Springer, October 2001.

[51] Guillaume Pierre and Maarten van Steen. Globule: a Collaborative Content Delivery Network. *IEEE Communications Magazine*, Vol. 44, No. 8, pp. 127–133, August 2006.

[52] Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Latency-Driven Replica Placement. In *Proceedings of the 2005 IEEE/IPSJ International Symposium on Applications and the Internet* (*SAINT '05*), pp. 399–405, February 2005.

[53] Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Latency-Driven Replica Placement. *IPSJ Journal*, Vol. 47, No. 8, pp. 2613–2624, August 2006.

[54] Chenyu Pan, Merdan Atajanov, Mohammad Belayet Hossain, Toshihiko Shimokawa, and Norihiko Yoshida. FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies. *IEICE Transactions on Communications*, Vol. E89-B, No. 4, pp. 1119–1126, April 2006.

[55] Vivek S. Pai, Limin Wang, KyoungSoo Park, Ruoming Pang, and Larry Peterson. The Dark Side of the Web: An Open Proxy's View. In *Proceedings of the 2nd Workshop on Hot Topics in Networks* (*HOTNETS-II*), 6 pages, November 2003.

[56] Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek Pai, and Larry Peterson. Reliability and Security in the CoDeeN Content Distribution Network. In *Proceedings of the USENIX Annual Technical Conference* (*USENIX '04*), pp. 171–184, June 2004.

[57] Weibin Zhao and Henning Schulzrinne. DotSlash: A Self-configuring and Scalable Rescue System for Handling Web Hotspots Effectively. In *Proceedings of the 9th International Workshop on Web Content Caching and Distribution* (*WCW*

'04), Vol. 3293 of *Lecture Notes in Computer Science*, pp. 1–18. Springer, October 2004.

[58] B. Molina Moreno, C.E. Palau Salvador, M. Esteve Domingo, I. Alonso Peña, and V. Ruiz Extremera. On Content Delivery Network Implementation. *Computer Communications*, Vol. 29, pp. 2396–2412, August 2006.

[59] Antonella Di Stefano and Corrado Santoro. An Economic Model for Resource Management in a Grid-based Content Distribution Network. *Future Generation Computer Systems*, Vol. 24, No. 3, pp. 202–212, 2008.

[60] Bram Cohen. Incentives Build Robustness in BitTorrent., 2003. `http://bitconjurer.org/BitTorrent/bittorrentecon.pdf`.

[61] KyoungSoo Park and Vivek S. Pai. Scale and Performance in the CoBlitz Large-File Distribution Service. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '06*), pp. 29–44, May 2006.

[62] Mayur Deshpande, Abhishek Amit, Mason Chang, Nalini Venkatasubramanian, and Sharad Mehrotra. Flashback: A Peer-to-Peer Web Server for Flash Crowds. In *Proceedings of the 27th IEEE International Conference on Distributed Computing Systems* (*ICDCS '07*), 15 pages, June 2007.

[63] Michael J. Freedman, Eric Freudenthal, and David Mazières. Democratizing Content Publication with Coral. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '04*), pp. 239–252, March 2004.

[64] Antony Rowstron and Peter Druschel. Storage Managent and Caching in PAST, A Large-scale, Persistent Peer-to-Peer Storage Utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles* (*SOSP '01*), pp. 188–201, October 2001.

[65] Yan Chen, Randy H. Katz, and John D. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), Vol. 2429 of *Lecture Notes in Computer Science*, pp. 306–318. Springer, March 2002.

[66] Yan Chen, Randy H. Katz, and John D. Kubiatowicz. SCAN: A Dynamic, Scalable, and Efficient Content Distribution Network. In *Proceedings of the International Conference on Pervasive Computing* (*Pervasive '02*), Vol. 2414 of *Lecture Notes in Computer Science*, pp. 282–296. Springer, August 2002.

[67] Yee Jiun Song, Venugopalan Ramasubramanian, and Emin Gün Sirer. Optimal Resource Utilization in Content Distribution Networks. Technical Report TR2005-2004, Cornell University, November 2005.

[68] Marcel Waldvogel, Paul Hurley, and Daniel Bauer. Dynamic Replica Management in Distributed Hash Tables. Research Report RZ–3502, IBM, 2003.

[69] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of the 21st IEEE INFOCOM* (*INFOCOM '02*), Vol. 1, pp. 170–179, June 2002.

[70] T. S. Eugene Ng and Hui Zhang. A Network Positioning System for the Internet. In *Proceedings of the USENIX Annual Technical Conference* (*USENIX '04*), pp. 141–154, June 2004.

