多地点接続リアルタイム型アプリケーションに適用する分散処理型通信方式

川端　明生

電気通信大学大学院
情報理工学研究科 情報・通信工学専攻

博士 (工学) の学位申請論文
2016 年 7 月

博士論文審査委員会

主査　大木 英司 教授

委員　來住 直人 教授

委員　山尾 泰 教授

委員　伊藤 大雄 教授

委員　寺田 実 准教授

2016 年 7 月 28 日

概要

仮想化技術の進展によって，様々なアプリケーションがネットワーク内のクラウド上で動作可能となるが，広域なネットワークを介して多地点間通信を行うリアルタイム型アプリケーションでは，低遅延なエンド-エンド通信を実現する通信技術の確立が課題である．本研究は，ネットワーク上で動作するアプリケーションを通信サービスとして提供する場合のエンド-エンドの通信遅延時間を 低減することを目的とする分散処理型通信方式である．

提案方式は，ネットワーク内でユーザ端末と近いロケーションに配備された複数のサーバを用いてアプリケーションを分散処理をする．ユーザ端末は複数のサーバからエンド-エンドの遅延時間を最小化するサーバを選択し，分散処理する複数のサーバ間では処理結果の同報通信を行う．

ユーザ端末とネットワーク内に配備されたサーバとの通信は，ユーザ端末ごとにサーバとの通信遅延時間が異なるため，実際のイベント発生順序とネットワークを介したサーバへのイベント到着順序が異なる可能性があり，イベントの処理順序を補正する仕組みが必要となる．分散処理する各サーバでは，イベントの順序性を再現するために，現在時刻からイベントの順序性を再現可能な時刻まで時間を遅らせた仮想時刻をあらかじめ計算し，仮想時刻上でイベントの順序性を再現する．分散処理をするサーバでは，各ユーザ端末との通信遅延時間を事前に測定しておき，ユーザ端末毎の通信遅延時間に応じた待ち合わせを行うことで，仮想時刻上でイベント発生順序を再現する．

ネットワーク内の複数のサーバから，エンド-エンドの通信遅延時間を最小化するサーバを決定するためのサーバ選択問題として，現在時刻と仮想時刻の差であるユーザ端末補正時間を最小化する．サーバ選択問題についての計算複雑度の評価し，本問題は NP 困難であることを示す．サーバ選択問題を線形計画問題として定式化し，エンド-エンドの通信遅延時間を最小化する仮想時刻と各ユーザ端末が選択するサーバを線形計画問題を解くことで決定する．

提案方式の性能評価として，サーバ間ネットワークトポロジおよびネットワーク上のサーバ配備箇所によるエンド-エンドの通信遅延時間 を評価する．サーバ間ネットワークトポロジの評価では，同一のサーバ配備箇所で異なるリンクトポロジで改善効果を比較し，フルメッシュ型やリング型のようにサーバ間が，最短距離に近い距離のリンクを持っているトポロジのほうが，遅延特性の改善効果が高いことを示す．また，サーバ配備箇所としては，よりユーザに近いロケーションにサーバを配備すると遅延特性の改善効果が高いことを示す．サーバ選択問題の評価としては，特定エリア内に一様分布した 200 台の

ユーザ端末について，本研究で定式化した最適化問題を解くことで，遅延時間を最小にするサーバが選択されることを示す．

実際のネットワークトポロジに近い条件における特性改善効果の確認として，日本のバックボーンネットワークの典型的なモデルを用いて，全国に分散した複数のサーバで分散処理する場合と 1 台のサーバで集中処理する場合を比較し，東京のサーバで集中処理する場合との比較では約 25 %の改善効果，集中処理型で最も遅延特性のよい和歌山のサーバで集中処理する場合との比較では約 2 %の改善効果があること示す．

提案方式の第一の拡張として，通信遅延時間に許容最大値のあるアプリケーションへの適用を考慮する．本ケースへの適用として，遅延許容時間を導入する．定式化した最適化問題を拡張し，第一目的関数として遅延許容時間を超えてアプリケーションが利用できないユーザ端末数，第二目的関数をユーザ端末補正時間として，これらを最小化する遅延許容時間を考慮したサーバ選択問題として定式化する．

提案方式の第一の拡張に関する性能評価として，遅延許容時間を変化させた場合の遅延許容時間を越えたユーザ端末数とユーザ端末補正時間について，集中処理型と分散処理型の比較を行う．これらの評価から，提案方式は，第一の拡張によって，遅延許容時間を超えて利用できないユーザ端末数が集中処理型よりも削減され，ユーザ端末補正時間も短いことから，より多くのユーザが利用可能で，かつ，遅延特性に優れた通信方式であることを示す．

提案方式の第二の拡張として，ネットワーク輻輳時の遅延変動を考慮する．本ケースへの適用として，第一の拡張を行った最適化問題のユーザ端末とサーバ間の通信遅延時間に遅延変動率を導入し，遅延変動時の最大遅延時間をユーザ端末とサーバ間の遅延時間として扱う．また，前述の最大遅延時間について，全ユーザ総和を最適化問題の第三の目的関数として導入し，遅延変動を最小化するサーバ選択問題として定式化する．定式化した最適化問題は，エンド-エンドの通信遅延時間を最小化した上で，各ユーザ端末が複数のサーバから遅延時間の最も少ないサーバを選択する．遅延時間と伝送距離が比例する条件においては，定式化した最適化問題を解くことで，ユーザ端末が複数のサーバと接続可能なネットワークにおいて，より伝送距離の短いサーバ間を選択するネットワーク設計法しても利用可能である．

提案方式の第二の拡張に関する性能評価として，前述の日本のバックボーンネットワークの典型的なモデルの関東エリアノードをサーバが配備されている拠点として，関東エリア内に 200 台のユーザ端末が一様分布した条件で評価する．ネットワーク輻輳時の遅延時間の評価として，ユーザ端末が選択するサーバを，提案方式に第二の拡張を行った設計法と，拡張を行わない設計法で比較評価を行う．提案方式に第二の拡張を行った設計法は，ネットワークが輻輳してユーザ端末と特定サーバとの遅延時間が増加した場合に，

許容遅延時間を越えるユーザ端末数が少なく，ネットワーク輻輳を考慮したユーザ端末が選択するサーバの決定が可能であることを示す．

提案方式の第三の拡張として，時間経過とともにユーザ端末が適宜追加されるアプリケーションへの適用を考慮した逐次参加型のユーザ参加方法を導入する．逐次参加型のユーザ参加方法では，最適化問題の決定変数として扱っていたリンクの利用有無とサーバの利用有無を表すパラメータを，利用中ユーザ端末については，決定した値として扱うことで，選択するサーバを変更しない制約条件を加味したサーバ選択問題として拡張する．また，新規ユーザ参加時の待ち時間を短縮するユーザ端末参加方法として，計算対象を新規ユーザ端末に限定することで，計算量を削減する．

第三の拡張に関する性能評価として，遅延特性と計算量について評価を行う．逐次参加型では，ユーザ端末が同一の配備箇所でも参加する順序によりユーザ端末補正時間が変わるものの，逐次参加型のいずれのパターンにおいても，集中処理型より低い値となっている．これらの結果から，逐次参加型でユーザ端末が参加する利用形態のアプリケーションにおいても，分散処理型通信方式の有効性を確認する．ユーザ端末がアプリケーションを利用開始する際の待ち時間の評価として，ユーザ参加時間の短縮化の拡張を行った参加方法について，計算時間の短縮効果について評価を行う．

逐次参加型は，利用中ユーザ端末が選択するサーバを既に利用中のサーバを選択する制約条件としているため，サーバを選択するための計算量が削減され，一斉参加型と比較し 1/100 以下に処理時間が短くなっており，ユーザ端末の待ち時間が短縮化されたユーザ参加方法であることを示す．また，ユーザ参加時間を短縮化したユーザ参加方法の導入により，さらに 70 %以上計算時間が削減されることを示す．

前述の評価結果から，提案する分散処理型通信方式は，許容遅延時間のあるアプリケーションでは利用ユーザ端末数を最大化することが可能で，低遅延なエンド-エンド通信を幅広いユーザに提供可能な通信方式である．また，ネットワーク輻輳や逐次参加型のユーザ参加方法についても提案方式の拡張を行い，様々なアプリケーションや通信環境への適用が可能となる．仮想化技術の進展とともに，ネットワーク内に様々なアプリケーションを配備する環境において，本研究により遅延特性に優れた通信環境を実現することが可能となり，より簡易にアプリケーションを利用するネットワークサービスの実現が期待される．

Distributed Processing Communication Scheme for Real-time Applications at Multiple Locations

Abstract

Various real-time applications, such as network games and video conferences, can be adaptable to work on the cloud servers in the network due to development of visualization technology. Delay-sensitive communication is required for using the real-time applications at cloud servers. This study addresses a distributed processing scheme for providing an application as communication services with end-to-end low delay time. In the proposed scheme, the application is processed with the help of distributed servers close to user location.

This thesis proposes a scheme that achieves the delay-sensitive communication for the real time applications at multiple locations. In the proposed scheme, the user communicates with a server, which is selected from distributed servers. Each distributed server multi-cast communicates with all other servers to synchronize the data of all users.

In an interactive communication type application, the application must work under the condition that the order at which events occur at user's terminals is maintained. In the proposed scheme, an event occurrence order is reproduced in the virtual time calculated at each server. The virtual time is defined as the time that the event occurrence order at servers is corrected to keep the same order at user terminals. To reproduce the order of events at the distributed servers, each event is queued in accordance with the delay time of each user that was previously measured.

To minimize the end-to-end delay time for applications, we introduce a server selection problem that minimizes the user correction time, which is the difference between the virtual time and actual time. We prove that the server selection problem is nondeterministic polynomial time (NP)-hard. We formulate the problem as an integer linear programming (ILP) problem.

To evaluate the basic characteristics of the proposed scheme, we evaluate the end-to-end delay time, which is dependent on the network topology of distributed servers

and server location at the network. Full-mesh or ring based network topology with shorter distances between servers is effective to improve the end-to-end delay time compared to other topologies with longer distance between servers. The location of near users is effective to improve end-to-end delay time by the evaluation of server locations in the network. The server selection of 200 users, which are uniformly distributed in a specific area, is determined to minimize the end-to-end delay time by solving the formulated ILP.

We evaluate the performance of the proposed scheme by using a typical Japanese backbone network model. In the central processing scheme, the server is located at a specified central node. In the distributed processing scheme, the servers are located at nationwide nodes. In this evaluation, the distributed processing scheme has an improvement of 25 percent in terms of delay time compared to the central processing scheme, and the distributed processing scheme has an improvement of 2 percent in terms of delay time compared to the central processing scheme that sentral server located at Wakayama node.

As the first extension of the proposed scheme, we introduce admissible delay time to apply the real-time applications that require low delay time in use. We introduce the number of users beyond the admissible delay time as out-of-service. We also formulate an optimization problem that minimizes the number of out-of-service users as the first objective function, and the correction time as the second objective function. To observe the effect of the first extension, we investigate the dependence of the number of out-of-service users and the correction time for the value of admissible delay time. We observe that the first extension of the proposed scheme reduces the number of out-of-service users and the correction time, compared to those of the central processing scheme.

As the second extension of the proposed scheme, we introduce the delay variation parameter to apply a situation of network congestion. In this scenario, the formulated optimization problem treats the delay time of links between users and servers as the maximum value of the delay variation rate. The sum of the delay time between the user and the server is considered as the third objective function. The formulated optimization problem with the second extension is effective for selecting the minimum distance links of users and servers, at the situation of distance proportional to the delay time.

To evaluate the effect of the second extension of the proposed scheme, we solve the

formulated ILP at the situation of 200 users uniformly distributed in Kanto region of the typical Japan backbone network model. We investigate network congestion with the topologies determined by the second extension scenario and first extension scenario at the location of servers. We observe that the second extension of the proposed scheme reduces the number of out-of-service users and the correction time, when the delay time of the link between the user and server increases.

As the third extension of the proposed scheme, we introduce the successive participation method for the case that users participate successively. In the successive participation method, we formulate the optimization problem in which the parameters for in-use users are treated as fixed parameters. For waiting time for participation without depending on the number of in-use users formulated optimization problem with the third extension is limited to participating users.

To evaluate the effect of the third extension of the proposed scheme, we investigate the performance of delay time by comparing the successive participation method with the simultaneous participation method. The delay time depends on the way uses participate; the user terminal correction time of the successive participation method is lower than the simultaneous participation method, which consider all the possible participation patterns. In addition, we investigate the performance of delay time for our introduced improved successive participation method. The processing time of the successive participation method is less than 1/100 of that of the simultaneous participation method, by limiting the calculation range of the participating users. For the improved successive participation method, the processing time is reduced more than 70 percent, compared to the original successive participation method.

From these results, the proposed scheme maximizes the number of in-use users, and provides delay-sensitive communications. We introduce several extensions of the proposed scheme to apply network congestion or user participation variations. Thus, this study can be applied to a variety of real-time applications.

謝辞

本論文は，電気通信大学　情報理工学研究科　情報・通信工学専攻 大木英司教授の御指導のもとに，著者が電気通信大学情報理工学研究科博士後期課程在学中に行った研究成果をまとめたものである．本研究を遂行するにあたり深甚なる御指導・御鞭撻を賜りました主任指導教員の大木英司教授に衷心より謝恩の意を表します．

学位審査の予備審査において，貴重なコメントをいただいた審査委員の先生方に心より感謝致します．特に，全体的な論文の論理展開・バランスや評価の妥当性について助言いただいた指導教員である來住直人教授，従来技術からの新規性とシステムトータル動作の明確化の観点から助言を頂いた山尾泰教授，関連研究と理論的・数学的な観点での結果分析について助言を頂いた伊藤大雄教授，本研究のトポロジ毎の優位性と新規性の明確化について助言を頂いた寺田実准教授に感謝致します．

本研究を進めるに際し，共に研究を進め，熱心な討論と有益な助言を頂きました Bijoy Chand Chatterjee 博士研究員，Praphan Pavarangkoon 博士研究員，Seydou Ba 氏をはじめとする大木研究室の皆様に感謝致します．

目次

図目次

表目次

第 1 章

はじめに

1.1 研究の背景

1.1.1 仮想化技術の進展と課題

Network Function Virtualization (NFV) [1–7] や Software Defined Network (SDN) [8] などの仮想化技術に関する標準化や製品化の取り組みによって，ネットワークの様々な機能が専用ハードウェアに依存せずに，汎用的なサーバ上で提供できるようになりつつある．仮想化技術を活用したクラウド環境では，従来はユーザの宅内環境にあった様々なアプリケーションをクラウド上のサービスとして実現することが可能となる．一方，このような仮想化された環境では，リアルタイム性に厳しいアプリケーションでは，ネットワークを経由した通信を行う際のエンド-エンドの通信遅延時間　(以下，遅延時間) に伴う課題が発生する．特に，ユーザ端末間で双方向通信のあるアプリケーションでは，ネットワークの遅延時間の増加に伴いユーザが体感上で違和感を覚えたり，1 つのシナリオを複数ユーザで共有するアプリケーションでは，ユーザ端末の遅延時間があまりにも大きいためにアプリケーションが利用できないケースも想定される．

また，図 1.1 に示すように，ユーザ端末とサーバ間の遅延時間がユーザ端末ごとに異なるため，実際のイベント発生順序とサーバにイベントが到着する順序が異なる．例えば，電話会議システム，対戦型ネットゲームや複数ロケーションの合奏 (以下，ネットセッション) 等の複数のユーザ端末間で双方向通信のあるアプリケーションでは，これらのイベントの順序逆転がアプリケーションが動作する上で問題となる．ネットワーク内に配備されたサーバでアプリケーションを動作させる場合は，イベント処理の順序性を保証する仕組みが必要となる．