[71] Marcelo Pias, Jon Crowcroft, Steve R. Wilbur, Timothy L. Harris, and Saleem N. Bhatti. Lighthouses for Scalable Distributed Location. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems* (*IPTPS '03*), Vol. 2735 of *Lecture Notes in Computer Science*, pp. 278–291. Springer, February 2003.

[72] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems* (*ICDCS '04*), pp. 178–187, March 2004.

[73] Michal Szymaniak, Guillaume Pierre, and Maarten van Steen. Scalable Cooperative Latency Estimation. In *Proceedings of the 10th IEEE International Conference on Parallel and Distributed Systems* (*ICPADS '04*), pp. 367–376, July 2004.

[74] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, Distributed Network Coordinates. In *Proceedings of the 2nd Workshop on Hot Topics in Networks* (*HOTNETS-II*), 6 pages, November 2003.

[75] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '04*), pp. 15–26, August 2004.

[76] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Stable and Accurate Network Coordinates. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems* (*ICDCS '06*), 10 pages, July 2006.

[77] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network Coordinates in the Wild. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '07*), pp. 299–311, April 2007.

[78] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '01*), pp. 149–160, 2001.

[79] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '01*), pp. 161–172, 2001.

[80] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms* (*Middleware '01*), Vol. 2218 of *Lecture Notes in Computer Science*, pp. 329–350. Springer, November 2001.

[81] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: a resilient global-scale overlay for service de-

ployment. *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, pp. 41–53, 2004.

[82] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*), Vol. 2429 of *Lecture Notes in Computer Science*, pp. 53–65. Springer, March 2002.

[83] Marcel Waldvogel and Roberto Rinaldi. Efficient Topology-Aware Overlay Network. *ACM SIGCOMM Computer Communications Review*, Vol. 33, No. 1, pp. 101–106, January 2003.

[84] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. OpenDHT: A Public DHT Service and Its Uses. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '05*), pp. 73–84, August 2005.

[85] 首藤一幸, 田中良夫, 関口智嗣. オーバレイ構築ツールキット Overlay Weaver. 情報処理学会論文誌: コンピューティングシステム, Vol. 47, No. SIG12 (ACS 15), pp. 358–367, September 2006.

[86] Kazuyuki Shudo, Yoshio Tanaka, and Satoshi Sekiguchi. Overlay Weaver: An Overlay Construction Toolkit. *Computer Communications* (*Special Issue on Foundations of Peer-to-Peer Computing*), Vol. 31, No. 2, pp. 402–412, February 2008. (available online on August 14, 2007).

[87] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to Model an Internetwork. In *Proceedings of the 15th IEEE INFOCOM* (*INFOCOM '96*), pp. 594–602, March 1996.

[88] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), March 1995.

[89] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[90] John T. Moy. The OSPF Specification. RFC 1131 (Proposed Standard), October 1989.

[91] John T. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998.

[92] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008.

[93] Pablo Rodriguez, Keith W. Ross, and Ernst W. Biersack. Distributing Frequently-Changin Documents in the Web: Multicasting or Hierarchical Caching? *Computer Networks and ISDN Systems. Selected Papers of the 3rd International Caching Workshop*, pp. 2223–2245, August 1998.

[94] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. Self-Organizing Wide-Area Network Caches. In *Proceedings of the 17th IEEE INFOCOM* (*INFOCOM '98*), Vol. 2, pp. 600–608, March 1998.

[95] P. Krishnan and Danny Raz. The Cache Location Problem. *IEEE Transactions on Networking*, Vol. 8, No. 5, pp. 568–582, October 2000.

[96] Athena Vakali and George Pallis. Content Delivery Networks: Status and Trends. *IEEE Internet Computing*, Vol. 7, No. 6, pp. 68–74, November/December 2003.

[97] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabián E. Bustamante. Drafting Behind Akamai (Travelocity-Based Detouring). In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '06*), pp. 435–446, September 2006.

[98] F. Thomson Leighton and Daniel M. Lewin. Global hosting system. US Patent 6,108,703, May 1999.

[99] Claudson F. Bornstein, Timothy K. Canfield, Gary L. Miller, Satish B. Rao, and Ravi Sundaram. Optimal route selection in a content delivery network. US Patent Application 60273241, March 2001.

[100] Ratul Mahajan. How Akamai Works. `http://www.cs.washington.edu/homes/ratul/akamai.html`.