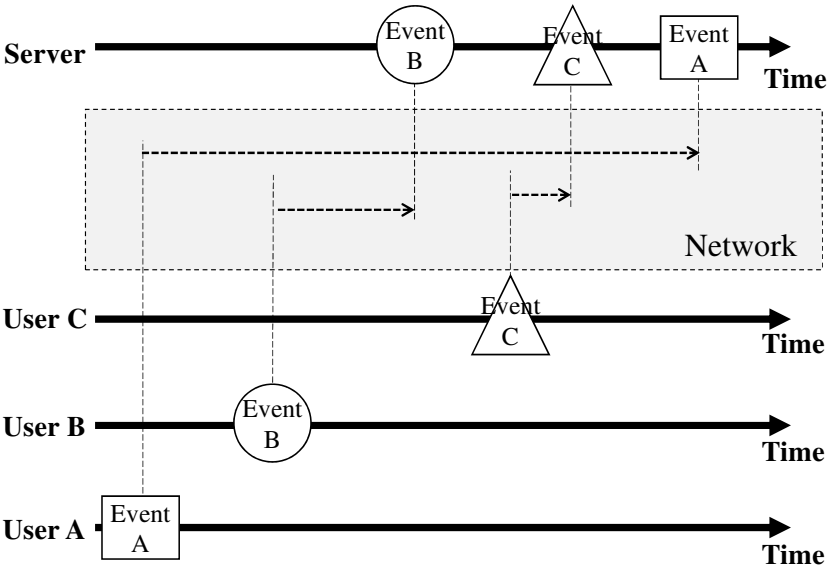


図 1.1　ネットワーク遅延による順序逆転

1.2　関連研究

1.2.1　分散処理に関する関連研究

イベントの発生順序を保証した分散処理に関する研究

本研究で対象とする多地点間で通信するアプリケーションを複数のサーバで分散処理をする研究としては，マルチプロセッサシステムや複数のプロセッサで一連のアプリケーションを分散処理する離散事象シミュレーションの研究分野で様々な方式が提案されている [9,10]．離散事象シミュレーションでは，分散処理をする論理プロセス間で処理の並列性を保ちながらイベントの発生順序を保証して処理を進めることが研究課題となっている [11]．これらの研究においては，イベント順序性を保証するアルゴリズムとして，保守的同期アルゴリズム [12,13] と楽観的同期アルゴリズム [14,15] に大別される [9,16]．

保守的同期アルゴリズムでは，イベントに時刻情報を付加して送信することで，イベント実行の際に時刻情報に基づいた順序で処理を行うことでイベントの順序性を保証する方式が取られ，初期の方式ではサーバ間の時刻の同期に合わせてサーバ間でイベントがなくても空のメッセージを送るヌルメッセージ法 [13,17] が検討された．ヌルメッセージ法ではイベントが発生していない場合でも，サーバ間のメッセージ通信が発生するため．通信

量を削減するための様々な改善方式 [18–25] が検討され，特性評価 [26] も行われている．代表的には受信サーバから必要な時のみ問い合わせメッセージを送信する方式 [27] や各サーバで時刻同期のためのサーバ間通信がなく処理が進められるようにサーバ間で共通の時刻情報をそれぞれのサーバで持つことでメッセージ数を削減する方式 [28] が検討されている．

データベースの同時実行のために考案 [29] された楽観的同期アルゴリズムは，イベントの順序補正処理を行わないが，過去のイベントを検出した場合は，処理を過去に遡り補正するロールバック処理によってイベントの順序性を保証を行う．ロールバック処理の実現方式として，Time Warp [15] が知られているが，ロールバックのために処理履歴を記憶し続けるためメモリの消費量が課題であり，メモリ量削減のための最適な履歴記憶間隔に関する研究 [30] や，その評価 [31, 32] など様々な効率的な履歴記憶法が検討されている．また，Time Warp において，ロールバックが不要となり履歴を消去可能な時刻である未完了イベントの時刻情報の最小値である GVT (Global Virtual Time) [33] の分散環境での正確な決定法 [34] なども検討されている．

イベントの順序補正に関する研究

分散処理を行う複数のサーバがネットワークを経由して接続されるサーバ間の遅延時間が異なる場合は，イベント発生時と同一の順序性を保証して処理をする必要があり，様々な研究が行われている [9, 35–37]．イベントの順序性保証を行う方式としては，Causal ordering と Total ordering が代表的な方式であり，図 1.2 に Causal Order と Total Order のイベント順序の補正例を示す．Causal ordering は，送信側でタイムスタンプを付与してイベントを送信し，タイムスタンプの値に基づいてイベント処理を行うことでイベントの順序性を保証する方式で，様々な実装例 [40–46] が報告されている．また，Total ordering は，全サーバにおいて，全てのイベントを同一順序で処理をする方式で，様々な研究例・実装例 [47–50] が報告されている．

時刻情報の取得に関する研究

タイムスタンプで用いる時刻情報については，時刻同期プロトコルである Network Time Protocol (NTP) [51] やより高精度な Precision Time Protocol (PTP) [52] の標準化に伴い多くの実装が行われ，UTC (協定世界時) の高精度な時刻情報の取得が可能となっている．また，PTP を用いることで μ 秒オーダでの時刻精度がの実現も可能となっている．

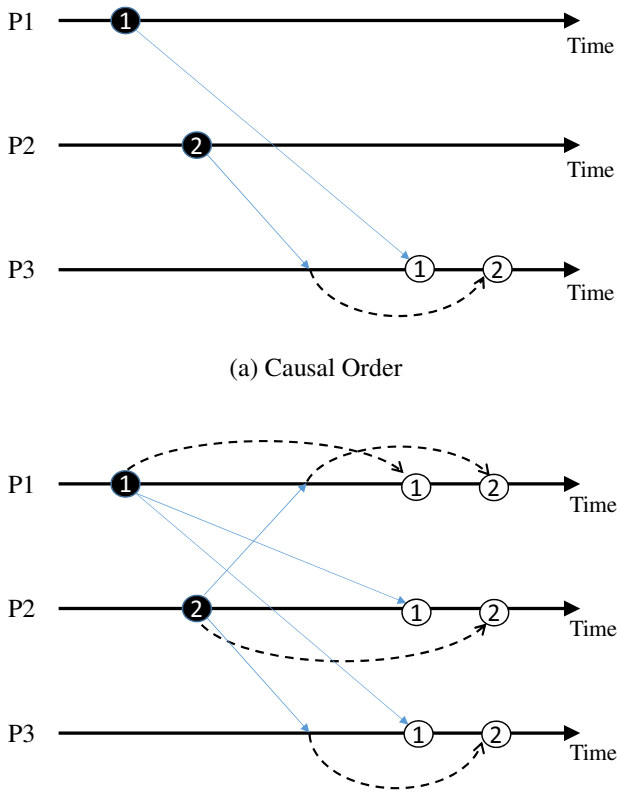


図 1.2　Causal Order と Total Order による順序補正例

1.2.2　分散処理の産業分野での適用技術

産業分野においては，ネットワークゲームの分野やネットワークセッションなどの音楽分野で，様々な技術が開発されている．例えば，格闘型ゲームでは，プレイヤー間で同一の状態を保持する完全同期型の通信方式を用いていることが報告 [53] されている．完全同期型の通信方式では，あらかじめ測定したプレイヤー間の遅延時間に基づいて，相手プレイヤーからのイベントが到着するまでの遅延時間分だけ自分のイベントを待ち合わせて処理を行うことで，イベントの順序性を保証しており，保守的同期アルゴリズムに分類される．

ロールプレイングゲームでは，プレイヤー間で同一の状態を保持しない非同期型の通信方式をとっていることが報告 [53] されている．非同期型の通信方式では，全体の状態管理を役割のプレイヤーが存在し，過去のイベントはロールバックして処理を行うことで，

イベントの順序性を保証しており，楽観的同期アルゴリズムに分類される．

音楽分野では，ネットセッションのアプリケーションである NETDUETTO [54] では，相手演奏者との遅延時間を測定し遅延時間に応じて，自分のモニタ音に遅延を入れることで演奏のタイミングを合わせる工夫がされており，保守的同期アルゴリズムに分類される．このように，学術的な研究のみでなく，産業分野でも多くの取り組みが行われ，アプリケーションに応じて様々な工夫がなされている．

1.2.3　低遅延な通信遅延を実現するサーバ配備に関する関連研究

ネットワークを介して利用するアプリケーションは，ネットワーク上の集約拠点にあるサーバ上で動作し，ユーザ端末がサーバと通信をしながらアプリケーションを利用する形態，もしくは，アプリケーションがユーザ端末上で動作し，ユーザ端末同士がネットワークを介して通信しながらアプリケーションを利用する形態がある．前述の集約拠点に配備されたサーバ上で処理されていた Web サービス等のアプリケーションを，よりユーザ端末と遅延時間の少ないロケーションに配備したサーバで処理することで遅延特性を改善し，リアルタイム性の厳しいアプリケーションにも対応しようというエッジコンピューティング [55–59] の検討がされている．エッジコンピューティングの実装例として，分散型の Web アプリ実行エンジンを活用し，Web コンテンツの取得解析とスクリプト処理をユーザに近いサーバで実行し，端末ではコンテンツの描画結果の再構成処理のみを行うことで．Android 上の標準 Web エンジンよりも 20〜70 %早く表示が完了するとの研究 [60] がされている．

一方，これらの技術は，複数拠点間で双方向通信のあるアプリケーションでは，ユーザ端末毎に異なるサーバを選択されている場合は，ネットワークを経由したサーバ間の通信が必要となる．このように，ネットワークを経由したサーバ間の通信が必要な場合は，エッジコンピューティングの特徴であるユーザに近いサーバで一部の処理を実行したとしても，エンド-エンドの遅延時間は短縮化されないという課題がある．

1.3　リアルタイム型アプリケーションにおける遅延時間の要求条件

図 1.3 に示すように，電話サービス，ネットゲーム，ネットセッションなどのリアルタイム性に厳しいアプリケーションでは，ネットワークの遅延時間を数十 ms 以内にすることが推奨され，以下のような具体的な評価事例も報告されている．

加入電話と同等な品質が求められる 0AB〜J 番号を用いる IP 電話サービスでは，エン

ド-エンドの遅延時間が　70ms 以下とされている [61]．長野五輪の閉会式の際に実施した 5 大陸の同時合唱の実験結果では，往復伝送の遅延時間が 60ms 以下であることが望ましいとされている [62]．ネットワークゲームにおいては，ゲームの種類によって遅延許容時間が異なるが，50ms 以内であればユーザに気づかれることなくゲームが可能で [64]，100-500ms を超えるとゲームの種類によっては，ゲームが成り立たなくなるとの調査結果が報告されている [14, 63–69]．一方で，品質保証型の国内通信サービスの Service Level Agreement (SLA) では，ネットワークの遅延時間が 35ms 程度 [70, 71] であり，エンド-エンドの遅延時間を抑える通信方式が，リアルタイム性や遅延特性の厳しいアプリケーションでは必要となる．

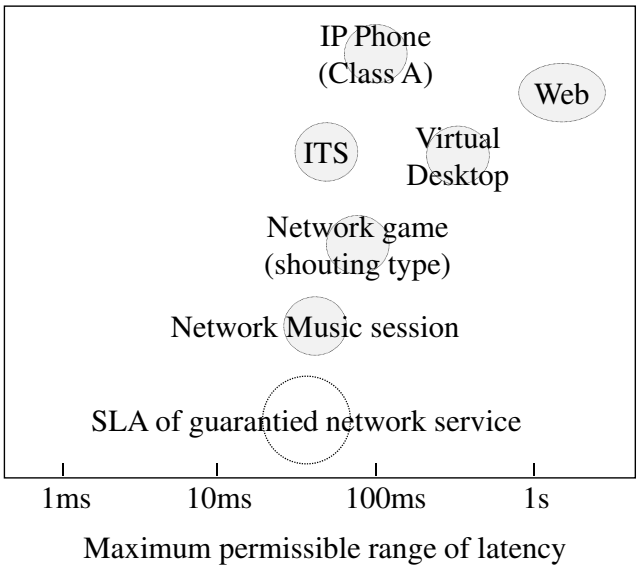


図 1.3　アプリケーションごとの許容遅延

1.4　本研究の狙いと解決する課題

1.4.1　本研究の狙い

本研究は，広域なネットワークを介した多地点間でリアルタイム性の高いアプリケーションを利用する場合に，遅延時間を最小限にするための分散処理型通信方式を提案する．該当するアプリケーションとしては，複数拠点間の電話会議，ネットセッション，ネットワークゲーム，オンライントレーディング，交通管制システム等が挙げられる．

図 1.4 に，処理方式に関する従来技術と比較した本研究のターゲット領域を示す．保守的同期アルゴリズムでは，前述のようにアプリケーションがユーザ端末上で動作し，通信

相手との遅延時間に応じて先に到着した処理を待ち合わせることでイベント発生順序を再現する方式 [53, 72–74] が取られる．上記の保守的同期アルゴリズムの処理方式は，あらかじめ定めておいた仮想時間をユーザ端末ごとに計算し，イベントの発生順序が再現された仮想時刻でアプリケーションがイベント処理をする分散処理方式と考えることができる．

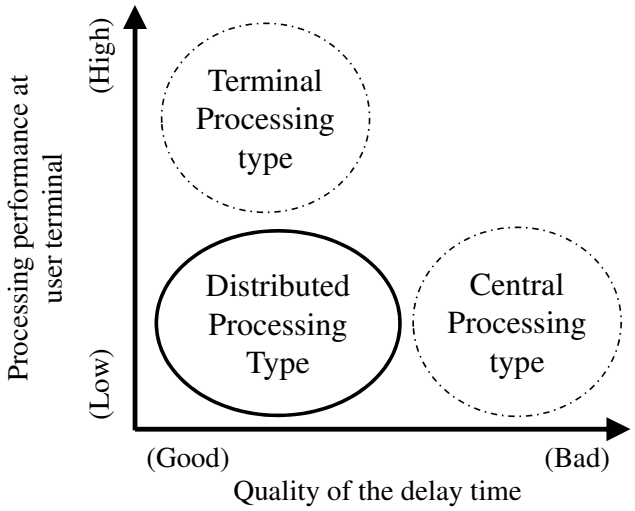


図 1.4　従来方式の適用領域と本研究の狙い

1.4.2　本研究が解決する課題

前述の 1.2.1 章における分散処理の関連研究は，処理方式やイベント順序性保証に関する研究であり，分散処理を通信サービスに適用した場合のエンド-エンドの遅延特性に着目した通信方式の研究はされていない．本研究では，広域なネットワーク内に複数のサーバが存在する場合に，エンド-エンドの遅延時間を 低減するためのサーバの選択法とユーザ端末-サーバ間およびサーバ-サーバ間の通信方式に着目した応用研究であり，分散処理における処理の順序性やイベントの順序補正については，従来研究のアルゴリズムに沿った実現方式を採用している．

1.5　本論文の構成

本論文の構成を図 1.5 に示す．

1 章では，研究背景としての前提条件および関連研究を述べる．

2 章では，1 章の研究の背景を受けて，エッジコンピューティング技術を活用すること

で，従来の課題である遅延特性を改善する提案方式を述べる．

3 章では，2 章の提案方式を用いた場合の遅延特性の改善効果について評価する．

4 章では，遅延特性に厳しいリアルタイム型アプリケーションへの適用を考慮して，2 章で定式化した線形計画法の最適化問題について遅延許容時間を考慮した拡張を行う．

5 章では，ネットワーク輻輳を考慮して，4 章の最適化問題について，遅延変動率を考慮した拡張を行う．提案方式の第二の拡張として，ネットワーク輻輳時の遅延変動を考慮する．