[101] Samee Ullah Khan and Ishfaq Ahmad. Heuristics-based Replication Schemas for Fast Information Retrieval over the Internet. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing and Systems* (*PDCS '04*), pp. 278–283, November 2004.

[102] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the Placement of Web Server Replicas. In *Proceedings of the 20th IEEE INFOCOM* (*INFOCOM '01*), pp. 1587–1596, April 2001.

[103] Thanasis Loukopoulos and Ishfaq Ahmad. Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, Vol. 64, No. 11, pp. 1270–1285, November 2004.

[104] Israel Cidon, Shay Kutten, and Ran Soffer. Optimal allocation of electronic content. In *Proceedings of the 20th IEEE INFOCOM* (*INFOCOM '01*), Vol. 3, pp. 1773–1780, April 2001.

[105] Andy C. Bavier, Mic Bowman, Brent N. Chun, David E. Culler, Scott Karlin, Steve Muir, Larry L. Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating Systems Support for Planetary-Scale Network Services. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '04*), pp. 253–266, March 2004.

[106] Larry Peterson, Andy Bavier, Marc Fiuczynski, and Steve Muir. Experiences Building PlanetLab. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '06*), pp. 351–366, November 2006.

[107] Antonella Di Stefano and Corrado Santoro. A Marketplace-based Economical Model for Resource Provisioning Agreement in a P2P Grid. In *Proceedings of the HPDC Workshop on the Use of P2P, GRID and Agents for the Development of Content Distribution Networks* (*UPGRADE-CDN '06*), June 2006.

[108] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigraphy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (*STOC '98*), pp. 654–663, May 1997.

[109] David G. Thaler and Chinya V. Ravishankar. Using Name-Based Mapping to Increase Hit Rates. *IEEE Transactions on Networking*, Vol. 6, No. 1, pp. 1–14, February 1998.

[110] Michael J. Freedman and David Mazières. Sloppy Hashing and Self-Organizing Clusters. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems* (*IPTPS '03*), Vol. 2735 of *Lecture Notes in Computer Science*, pp. 45–55. Springer, February 2003.

[111] John Ashworth Nelder and Roger Mead. A Simplex Method for Function Minimization. *Computer Journal*, Vol. 7, pp. 308–313, 1965.

[112] Zhichen Xu, Chunqiang Tang, and Zheng Zhang. Building Topology-Aware Overlays Using Global Soft-Stat. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems* (*ICDCS '03*), pp. 500–508, May 2003.

[113] Cisco Systems, 2002. http://www.cisco.com/.

[114] F5 Networks, 2002. http://www.f5labs.com/.

[115] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement* (*IMW '02*), pp. 5–18, November 2002.

[116] Kenneth Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, Vol. 35, pp. 160–163, June 1997.

[117] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (*SIGCOMM '05*), pp. 85–96, August 2005.

[118] Ai Hayakawa, Masato Asahara, Kenji Kono, and Toshinori Kojima. Efficient Update Propagation by Speculating Replica Locations on Peer-to-Peer Networks.

In *Proceedings of the 15th IEEE International Conference on Parallel and Distributed Systems* (*ICPADS '09*), pp. 422–431, December 2009.

[119] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Chris Wells, and Ben Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS 2000*), pp. 190–201, 2000.

[120] Yasushi Saito, Christos Karamanolis, Magnus Karlsson, and Mallik Mahalingam. Taming aggressive replication in the Pangaea wide-area file system. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '02*), pp. 15–30, December 2002.

[121] Jeremy Stribling, Yair Sovran, Irene Zhang, Xavid Pretzer, Jinyang Li, M. Frans Kaashoek, and Robert Morris. Flexible, Wide-Area Storage for Distributed Systems with WheelFS. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '09*), pp. 43–58, April 2009.

[122] Tobias Groothuyse, Swaminathan Sivasubramanian, and Guillaume Pierre. GlobeTP: Template-Based Database Replication for Scalable Web Applications. In *Proceedings of the 16th ACM International World Wide Web Conference* (*WWW '07*), pp. 301–310, May 2007.

[123] Ajay Mahimkar, Jasraj Dange, Vitaly Shmatikov, Harrick Vin, and Yin Zhang. dFence: Transparent Network-based Denial of Service Mitigation. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI '07*), pp. 327–340, April 2007.

[124] Mark Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based Aggregation in Large Dynamic Networks. *ACM Transactions on Computer Systems*, Vol. 23, No. 3, pp. 219–252, August 2005.