6 章では，複数のユーザ端末が適宜追加されるアプリケーションへの適用を考慮して，逐次参加型のユーザ参加方法について拡張を行う．

7 章では，結論と今後の課題について述べる．

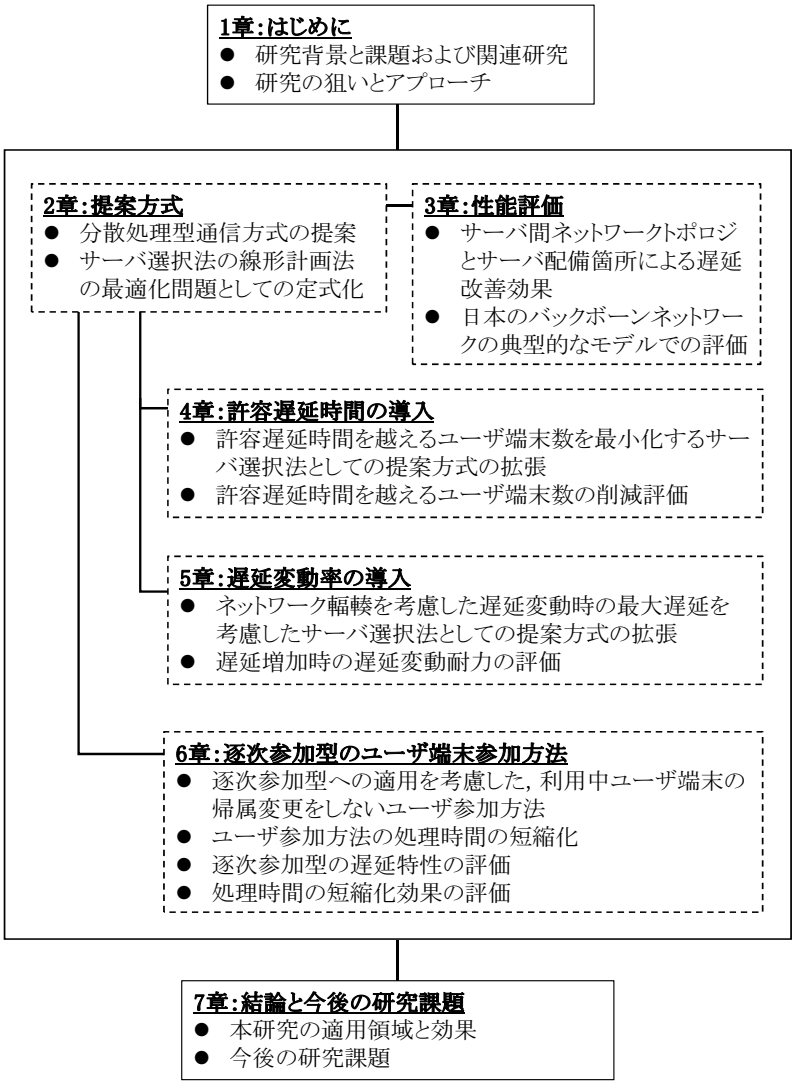


図 1.5　本論文の構成

第 2 章

提案方式

2.1　提案方式の動作条件

提案方式は，広域なネットワークを介して複数のユーザが利用可能な通信サービスのエンド-エンドの遅延特性を改善することを目的とする．アプリケーションは，　1.2.3 章に記載のエッジコンピューティングで用いられるユーザに近いロケーションに配備された複数のサーバで分散処理される．ユーザ端末は複数のサーバから最適な 1 つのサーバを選択し，選択したサーバ上で動作しているアプリケーションと通信を行う．アプリケーションが動作している複数のサーバ間では，自サーバ上で処理した結果を，他のサーバに同報送信を行うことで，複数のサーバ上で動作しているアプリケーション間でデータの送受信を行う．図 2.1 に従来方式と比較した提案方式におけるユーザ端末とサーバ間の通信方式を示す．図 2.1(a) は，サーバをセンタに配備した集中処理型通信方式で，(b) は端末同士が直接通信しあう方式である．図 2.1(c) が提案方式である分散処理型通信方式である．

イベント処理の順序性保証としては，分散処理による既存のアプリケーションへの変更を最小限にするため，1.2.1 章の分散処理は保守型同期アルゴリズムを採用し，1.2.1 章に記載のイベントの順序補正は Total Ordering を採用し，アプリケーション処理の前段でイベントの順序補正を全サーバ共通的に行うことで，アプリケーションが集中処理型で動作している場合と同様な条件での動作を行う．イベントの順序補正方式については 2.2 章で，アプリケーションの分散処理システム上での動作については，2.3 章で詳細を記載する．

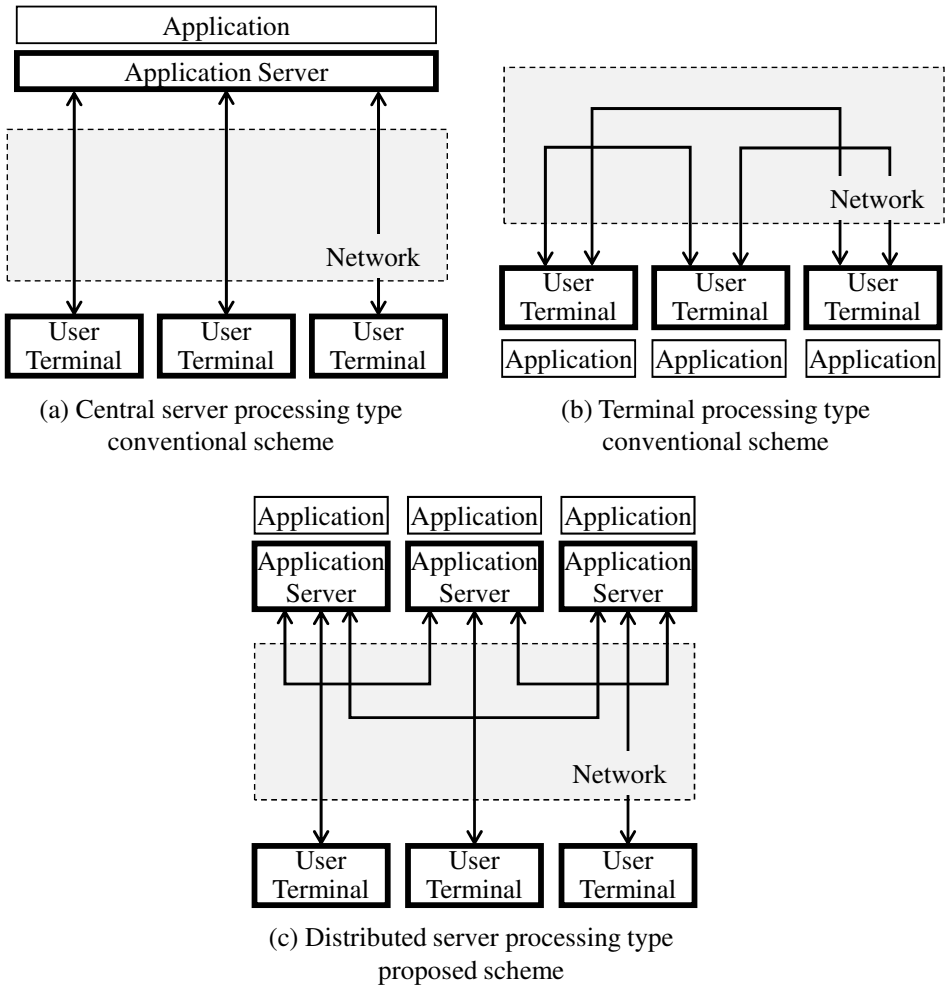


図 2.1　集中処理型と分散処理型の通信方式

2.2　仮想時刻を用いたイベント順序補正処理

イベント処理の順序性を保証するために，現在時刻 T に補正時間を加えた仮想時刻をアプリケーションが動作するサーバ上で導入する．補正時間は，あらかじめ測定した各ユーザ端末とサーバ間の遅延時間に基づいて事前に計算をしておく．各ユーザ端末のイベントをサーバ上の仮想時刻で発生順序を再現し，1.2.1 章に記載の Total Ordering を実現することで，各サーバ上で動作するアプリケーションは，全ユーザ端末の順序性が保証された仮想時刻で処理を行う．

2.2.1　ユーザ端末が選択したサーバにおけるイベント補正処理

ユーザ端末 p が通信するサーバとのリンク遅延時間を D_p と表す．全ユーザ端末の D_p の最大値を $D_U^{\max}$ と表すと，ある時刻での全ユーザ端末のイベントは $D_U^{\max}$ 待てば，必ず通信するサーバに到着する．つまり，サーバの仮想時刻を現在時刻 T から $D_U^{\max}$ 遅らせれば，全サーバでユーザ端末のイベント発生順序の再現が可能である．サーバでの処理としては，ユーザ端末 p からの到着したイベントは，サーバに到着後，$D_U^{\max}$- D_p 時間の待ち合わせを行った後で，アプリケーションが処理する．

図 2.2 の例では，ユーザ端末 A が最もサーバとの遅延時間が大きい端末であり，$D_U^{\max} = D_a$ がとなっており，仮想時刻は，$T+D_a$ と表される．仮想時刻 $T+D_a$ で，実際のイベント発生順序に並び替えを行うため，端末 B のイベントはサーバ到着後 $D_a - D_b$ 待ち合わせた上で処理される．

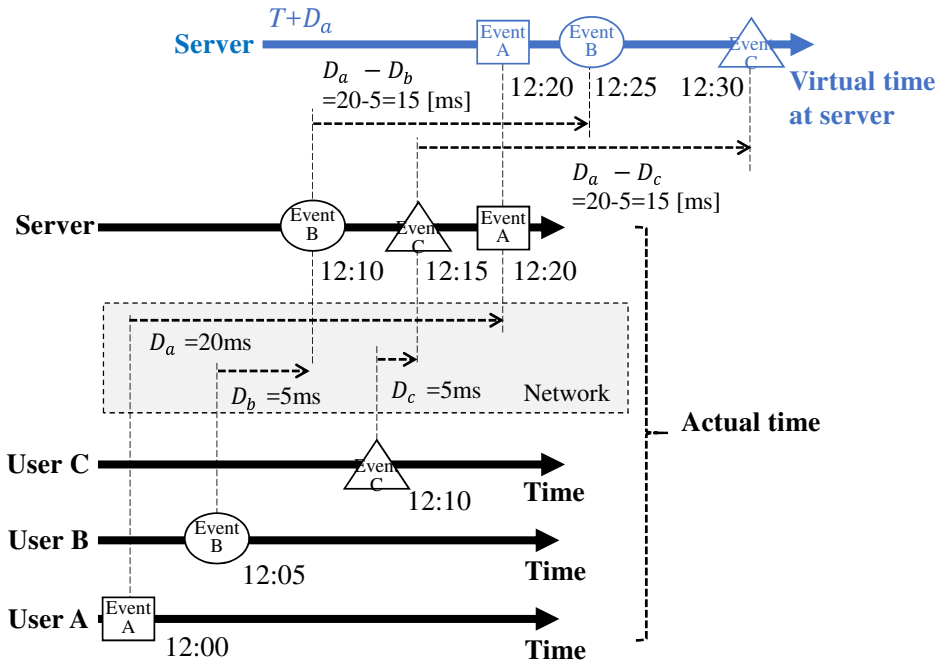


図 2.2　サーバでの順序補正処理

図 2.2 の例で処理の概要を説明する．簡単化のため，ms 単位の時刻として各時刻情報を扱う．ユーザ端末 3 台がサーバを選択している条件で，各ユーザ端末でのイベント発生時刻をユーザ端末 A は 12:00，ユーザ端末 B は 12:05，ユーザ端末 C は 12:10 と仮定する．ユーザ端末とサーバ間の通信遅延時間は，それぞれユーザ端末 A は 20ms，ユーザ

端末 B とユーザ端末 C は 5ms とすると，サーバへの到着時刻はユーザ端末 B は 12:10，ユーザ端末 C は 12:15，ユーザ端末 A は 12:20 となりイベントの到着順序が発生時と異なった順序となっている．本ケースでは，ユーザ端末 A が最もサーバとの通信遅延時間が大きい端末であり，$D_U^{\max} = D_a$ =20ms となっており，仮想時刻は $T + D_a$ と表される．仮想時刻上でのイベント発生順序を再現するため，$D_U^{\max}$- D_p の待ち合わせを行う．ユーザ端末 A は，$D_U^{\max}$- D_a=20ms-20ms=0ms で待ち合わせなくサーバ到着時刻の 12:20 で仮想時刻上で処理する．ユーザ端末 B は，$D_U^{\max}$- D_b=20ms-5ms=15ms でサーバ到着時刻の 12:10 から 15ms を待ち合わせた 12:25 で仮想時刻上で処理する．ユーザ端末 C は，$D_U^{\max}$- D_c=20ms-5ms=15ms でサーバ到着時刻の 12:15 から 15ms を待ち合わせた 12:30 で仮想時刻上で処理する．上記のとおり，仮想時刻上では各ユーザ端末でのイベント発生時刻に $D_U^{\max}$=20ms が加算されたユーザ端末 A は 12:20，ユーザ端末 B は 12:25，ユーザ端末 C は 12:30 の時刻で，イベント発生順序が再現される．図 2.2 の例は，単独のサーバ上での処理例だが，$D_U^{\max}$ を全ユーザ端末の D_p の最大値とすることで，分散処理する各サーバの時刻が 1.2.1 章に記載の UTC と同期することで仮想時刻上で全ユーザ端末のイベント順序が再現可能となる．

2.2.2　サーバ間のイベント補正処理

サーバ仮想時刻である $T + D_U^{\max}$ の時刻では，全ユーザ端末のイベントの順序性は再現されているものの，各ユーザ端末のイベントが複数のサーバに分散されているため，サーバ間でイベントの同期を取る必要がある．

サーバ間リンクの遅延時間の最大値を $D_S^{\max}$ とすると，各サーバの処理結果は全サーバに同報転送されるため $D_S^{\max}$ 待てば，必ず他のサーバへ到着することになる．つまり，サーバの仮想時刻を $T + D_U^{\max}$ からさらに $D_S^{\max}$ 遅らせれば，自サーバを選択しているユーザ端末のみでなく，全ユーザ端末のイベントが行き渡ることとなる．

図 2.3 の例で処理の概要を説明する．簡単化のため，ms 単位の時刻として各時刻情報を扱う．サーバ 3 台で分散処理をしており，図ではサーバ 3 での処理について記載する．各サーバでのイベント処理時刻をサーバ 1 は 12:00，サーバ 2 は 12:05，サーバ 3 の自サーバ上は 12:15 と仮定する．サーバ 3 と他サーバとの通信遅延時間が，サーバ 1 は 20ms，サーバ 2 は 5ms とすると，サーバ 3 への到着時刻はサーバ 1 は 12:20，サーバ 2 は 12:10 となる．本ケースでは，サーバ 1 とサーバ 3 間が最も通信遅延時間が大きいサーバ間リンクとすると，$D_S^{\max}$=D_{31}=20ms となっており，仮想時刻は T+D_{31} と表される．サーバ i において，仮想時刻上でのイベント発生順序を再現するため，サーバ j からのイベントについては，$D_S^{\max}$ - D_{ij} の待ち合わせを行う．本ケースのサーバ 3 においては，サー

バ 1 は $D_S^{\max}$ -　D_{31}=20ms-20ms=0ms で待ち合わせなくサーバ到着時刻の 12:20 で仮想時刻上で処理する．サーバ 2 は，$D_S^{\max}$ -　D_{23}=20ms-5ms=15ms でサーバ 3 到着時刻の 12:15 から 15ms 待ち合わせを行った 12:30 で仮想時刻上で処理する．自サーバのサーバ 3 は，$D_S^{\max}$=20ms=20ms で，発生時刻 12:15 から 20ms 待ち合わせを行った 12:35 で仮想時刻上で処理する．上記のとおり，仮想時刻上では各ユーザ端末でのイベント処理時刻に $D_S^{\max}$=20ms が加算されたサーバ 1 は 12:20，サーバ 2 は 12:25，サーバ 3 は 12:35 の時刻で発生順序が再現される．図 2.3 の例は，サーバ 3 での処理例だが，$D_S^{\max}$ を分散処理する全サーバ間リンクの最大値とすることで，分散処理する各サーバの時刻が 1.2.1 章に記載の UTC と同期することで仮想時刻上でイベント順序が再現可能となる．

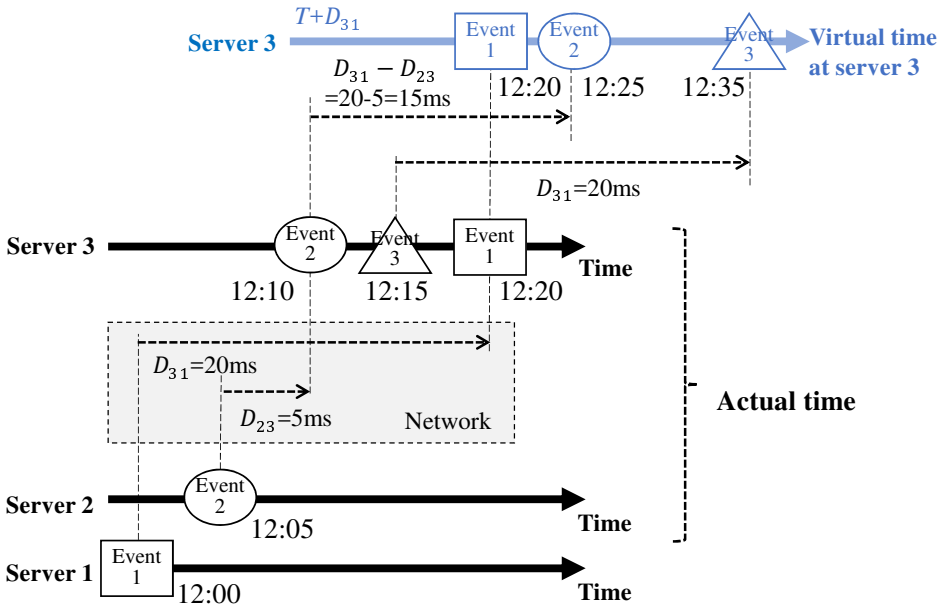


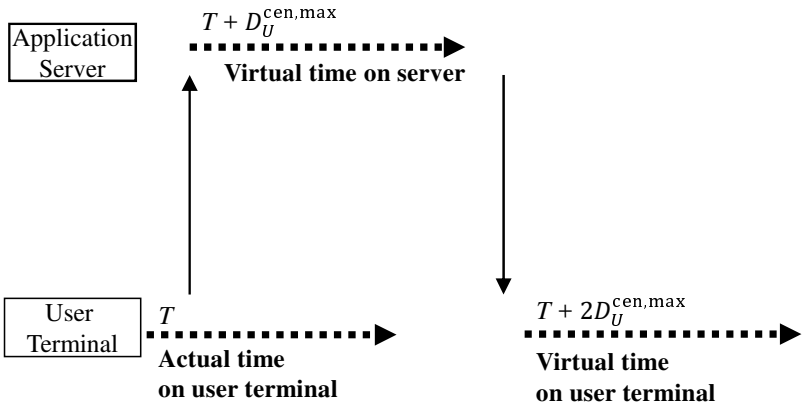
図 2.3　サーバ間の順序補正処理

2.2.3　ユーザ端末でのイベント補正処理

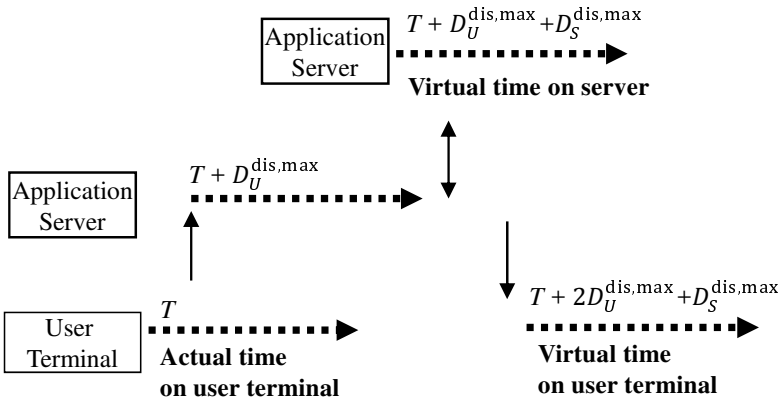
各サーバの処理結果は，ユーザ端末に送信されることなるが，ユーザ端末とサーバ間遅延時間の最大値である $D_U^{\max}$ 待てば，必ず全ての端末に各サーバの処理結果が配信されることとなる．つまり，ユーザ端末では，サーバ仮想時刻から $D_U^{\max}$ 時刻を遅らせることで，仮想時刻上での全ユーザ端末のイベントの順序性が再現可能である．実際は，サーバから $D_U^{\max}$ -　D_p 時間の待ち合わせを行った後で，各ユーザ端末へ処理結果を送信することで，ユーザ端末での待ち合わせ処理が不要となる．

2.2.4 サーバとユーザ端末での補正時間と仮想時刻

集中処理型通信方式における $D_U^{\max}$ を $D_U^{\mathrm{cen,max}}$ と表し，分散処理型通信方式における $D_U^{\max}$ を $D_U^{\mathrm{dis,max}}$，$D_S^{\max}$ を $D_S^{\mathrm{dis,max}}$ と表す．図 2.4(a) に示すように，集中処理型の通信方式では，ユーザ端末とサーバ間リンク遅延時間の最大値である $D_U^{\max}$ でサーバ上の仮想時刻を補正するため，以下のように仮想時刻を表すことができる．



(a) Virtual time at central processing scheme



(b) Virtual time at distributed processing scheme

図 2.4 サーバとユーザ端末での仮想時刻

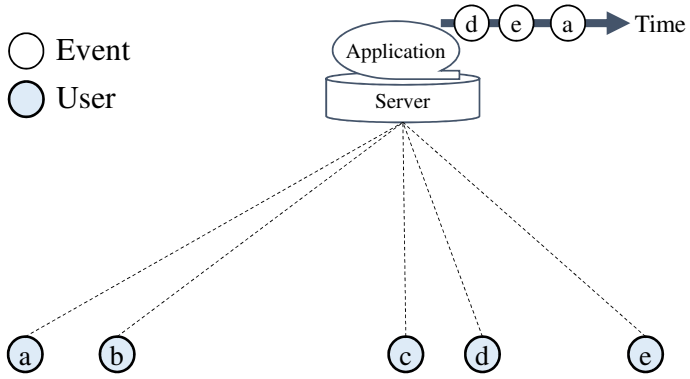
1. サーバ補正時間 $D_U^{\mathrm{cen,max}}$
2. サーバ仮想時刻 $T + D_U^{\mathrm{cen,max}}$
3. ユーザ端末補正時間 $2D_U^{\mathrm{cen,max}}$
4. ユーザ端末仮想時刻 $T + 2D_U^{\mathrm{cen,max}}$

分散処理型の通信方式では，図 2.4(b) に示すように，ユーザ端末のイベントがサーバに到着する待ち時間 $D_U^{\max}$ に，サーバ間で同報通信を行いサーバ間でデータ同期を行う待ち時間 $D_S^{\max}$ を加えた $D_U^{\max} + D_S^{\max}$ がサーバ上の補正時間となり，以下のように仮想時刻を表すことができる．

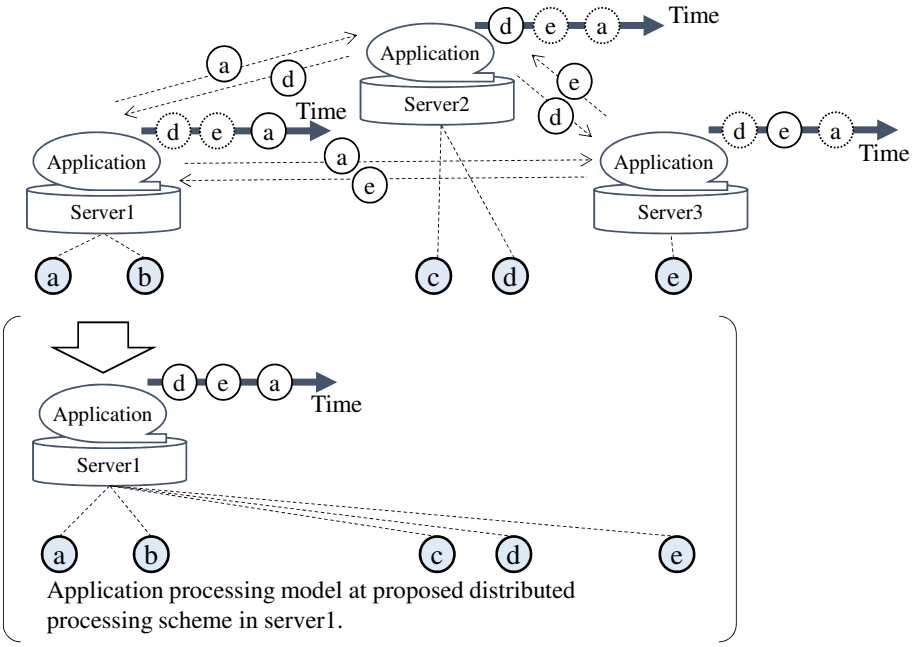
1. サーバ補正時間 $D_U^{\mathrm{cen,max}} + D_S^{\mathrm{dis,max}}$
2. サーバ仮想時刻 $T + D_U^{\mathrm{cen,max}} + D_S^{\mathrm{dis,max}}$
3. ユーザ端末補正時間 $2D_U^{\max} + D_S^{\mathrm{dis,max}}$
4. ユーザ端末仮想時刻 $T + 2D_U^{\max} + D_S^{\mathrm{dis,max}}$

2.3　提案方式におけるアプリケーションの動作

図 2.5 に集中処理型と比較したアプリケーションの動作例を示す．集中処理型では，１つのセンタサーバ上で動作しているアプリケーションで全ユーザのイベントを集中的に処理をするが，提案方式ではアプリケーションは複数のサーバ上で動作し，自サーバを選択しているユーザ端末以外のイベントは，他サーバ経由で通知される．



(a) Example of application processing model at central processing scheme.



(b) Example of application processing model at proposed distributed processing scheme.

図 2.5　集中処理型と分散処理型のアプリケーションの動作例

各サーバでは，全ユーザ端末のイベントがイベント発生時の順序が再現されて処理されるため，分散処理をしている各サーバは同一のイベントを同一の条件で，それぞれ処理をする．図 2.5 の例では，サーバ 1，サーバ 2，サーバ 3 とも同一のイベント条件でアプリケーションが動作しており，特に，サーバ 1 に着目すると，集中処理型と同じように全ユーザ端末をサーバ 1 を選択しているのと同じようにアプリケーションが動作すると想定する．

2.4　分散処理型通信方式におけるサーバ選択問題

2.4.1　提案方式におけるサーバ選択問題の必要性

提案方式においては，ユーザ端末が複数のサーバと接続されている場合は，選択されたサーバによって遅延時間 D_p の値が異なるため決定される $D_U^{\mathrm{dis,max}}$ が異なる．また，分散処理するサーバの選択によってサーバ間遅延時間 d_{ij} が異なるため決定される $D_S^{\mathrm{dis,max}}$ が異なる．これらのユーザ端末が選択するサーバおよびアプリケーションを動作させるサーバを決定するサーバ選択問題をユーザ端末補正時間を最小化する条件で定式化する必要がある．図 2.6 に，同一のトポロジにおける 2 種類のサーバ選択例を示す．図 2.6(b) では，2 つのサーバ選択において，ユーザ端末補正時間 $2D_U^{\max} + D_S^{\mathrm{dis,max}}$ が最小となっていないことがわかる．

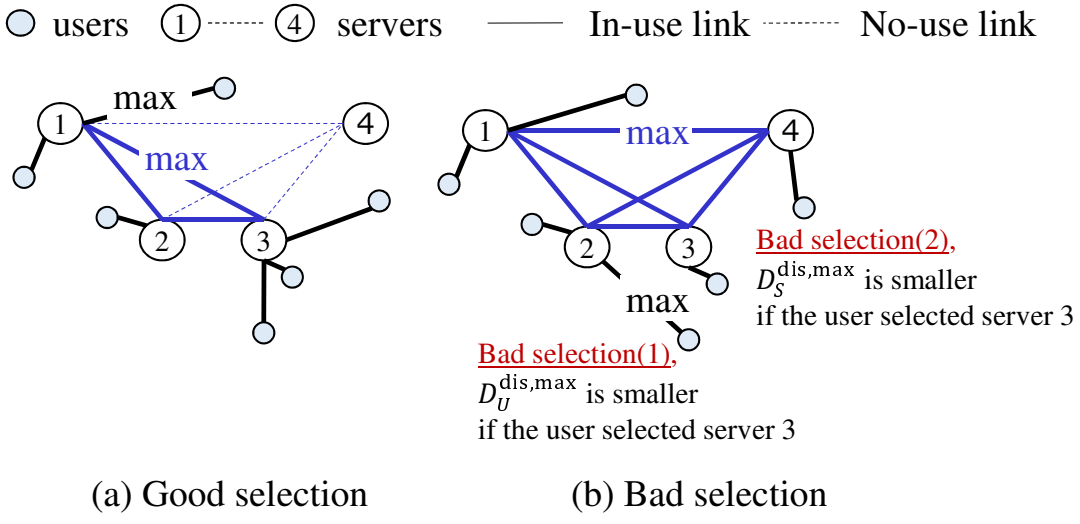


図 2.6　サーバ選択問題の必要性

2.4.2 ネットワーク構成モデルと定式化

本論文では，サーバの選択法を，ユーザ端末補正時間を最小化することで遅延時間を最小化する整数線形計画問題の最適化問題として定式化する．評価のためのネットワーク構成モデルとして，無向グラフ $G(V, E)$ と考える．V はユーザ端末およびサーバから構成される全ノードの集合，E はユーザ端末とサーバ間リンクおよびサーバとサーバ間リンクの全無向リンクの集合とする．ユーザ端末を表すノードの集合を V_U とし，ユーザ端末を表すノードを $p \in V_U$ で表す．サーバを表すノードの集合を V_S とし，サーバを表すノードを $i \in V_S$ で表す．ノードはユーザ端末とサーバのみであることから $V_U \cup V_S = V$ となり，ユーザ端末かつサーバというノードは存在しえないことから $V_U \cap V_S = \emptyset$ である．ユーザ端末とサーバ間リンクの集合を E_U とし，ユーザ端末 $p \in V_U$ とサーバ $i \in V_S$ の間のリンクを $(p, i) \in E_U$ と表す．リンク $(p, i) \in E_U$ に関するバイナリ変数として u_{pi} を導入し，リンク (p, i) が利用可能な場合は $u_{pi}= 1$，リンク (p, i) が利用できない場合は，$u_{pi}= 0$ と表す．サーバとサーバ間リンクの集合を E_S とし，サーバ $i \in V_S$ とサーバ $j \in V_S$ の間のリンクを $(i, j) \in E_S$ と表す．$(i, j) \in E_S$ は，すべてのサーバ $i \in V_S$ とすべてのサーバ $j \in V_S$ (ただし，$i \neq j$) の間に設定されるものとする．リンクは，ユーザ端末とサーバ間およびサーバとサーバ間のみであることから $E_U \cup E_S = E$ であり，ユーザ端末とサーバ間であり，かつサーバとサーバ間というリンクは存在しないため $E_U \cap E_S = \emptyset$ である．分散処理を行うサーバの最大サーバ台数を Y_{MAX} と表す．

ユーザ端末 $p \in V_U$ とサーバ $i \in V_S$ 間リンクの遅延時間を d_{pi} と表す．使用されるリンクの d_{pi} の値のうち $(p, i) \in E_U$ に関する最大値を $D_U^{\max}$ と表す．サーバ $i \in V_S$ とサーバ $j \in V_S$ 間リンク (i, j) の遅延時間を d_{ij} と表す．使用されるリンクの d_{ij} の値のうち $(i, j) \in E_S$ に関する最大値を $D_S^{\max}$ と表す．また，$i \in V_S$ に関するパラメータ M_i は，サーバ i が受け入れられる最大のユーザ端末数を表す．利用されないサーバや利用されないリンクもありえることから，x_{kl} は，リンク $(k, l) \in E$ に関するバイナリ変数であり，リンク (k, l) が利用されれば $x_{kl} = 1$，利用されなければ $x_{kl} = 0$ とする．y_i は，$i \in V_S$ に関するバイナリ変数であり，サーバ i が少なくとも 1 台のユーザ端末に使用されれば $y_i = 1$，そうでなければ $y_i = 0$ とする．$y_k \cdot y_l = x_{kl}$ とする．ユーザ端末が選択するサーバの決定法は，次式で定式化される分散処理サーバの選択最適化問題として扱うことができる．

$$\text{Objective} \ \min(2D_U^{\max} + D_S^{\max}) \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{i \in V_S} u_{pi} x_{pi} = 1, \forall p \in V_U \tag{2.1b}$$

$$\sum_{p \in V_U} u_{pi} x_{pi} \leq M_i, \forall i \in V_S \tag{2.1c}$$

$$\sum_{i \in V_S} y_i \leq Y_{\mathrm{MAX}} \tag{2.1d}$$

$$u_{pi} x_{pi} d_{pi} \leq D_U^{\max}, \forall (p,i) \in E_U \tag{2.1e}$$

$$x_{ij} d_{ij} \leq D_S^{\max}, \forall (i,j) \in E_S \tag{2.1f}$$

$$y_i \geq u_{pi} x_{pi}, \forall p \in V_U, i \in V_S \tag{2.1g}$$

$$y_i + y_j - 1 \leq x_{ij}, \forall (i,j) \in E_S \tag{2.1h}$$

$$x_{ij} \leq y_i, \forall i \in V_S, (i.j) \in E_S \tag{2.1i}$$

$$x_{ij} \leq y_j, \forall j \in V_S, (i,j) \in E_S \tag{2.1j}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E_S \cup E_U \tag{2.1k}$$

$$y_i \in \{0,1\}, \forall i \in V_S \tag{2.1l}$$

式 (2.1a) は，$D_U^{\mathrm{dis,max}}$ の値の 2 倍と $D_S^{\mathrm{dis,max}}$ の合計値を目的関数とし，これを最小とすることを示す．式 (2.1b) は，ユーザ端末とサーバ間に 1 つのリンクが使用されることを示す．式 (2.1c) は，サーバ i が受け入れられるユーザ端末数は，M_i を超えないことを示す．式 (2.1d) は，分散処理に利用可能なサーバの最大数を示す．式 (2.1e) は，式 (2.1a) の目的関数を最小化する問題において，$D_U^{\max}$ は，使用されるリンク $(p,i) \in E_U$ に関する d_{pi} の最大値であることを示す．式 (2.1f) は，式 (2.1a) の目的関数を最小化する問題において，$D_S^{\max}$ は，使用されるリンク $(i,j) \in E_S$ に関する d_{ij} の最大値であることを示す．

式 (2.1g) は，サーバ i が少なくとも 1 つのユーザ端末に使用されれば 1 とすることを示す．式 (2.1h) -(2.1j) は，$y_k \cdot y_l = x_{kl}$ を示す線形式を用いた表現である．式 (2.1k) ，式 (2.1l) は，x_{ij}，y_{ij} がそれぞれバイナリ変数であることを示す．

集中処理型通信方式については，利用可能なサーバ数を 1 つに決定することになるため，式 (2.1d) において $Y_{\mathrm{MAX}} = 1$ の条件とすることで，集中処理型通信方式におけるユーザ端末補正時間を求めることができる．また，集中処理するサーバが特定サーバに決定している場合は，集中処理を行うサーバ以外のサーバのパラメータを M_i=0 の条件で解くことで，特定サーバを利用した集中処理型通信方式のユーザ端末補正時間の算出に用いることができる．表 2.1 に定式化した最適化問題のパラメータと決定変数を示す．

表 2.1　2.4.2 章で定式化した最適化問題のパラメータ，決定変数一覧

パラメータ	決定変数
Y_{MAX}, M_i, d_{pi}, u_{pi}, d_{ij}	$D_U^{\max}$, $D_S^{\max}$, y_i, x_{pi}

2.5　サーバ選択問題の計算複雑度評価

2.4.2 章で扱った分散処理サーバの選択最適化問題 (DSSO：Distributed server selection optimization problem) について，DSSO が Non-deterministic-Polynomial (NP)-hard (NP 困難) であることを，次に示す分散サーバ選択決定問題 (DSSD: Distributed server selection decision problem) が NP-complete (NP 完全) であることを証明 [75] することによって示す．

分散処理サーバ選択決定問題 (DSSD)　サーバの集合 N，ユーザ端末の集合 P，サーバ i とサーバ j 間のリンク長 d_{ij}，ユーザ端末 $p \in P$ が選択したサーバ i とのリンク長 D_p，サーバあたりの最大ユーザ端末収容数 M_i が与えられたときに，D_p の最大値を $D_U^{\max}$，ユーザ端末から選択されたサーバにおける d_{ij} の最大値を $D_S^{\max}$ と表す．$w = 2D_U^{\max} + D_S^{\max}$ が $w \leq h$ (h は与えられる定数) を満たすためのユーザ端末が選択するサーバの決定が可能であるか？

定理. DDSD は，NP-complete である．

証明. はじめに，DDSD のインスタンスが多項式時間で判定可能であることから，DDSD は NP であることを示す．p 台のユーザ端末が 1 台のサーバと接続されている場合，ユーザ端末-サーバ間リンク遅延時間の最大値　$D_U^{\max}$ を求める計算量は，$O(p)$ である．選択されたサーバが N 台の条件では，サーバ間リンク遅延時間のの最大値 $D_S^{\max}$ を求める計算量は $O(N^2)$ 以内である．w は，$2D_U^{\max} + D_S^{\max}$ により，$O(1)$ で計算できる．したがって，$w \leq h$ を判定する計算量は $O(N^2 + P)$ である．

NP-complete であることが既知である 3-SAT 問題 [76] が DDSD に多項式時間で帰着可能であることを示す．

3-SAT 問題のインスタンス　任意の 3-SAT 問題のインスタンスを以下のように構成する．

- $3k$ 個の要素を含むブール変数 V_{ij} の集合を X と表す．　($i = 1, 2, \cdots, k$,$j = 1, 2, 3$)
- 集合 X 上の項からなる 3 個つ論理和は，$V_{i1} \vee V_{i2} \vee V_{i3}$ を C_i と表す．
- クローズ C_i の k 個の論理積は，$C_1 \wedge C_2 \wedge \cdots \wedge C_k$ と表す．

DDSD のインスタンス　分散処理サーバ決定問題のインスタンスを以下のように構成する．構成例を図 2.7 に示す．

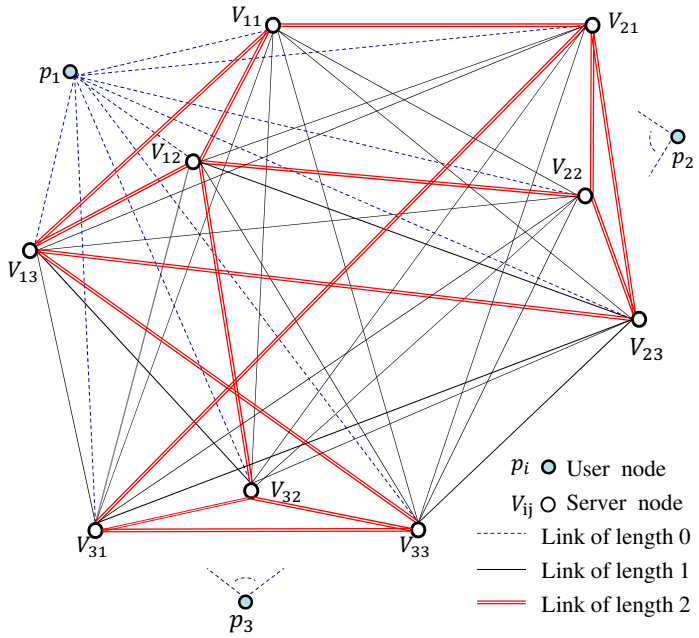


図 2.7　Graph G corresponding to 3-SAT problem with three clauses, which are $C_1 = x_1 \vee x_2 \vee x_3$, $C_2 = \overline{x_1} \vee \overline{x_2} \vee \overline{x_3}$, and $C_3 = x_1 \vee \overline{x_2} \vee \overline{x_3}$, where x_1, x_2, and x_3 are boolean variables.

- グラフ G は，$3k$ 個のサーバと k 個のユーザ端末からなるノードで構成される．
- サーバ-サーバ間とユーザ端末-サーバ間はリンクで接続される．
- $3k$ 個のサーバノードは，3 個のサーバノード V_{ij} から構成される k セット $(i = 1, 2, \cdots, k\ ,\ j = 1, 2, 3)$ で構成される．
- $i \neq i'$ であり，かつ，V_{ij} は，$V_{i'j'}$ の否定でないリンク $(V_{ij}, V_{i'j'})$ の長さを 1 とする．そうでなければ，$V_{ij}, V_{i'j'}$ 間のリンク長は 2 とする．
- すべてのユーザ端末ノード p は全てのサーバノード i とのリンク長を 0 とする．
- 全サーバの最大ユーザ端末収容数は M_i=1 とする．

3-SAT 問題のインスタンスが YES のとき，DDSD インスタンスが YES となることを示す．3-SAT 問題のインスタンスが YES であるならば，各クローズには真となるブール変数が少なくとも 1 つ以上存在し，各クローズにおいて真となる任意のブール変数を 1 つ選択することができる．グラフ G においては，k 個の各クローズ内の V_{i1}, V_{i2}, V_{i3} の

うち少なくとも 1 つ以上が選択可能なサーバとして存在し，各クローズで 1 つ選択されたサーバノード同士（合計 k 個）のすべてのリンク長が 1 となる．この条件で，最大遅延時間 w は $w = 1$ となり，$w \leq h$ を満たす．したがって，このとき，DSSD のインスタンスは，YES となる．

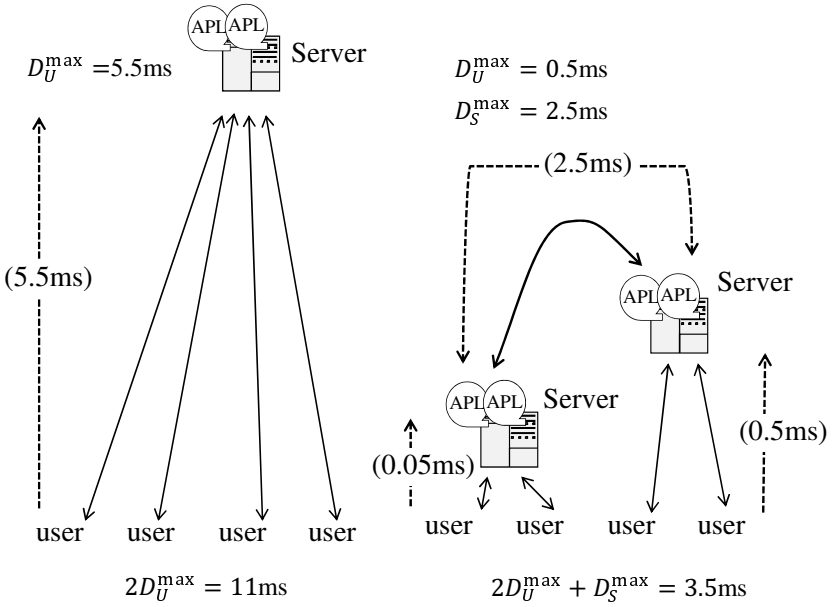
次に，DSSD のインスタンスが YES のとき，3-SAT 問題のインスタンスが YES であることを示す．DDSD のインスタンスが YES であるならば，全てのサーバノードはリンク長 1 でそれぞれ接続され，グラフ G において，これらの条件を満たすための選択されたサーバノード k の組み合わせは，3-SAT 問題のすべてのクローズを 1 にする条件を満たしている．したがって，このとき，3-SAT 問題のインスタンスは，YES となる．

□

上記の DSSD が NP 完全であることが証明されたので，DSSO は NP 困難である．したがって，DSSO に対する効率的な (多項式時間で解く) 解法は，現在のところ発見されていない．本研究では，DSSO を線形計画問題として解くアプローチを採用する．

2.6　集中処理型と分散処理型の比較例

具体的な事例として，図 2.8，図 2.9 に示す簡単なネットワークモデルを考える．

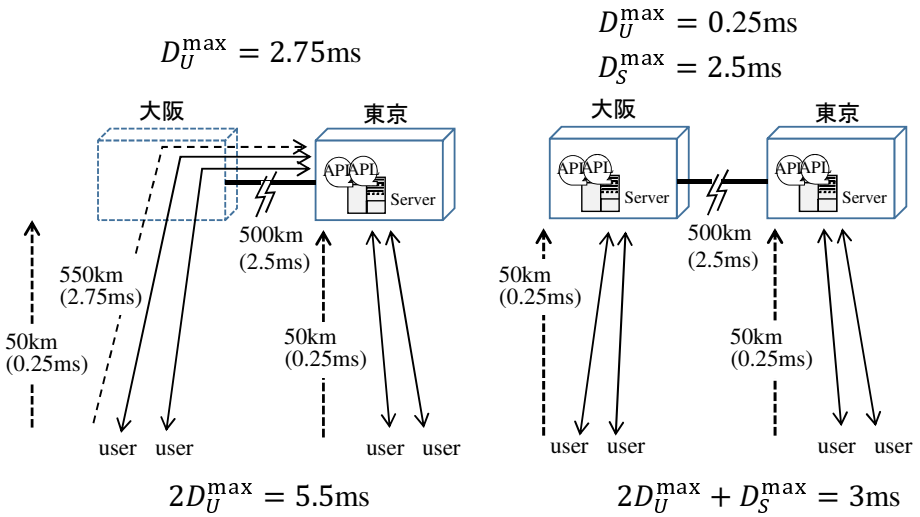


(a) Central processing scheme　(b) Distributed processing scheme

図 2.8　集中処理型と分散処理型の比較例 (1)

図 2.8 では，集中処理型通信方式では $D_U^{\mathrm{cen,max}}$=5.5ms から，ユーザ端末仮想時刻は T+11ms となる．一方，分散処理型通信方式では，ユーザ端末とサーバ間遅延時間の最大値が 0.5ms となり $D_U^{\mathrm{dis,max}}$=0.5ms となり，サーバ間遅延時間の最大値は 2.5ms のため $D_S^{\mathrm{dis,max}}$=2.5ms となることから，ユーザ端末補正時刻は T+3.5ms となり集中処理型と比較し，集中処理するサーバがユーザよりより遠い場所に配備されていることから遅延時間が大きく改善されている．

具体的な事例として，図 2.9 に示す東京と大阪のサーバをユーザ端末が選択している場合を考える．東京と大阪間の伝送距離を 500km で遅延時間を 2.5ms として，東京，大阪ともサーバから伝送距離 50km で 0.25ms の遅延時間の距離に最も遠いユーザ端末が接続されているとする．本ケースでは，集中処理型通信方式では $D_U^{\mathrm{cen,max}}$=2.75ms から，ユーザ端末仮想時刻は T+5.5ms となる．分散処理型通信方式では，$D_U^{\mathrm{dis,max}}$=0.25ms, $D_S^{\mathrm{dis,max}}$=2.5ms となることから，ユーザ端末補正時刻は T+3ms となり集中処理型と比較し，遅延時間が改善されていることがわかる．



(a) Central processing scheme　　(b) Distributed processing scheme

図 2.9　集中処理型と分散処理型の比較例 (2)

2.7　まとめ

本章では，提案方式の通信処理とエンド-エンドの通信遅延時間を最小化する サーバ選択問題について述べた．提案方式は，アプリケーションをユーザ端末に近いロケーションに配備された複数のサーバで分散処理を行う．ユーザ端末は複数のサーバから最適な 1 つのサーバを選択し，サーバ間では処理結果を同報送信する．提案方式では．イベント処理の順序性を保証するために，現在時刻から補正時間分を遅らせた全サーバで共通の仮想時刻をアプリケーションが動作するサーバで計算し，仮想時刻上でイベントの発生順序を再現する．ネットワーク内の複数のサーバから，エンド-エンドの通信遅延時間を最小化するサーバを決定するためのサーバ選択問題が計算複雑度の評価から，本問題が NP 困難であることを示す．サーバ選択問題を線形計画問題として定式化し，エンド-エンドの通信遅延時間を最小化する仮想時刻と各ユーザ端末が選択するサーバを定式化した線形計画問題を解くことで決定する．

第 3 章

性能評価

3.1　評価項目と評価方法

提案方式の性能評価にあたり，分散処理型通信方式の遅延時間の改善効果を，遅延特性比 R として，以下の式 (3.1a) で定義する．遅延特性比 R は，0 から 1 の間の値をとり，R が小さいほど遅延削減効果が高い．

$$R \ = \frac{\min(\ 2D_U^{\mathrm{dis,max}} + D_S^{\mathrm{dis,max}})}{\min\ 2D_U^{\mathrm{cen,max}}} \tag{3.1a}$$

提案方式の評価においては，ユーザ端末は特定のサーバを選択している前提で，以下の条件が遅延特性に与える影響について評価を行う (評価 1) ．

1) サーバ間のネットワークトポロジ

2) ユーザ端末とユーザ端末間の通信経路におけるサーバの配備箇所

1) についてはトポロジ依存性を比較するため，サーバのロケーションを固定し，サーバ間リンク群が形成するネットワークトポロジを変化させ，遅延特性の違いを比較する．2) については，ユーザ端末間通信におけるサーバの配備箇所が遅延特性に与える影響を評価するため，サーバのロケーションを固定し，サーバとユーザ端末の遅延時間を変化させることで，サーバ配備箇所に依存した遅延特性比を比較する．

本評価により，サーバをユーザ端末からのネットワーク遅延少ないロケーションのサーバに配備した場合と，ネットワークの集約拠点に近いサーバに配備した場合の遅延特性比への影響を比較する．

評価 1 ではユーザ端末は特定のサーバを選択していると仮定したが，ユーザ端末が複数のサーバとリーチャビリティを持っている場合は，$2D_U^{\mathrm{dis,max}} + D_S^{\mathrm{dis,max}}$ が最小となるように選択サーバを決定する必要がある．2.4.2 章で定式化した最適化問題を解くことで，複数のサーバから各ユーザ端末が選択するサーバを決定されることを確認する (評価 2)．本

論文では，定式化した最適化問題は線形計画法ソルバである GNU Linear Programming Kit (GLPK) [77] および CPLEX [78] を用いて解いた [79, 80].

3.2　ネットワークトポロジとサーバの配備箇所による評価 (評価 1)

サーバのトポロジを固定した条件で，ユーザ端末 p とサーバ i 間の遅延時間 d_{pi} において選択されたサーバの d_{pi} である D_p を変化させることで，サーバの配備箇所による遅延特性比の変化を評価する．サーバ i が受け入れられるユーザ端末数に制限はないものとして，$M_i = \infty$ として評価を行う．ネットワークトポロジによる特性の比較のためサーバのロケーションを，図 3.1 に記載の 4 タイプで評価を行う．各サーバは正 8 角形の頂点に配備されていると仮定し，サーバ間を結ぶリンク構成によって図 3.1 に示した 4 タイプのトポロジを構成する．

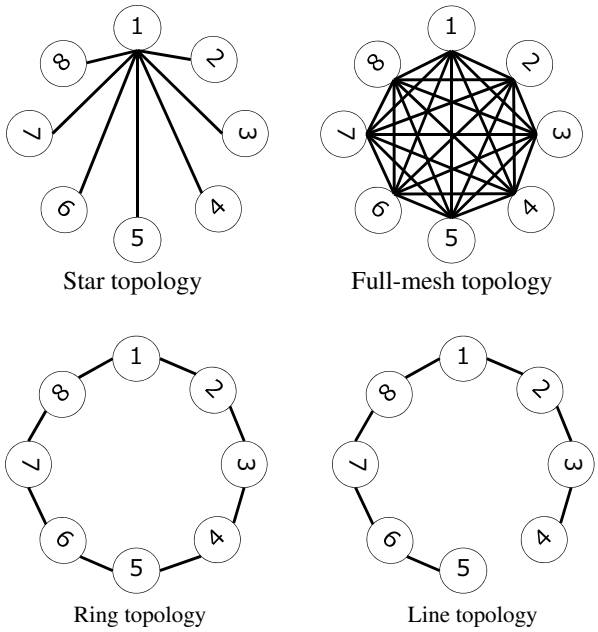


図 3.1　サーバのトポロジモデル

図 3.2 にスター型トポロジを例に，今回の評価方法を示す．ユーザ端末 p とサーバ i 間リンクの遅延時間を d_{pi} とする．8 つのサーバのうちサーバ i とサーバ j 間のリンク (i, j) は，トポロジ上の線分の長さを遅延時間として d_{ij} とする．集中処理型ではネットワーク上の集約拠点に配備された 1 つのサーバを利用するため 8 つのサーバのうち 1 つのみを

利用し，分散処理型は 8 つのサーバのうちで，ユーザ端末ごとに最も遅延時間の少ないサーバを選択していると仮定する．

本評価では，集中処理型と分散処理型の比較のために，図 3.2 に示すとおり，集中処理型のサーバ j とユーザ p と間に，ユーザ p とサーバ j の間にトラヒック中継のみを行うサーバ i があると仮定する．集中処理型の集約拠点のサーバ j とユーザ端末 p との遅延時間は，中継サーバ i を通るリンクでは $d_{pi} + d_{ij}$ と表される．2.4.2 章で定式化した式 (2.1a) における集中処理型モデルの $D_U^{\mathrm{cen,max}}$ については，遅延時間 $d_{pi} + d_{ij}$ の $(p,i) \in E_U, (i,j) \in E_S$ に関する使用されるリンクの最大値として表される．一方，分散処理型では，2.4.2 章の式 (2.1e) と式 (2.1f) で定式化したとおり，$D_U^{\mathrm{dis,max}}$ はユーザとサーバ間リンク (p,i) の遅延時間 d_{pi} の $(p,i) \in E_U$ に関する使用されるリンクの最大値であり，$D_S^{\mathrm{dis,max}}$ はサーバとサーバ間リンク (i,j) の遅延時間 d_{ij} の $(i,j) \in E_S$ に関する使用されるリンクの最大値として表される．

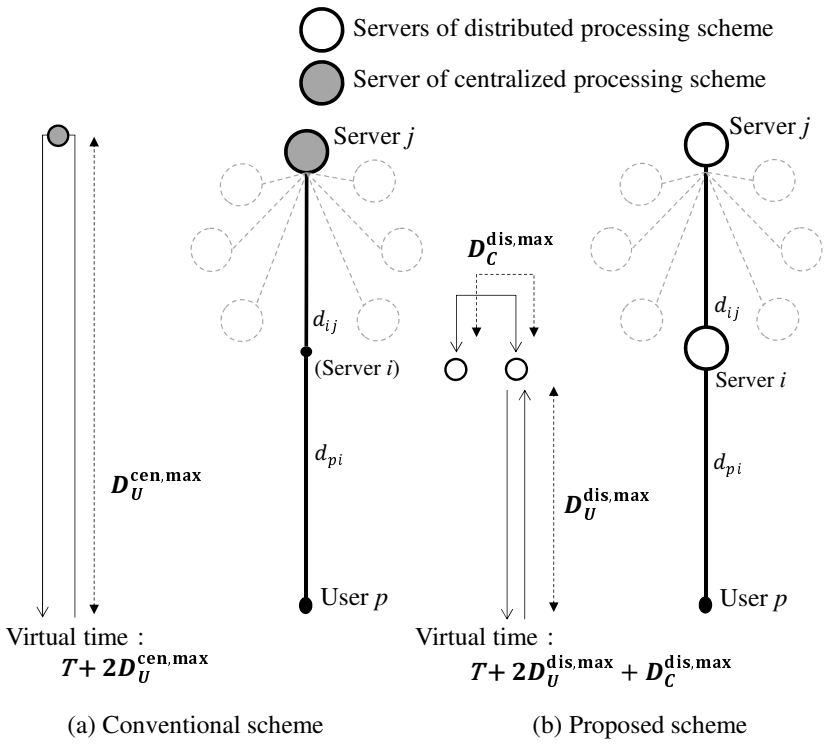
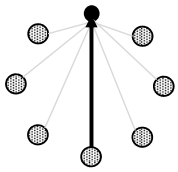
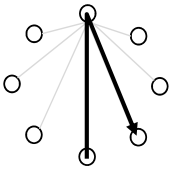
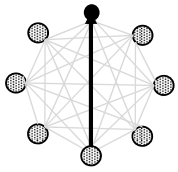
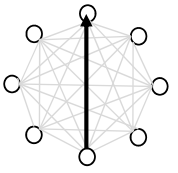
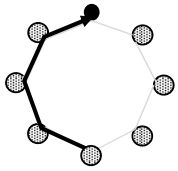
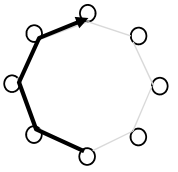
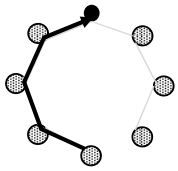
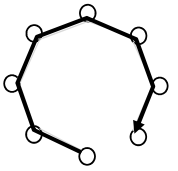


図 3.2　スター型トポロジにおける集中処理型と分散処理型と仮想時刻の例

本評価において，集中処理型における d_{ij} の最大値 $\max_{(i,j)\in E_S} d_{ij}$ を $d_S^{\mathrm{cen,max}}$ と表し，分散処理型における d_{ij} の最大値 $\max_{(i,j)\in E_S} d_{ij}$ を $d_S^{\mathrm{dis,max}}$ と表す．図 3.3 に，サーバが配置された正 8 角形の最大の対角線の遅延時間を 10ms と仮定した場合の各トポロジの

集中処理型と分散処理型の $d_S^{\mathrm{cen,max}}$ と $d_S^{\mathrm{dis,max}}$ の比較を示す．集中処理型では，集約拠点のサーバからリンクが直結していると仮定した中継サーバへ到達するリンクの最大値が $d_S^{\mathrm{cen,max}}$ となり，分散処理型では，全サーバ間で同報処理をするため全サーバ間リンクの最大値が $d_S^{\mathrm{dis,max}}$ となる．

	$d_S^{\mathrm{cen,max}}$ [ms]	$d_S^{\mathrm{dis,max}}$ [ms]	$\frac{d_S^{\mathrm{dis,max}}}{d_S^{\mathrm{cen,max}}}$
	Central (Conventional)	Distributed (Proposed)	
Star	 10	 $10+5\sqrt{2+\sqrt{2}}$ (≅19.2)	≅1.9
Full-mesh	 10	 10	=1.0
Ring	 $20\sqrt{2-\sqrt{2}}$ (≅15.3)	 $20\sqrt{2-\sqrt{2}}$ (≅15.3)	=1.0
Line	 $20\sqrt{2-\sqrt{2}}$ (≅15.3)	 $35\sqrt{2-\sqrt{2}}$ (≅26.8)	≅1.8

Central (Conventional)
● Location of servers
◍ Directly connected the link between user-server and server-server
Distributed (Proposed)
○ Location of servers

図 3.3　各トポロジにおける $d_{ij}^{\max}$ の値

図 3.4 に，図 3.3 のサーバトポロジにおいて，ユーザ端末とサーバ間遅延時間の最大値である $\max_{p \in V_U} D_p$ を 0.1~1,000ms まで変化させた場合の分散処理型通信方式におけるユーザ補正時間とを遅延特性比 R を示す.

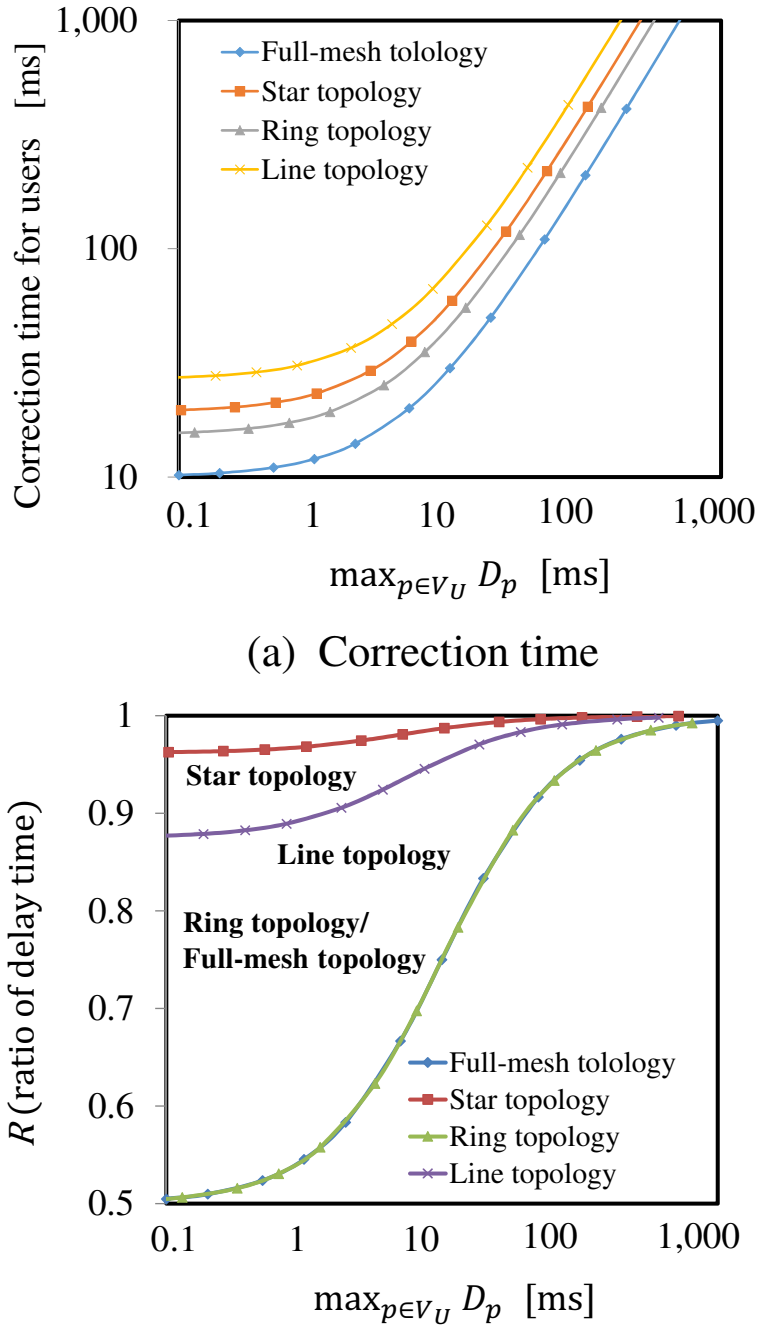


図 3.4　ユーザとサーバ間リンクの遅延時間によるユーザ補正時間と R の変化

本評価では，サーバ間の遅延時間を固定して，サーバからユーザ端末間の最大遅延時間を増加させているため，図 3.4(a) のとおり x 軸の増加に伴いユーザ補正時間が大きくなり，図 3.4(b) のとおり遅延特性比 R は，x 軸の増加に伴い悪化している．これらの結果からユーザ端末とネットワークの遅延時間が小さい拠点にサーバを配備したほうが，遅延時間の改善効果が高いことがわかる．

図 3.4(a) におけるユーザ端末補正時間の差は，図 3.3 の分散処理型通信方式の $d_S^{\max}$ 値 (スター型: 19.2ms，メッシュ型: 10ms，リング型: 15.3ms，直線型: 26.8ms) の差に基づいた差分が表れており，サーバ間リンクが最短距離に近い距離で結ばれているほうが遅延特性が良いことがわかる．

図 3.4(b) の遅延特性比は，図 3.3 の $d_S^{\max}$ の従来方式との比率 (スター型: 1.9，メッシュ型: 1.0，リング型: 1.0，直線型: 1.8) の差に基づいた差が表れている．評価結果から，遅延特性比はサーバ間のネットワークトポロジに依存して効果が変わり，$d_S^{\max}$ の値が集中処理型よりも小さくなるメッシュ型トポロジやリング型トポロジにおいて分散処理型の改善効果が高いことがわかる．

3.3　サーバ選択問題 (評価２)

ユーザ端末が複数のサーバと直接接続されているリンクを持っている場合の提案方式の評価として，ユーザ端末が一辺 20 の正方形内で分布していると仮定し，正方形の中心となる対角線の交点に図 3.1 に示したサーバトポロジの正 8 角形の中心を配置したモデルで評価を行う．正方形の左下を直交座標系の座標 $(0, 0)$ として，正方形の右上を座標 $(20, 20)$ として，図 3.5 を各サーバの座標とする．一様分布した 800 台のユーザ端末の座標軸上での分布を図 3.6 に示す．

本評価ではユーザ端末が全てのサーバと直接接続されているリンクを持っている条件で，2.4.2 章で定式化した最適化問題を解くことで，各ユーザ端末が選択するサーバが決定されることを示す．

評価 2 におけるパラメータを表 3.1 に示す．

表 3.1　評価 2 におけるパラメータ一覧

評価 2-1	評価 2-2
Y_{MAX}=8, d_{pi}=ユーザ端末とサーバの座標上の直線距離に比例した時間, u_{pi}=1, d_{ij}=評価 1 のサーバトポロジにおける座標上のサーバ間距離に比例した時間	
M_i=800 (i=1,2,3,4,5,6,7,8)	M_i=100 (i=1,3,5) M_i=800(i=2,4,6,7,8)

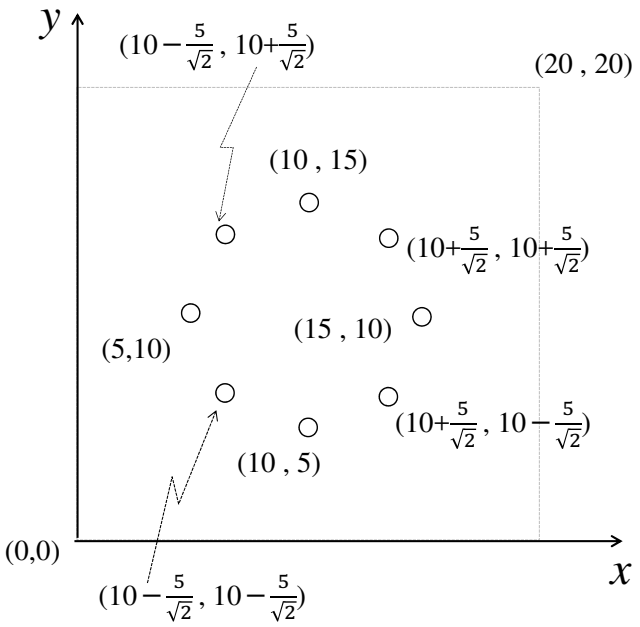


図 3.5　評価のための各サーバの座標情報

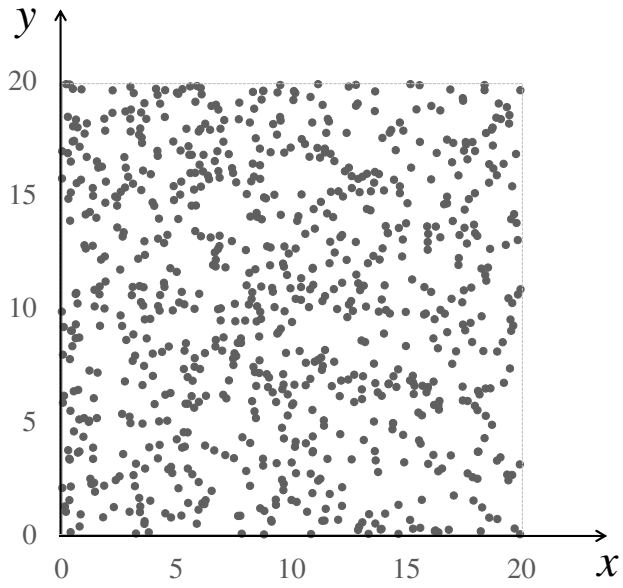


図 3.6　一様分布した 800 台のユーザ端末の座標軸上での分布

本評価では，遅延時間として，座標軸上での線分の長さを遅延時間として，ユーザ端末 p の各サーバの座標との直線距離をユーザ端末とサーバ間リンクの遅延時間とする．また，サーバ間リンクの遅延時間は，座標軸上の各サーバを図 3.1 に示す 4 つのトポロジを

構成する線分で結んだサーバ間の線分の長さをサーバ間リンクの遅延時間とする．評価にあたっては，サーバ 2，4，6，7，8 は $M_i = 800$，サーバ 1, 3, 5 は $M_i = 100$ とした場合と，全サーバを $M_i = 800$ とした場合の 2 つのケースを行う．定式化した最適化問題を線形計画法ソルバで解いた結果から得られた各ユーザ端末の選択サーバを図 3.7 に示す．

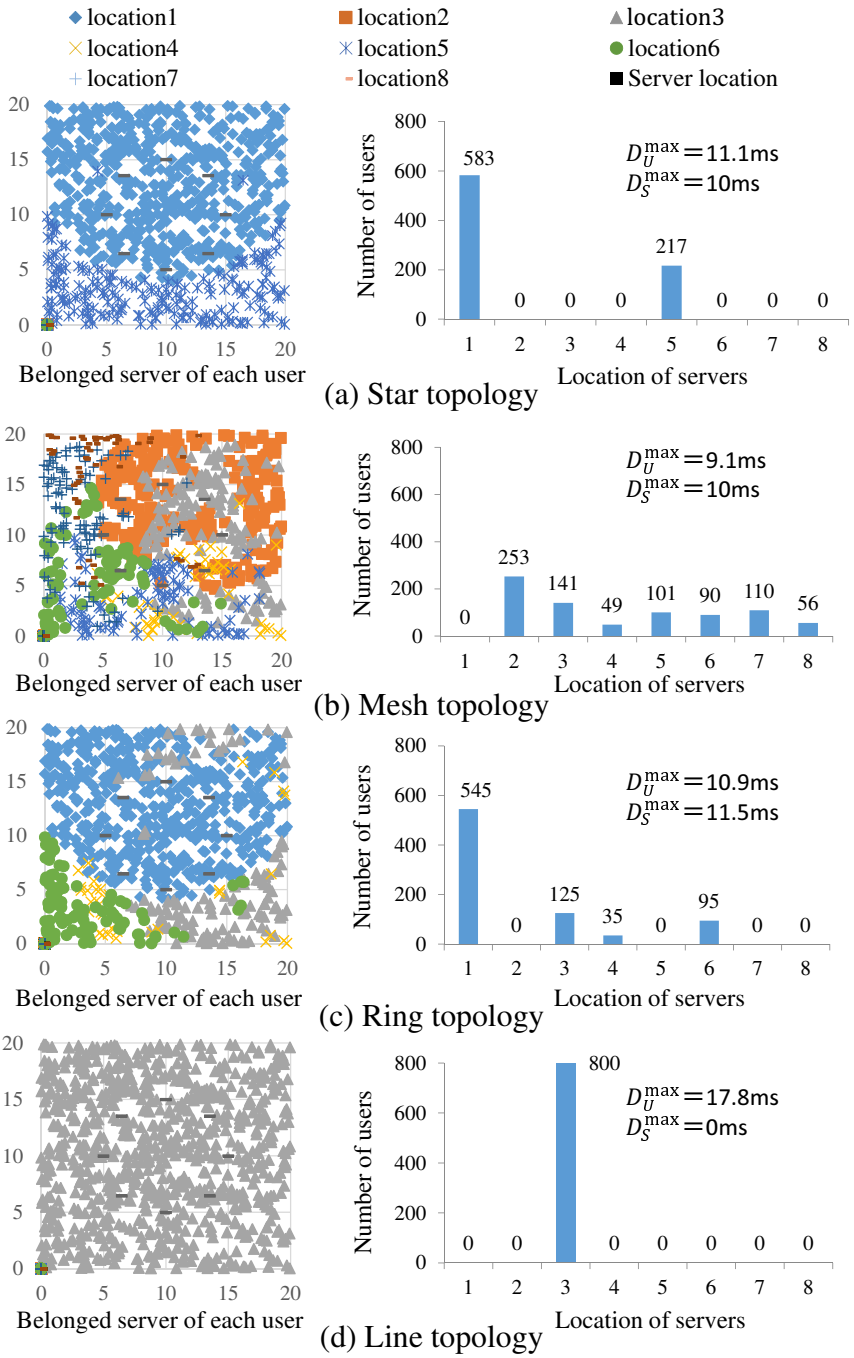


図 3.7 一様分布によりプロットされたユーザ端末と選択されたサーバ，$M_i = 800$, i=1, 2, 3, 4, 5, 6, 7, 8

スター型トポロジを用いて，具体的なサーバ選択例を図 3.8 に示す．目的関数である $2D_U^{\max} + D_S^{\max}$ の決定において，ユーザ端末 a は，$D_U^{\max}$ を決定づけるユーザ端末であり，サーバ 1 が選択されている．ユーザ端末 a が選択しているサーバ 1 とサーバ 5 間のリンク (d_{ij}=10ms) が $D_S^{\max}$ を決定しているリンクと仮定する．ユーザ端末 b は，d_{pi} の値としてはサーバ 4 が最も小さく，ユーザ端末 c はサーバ 6 が最も d_{pi} が小さいサーバだが，ユーザ端末 b がサーバ 4 を選択し，ユーザ端末 c がサーバ 6 を選択した場合，図 3.3 のとおり d_{ij}=$10+\sqrt{2+\sqrt{2}}$(ms) となり $D_S^{\max}$ の増加になるため，d_{pi} が最小ではないサーバ 5 を選択することで，目的関数が最小化される．また，ユーザ端末 d については，サーバ 1 を選択してもサーバ 5 を選択しても d_{pi} が $D_U^{\max}$ 以下のため，どちらのサーバを選択するかは，計算に用いるソルバのアルゴリズムに依存した結果となる．

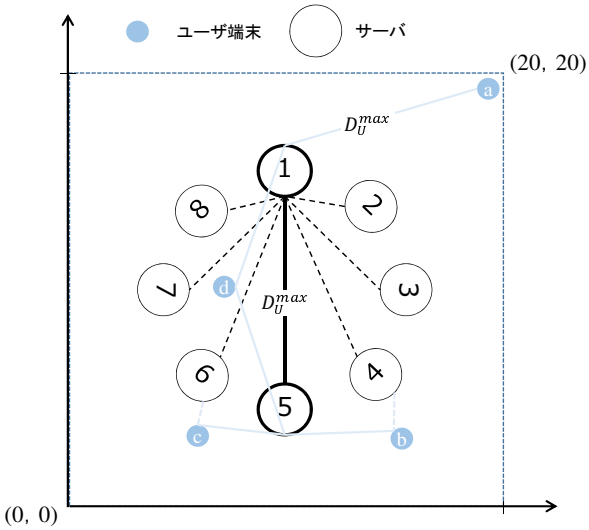


図 3.8　スター型トポロジでのサーバ選択例

図 3.7 (a) では，前述の例と同様にサーバ間通信の遅延時間が最小になるサーバ 1 とサーバ 5 のみが選択されている．サーバ 1，5 以外のサーバが選択されていない理由は，d_{pi} が小さく直線距離で最も近いサーバを選択しても，新たなサーバを選択することで $d_{ij}^{\max}$ が増加し，結果として目的関数である $2D_U^{\max} + DS^{\max}$ が増加するためである．また，サーバ 1 とサーバ 5 を選択したユーザ端末数が異なるのは，目的関数である $2D_U^{\max} + DS^{\max}$ の決定に影響のないユーザ端末は，どのサーバを選択してもよいため，ソルバのアルゴリズムによる結果である．

メッシュ型トポロジの図 3.7 (b) では，$DS^{\max}$=10ms でありトポロジ上の最大値で，

どのサーバを選択しても $DS^{\max}$ の増加はないため，ユーザ端末は $d_{pi} \leq D_U^{\max}$ の範囲内で，ソルバのアルゴリズムによるサーバが選択される．サーバ 1 のユーザ端末収容数が 0 となっているのは，今回評価したユーザ分布では，サーバ 1 を選択しなければ $d_{pi} \leq D_U^{\max}$ とならないユーザ端末が存在しなかったためである．

リング型トポロジの図 3.7 (c) では，$D_S^{\max}$=11.5ms となっており，リングトポロジにおけるトポロジ上の最大値 (15.3ms，図 3.3) よりも小さい値となっている．サーバ 1 を起点として考えた場合，図 3.9 のリング型トポロジでのサーバ選択例において，サーバ 5 近傍のユーザ端末では，破線のように d_{pi} が最小であるサーバ 5 を選択し，$D_S^{\max}$=15.3ms となるよりも，d_{pi} は最小でないがサーバ 4 を選択し $D_S^{\max}$=11.5ms としたほうが，目的関数が最小化されるため，サーバ 5 の収容数が 0 となっている．同様な理由から，選択されたサーバは $D_S^{\max}$ が最大となる対角線に配置されたサーバペアを避けた選択となっている．また，サーバ毎のユーザ端末の収容数が異なるのは，ソルバのアルゴリズムによる選択結果である．

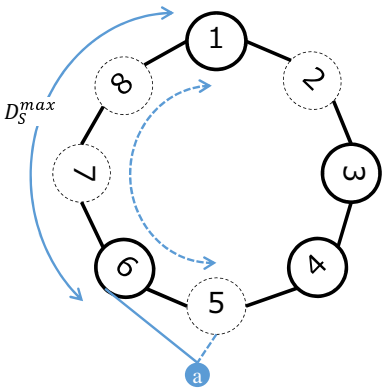


図 3.9 リング型トポロジでのサーバ選択例

ライン型トポロジの図 3.7 (d) では，サーバ 3 のみが選択され分散処理型の優位性がない結果となっている．選択されているサーバ 3 を起点として考えた場合，図 3.10 のライン型トポロジでのサーバ選択例において，サーバ 3 を起点として d_{ij} の最も大きいサーバ 5 近傍のユーザ端末 b は，サーバ 5 経由での接続では迂回ルートとなるためサーバ 3 への直接接続のほうが遅延時間が短くなる．本ケースのように，直接接続と比較しサーバ間が大きく迂回するルートで接続される場合は，サーバ間接続が選択されず，直接接続ルートが選択されることで，結果的に特定のサーバへの接続が集中する傾向になる．図 3.10 の例では，サーバ間が迂回ルートになるサーバ 4 とサーバ 5 近傍のユーザ端末が，直接接続を選択することになり，選択するサーバとの遅延時間 d_{pi} (D_p) が大きくなる傾向となる．

サーバ 7 近傍のユーザ端末 c においても，すでに D_p の値が大きなユーザ端末がいる状況であれば，サーバ 7 経由で接続し d_{ij} を加算するよりは，直接接続したほうが結果的に目的関数が最小化された結果となっている．

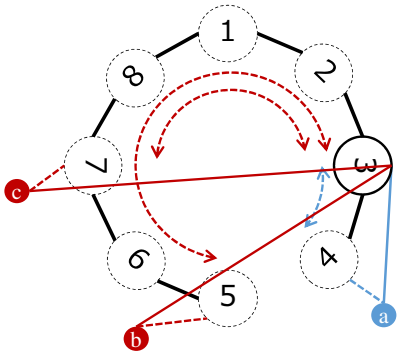


図 3.10　ライン型トポロジでのサーバ選択例

図 3.11 にサーバ 2，4，6，7，8 は $M_i = 800$，サーバ 1, 3, 5 は $M_i = 100$ とした場合の評価結果を示す．図 3.11 では，図 3.7 の条件に加えて，サーバ 1, 3, 5 を選択可能なユーザ端末数 M_i に制限があるため，他サーバに分散収容されていることがわかる．

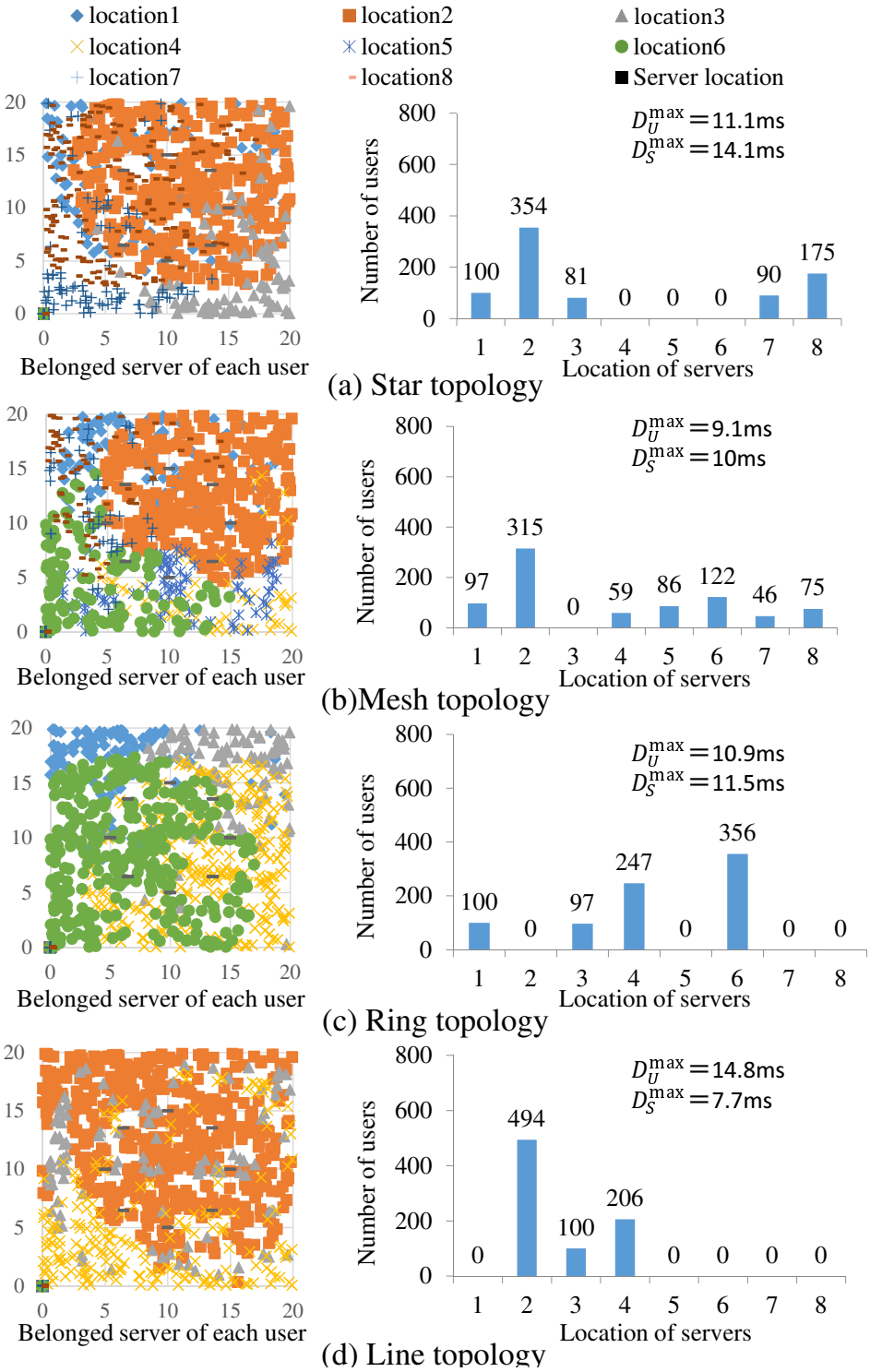


図 3.11 一様分布によりプロットされたユーザ端末と選択されたサーバ, $M_i = 100$, i=1, 3, 5 and $M_i = 800$, i=2, 3, 6, 7, 8

3.4 JPN48 モデルにおける効果測定

3.4.1 JPN48 モデルを用いた評価方法

実際のネットワークを擬似した評価として，サーバのネットワークトポロジとして，図 3.13 に示す JPN48 ノード [81] をサーバが配備されている拠点として評価する．全サーバが論理的にフルメッシュに接続されているとして，沖縄と札幌のように直接のリンクを持たないノード間は，隣接ノードを経由してトポロジ上の最短ルートで接続されているものとする．JPN48 ノード間の距離を図 3.12 に示す，

ユーザ端末とサーバ間トポロジは，図 3.13 に示すように，各 JPN48 ノードごとに 4 台のユーザ端末が遅延時間が 0.1ms，0.5ms，1ms，2ms のリンクで接続されていると仮定する．本評価では 1km を 5μs の遅延時間として評価を行っているため，ユーザ端末からサーバまでの遅延時間 0.1ms は 20km に相当し，2ms は 400km に相当する．

ユーザ端末は，図 3.13 に示すように，直接接続されたリンクを持たないノードに配備されたサーバとは，接続しているノードを経由してノード間リンクの最短距離で接続されるものと仮定する．本評価では，JPN48 ノードに配備されたサーバが 48 台，各ノードに 4 台のユーザ端末が接続しているためユーザ端末数は 192 (=48 $\times$ 4) 台となる．48 台のサーバがフルメッシュで接続されているためサーバ間リンク数は 1128 (= $\frac{48\times47}{2}$) となる．各ユーザ端末は接続しているノードに配備されたサーバとノードを経由して他ノードに配備された 48 台のサーバと接続されているため，ユーザ端末とサーバ間リンク数はリンク数は 9216 (=192 $\times$ 48) となる．

		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
札幌	1	476	410	593	537	654	672	839	836	942	915	966	945	993	974	810	1,064	1,124	1,200	1,079	1,021	1,303	1,125	1,272	1,339	1,358	1,348	1,387	1,424	1,390	1,464	1,602	1,724	1,568	1,729	1,862	1,529	1,604	1,795	1,686	2,010	2,063	2,164	2,110	1,962	2,169	2,280	2,922
青森	2		178	362	182	394	441	607	604	684	683	735	714	761	743	455	709	768	845	830	666	947	893	917	984	1,003	993	1,032	1,069	1,035	1,108	1,054	1,247	1,368	1,213	1,374	1,507	1,174	1,249	1,440	1,331	1,655	1,708	1,808	1,755	1,607	1,814	1,925	2,566
盛岡	3			184	127	245	263	429	426	532	505	556	535	583	564	400	654	714	791	669	612	893	716	862	929	949	939	978	1,015	862	828	902	1,054	1,192	1,314	1,158	1,319	1,452	1,120	967	1,042	1,233	1,124	1,448	1,501	1,602	1,754	1,700	1,759	1,870	2,512
仙台	4				273	61	79	245	242	349	322	373	352	399	381	248	502	562	638	486	459	741	532	710	777	796	786	825	862	828	902	1,040	1,162	1,006	1,167	1,300	967	1,042	1,233	1,124	1,448	1,501	1,602	1,548	1,400	1,607	1,718	2,360
秋田	5					212	352	518	515	502	577	643	607	598	635	273	527	587	663	648	484	765	771	735	802	821	811	850	887	853	926	1,065	1,186	1,031	1,192	1,325	992	1,067	1,258	1,149	1,473	1,526	1,627	1,573	1,425	1,632	1,743	2,385
山形	6						140	307	303	410	383	434	413	460	442	187	441	501	577	547	398	679	593	649	716	735	725	764	801	767	840	979	1,100	945	1,106	1,239	906	981	1,172	1,063	1,387	1,440	1,541	1,487	1,339	1,546	1,657	2,299
福島	7							259	163	270	243	309	273	320	302	180	434	494	570	407	387	668	453	638	705	728	718	757	794	760	834	972	1,094	938	1,099	1,232	899	974	1,165	1,056	1,380	1,433	1,534	1,480	1,332	1,539	1,650	2,292
水戸	8								96	192	117	128	147	195	176	421	502	561	638	281	309	544	328	513	580	651	661	700	737	669	776	914	1,036	880	1,042	1,174	842	916	1,108	999	1,322	1,376	1,476	1,422	1,274	1,481	1,593	2,234
宇都宮	9									107	79	145	110	157	138	335	417	476	553	553	244	224	505	290	475	541	612	622	661	698	631	737	876	997	842	1,003	1,136	803	878	1,069	960	1,284	1,337	1,438	1,384	1,236	1,443	1,554	2,196
前橋	10										75	141	105	96	133	229	310	370	446	183	117	399	284	368	435	506	516	555	592	524	631	769	891	735	896	1,029	697	771	963	853	1,177	1,231	1,331	1,277	1,129	1,336	1,448	2,089
大宮	11											66	30	78	59	304	385	444	521	164	192	427	211	396	463	534	543	544	583	620	552	659	797	919	763	925	1,057	725	799	991	882	1,205	1,259	1,359	1,305	1,157	1,364	1,476	2,117
千葉	12												39	87	68	370	451	510	587	173	258	436	219	405	472	543	353	592	629	561	668	806	928	772	933	1,066	734	808	1,000	890	1,214	1,268	1,368	1,314	1,166	1,373	1,485	2,126
東京	13													47	29	334	415	475	551	134	222	396	180	366	433	504	514	553	590	522	629	767	889	733	894	1,027	695	769	960	851	1,175	1,229	1,329	1,275	1,127	1,334	1,445	2,087
八王子	14														37	325	407	466	543	87	214	380	188	350	416	487	497	536	573	506	612	751	872	716	878	1,011	678	753	944	835	1,158	1,212	1,312	1,258	1,110	1,317	1,429	2,070
横浜	15															362	443	503	579	123	250	368	151	337	404	475	485	524	561	493	600	738	860	704	865	998	666	740	932	822	1,146	1,200	1,300	1,246	1,098	1,305	1,417	2,058
新潟	16																254	314	390	375	211	492	498	462	529	548	538	577	614	580	653	792	913	758	919	1,052	719	794	985	876	1,200	1,253	1,354	1,300	1,152	1,359	1,470	2,112
富山	17																	59	136	357	193	402	479	432	379	294	284	323	360	326	399	538	659	504	665	798	465	540	731	622	946	999	1,099	1,046	898	1,105	1,216	1,857
金沢	18																		77	416	252	342	539	372	319	235	225	264	301	267	340	478	600	444	605	738	406	480	672	562	886	940	1,040	986	838	1,045	1,157	1,798
福井	19																			493	329	265	482	296	243	158	148	187	224	190	263	402	523	367	529	662	329	404	595	486	809	863	963	909	761	968	1,080	1,721
甲府	20																				164	293	122	263	329	400	410	449	486	419	526	664	786	630	791	924	591	666	857	748	1,072	1,125	1,226	1,172	1,024	1,231	1,342	1,984
長野	21																					281	286	251	317	388	398	437	474	407	514	652	774	618	779	912	579	654	845	736	1,060	1,113	1,214	1,160	1,012	1,219	1,330	1,972
岐阜	22																						216	30	97	107	117	156	193	159	232	371	492	337	498	631	298	373	564	455	779	832	933	879	731	938	1,049	1,691
静岡	23																							186	252	323	333	372	409	342	449	587	709	553	714	847	514	589	780	671	995	1,048	1,149	1,095	947	1,154	1,265	1,907
名古屋	24																								67	138	148	187	224	156	263	401	523	367	528	661	329	403	594	485	809	863	963	909	761	968	1,079	1,721
津	25																									84	94	133	170	90	210	348	470	314	475	608	275	350	541	432	756	809	910	856	708	915	1,026	1,668
大津	26																										10	49	86	52	125	264	385	229	391	523	191	265	457	348	671	725	825	771	623	830	942	1,583
京都	27																											39	76	42	115	254	375	219	381	513	181	255	447	338	661	715	815	761	613	820	932	1,573
大阪	28																												37	52	76	293	414	180	342	474	142	216	408	299	622	676	776	722	574	781	893	1,534
神戸	29																													89	113	285	407	143	305	438	179	215	371	336	585	639	739	685	537	744	856	1,497
奈良	30																														128	295	417	232	394	526	194	268	460	351	674	728	828	774	626	833	945	1,586
和歌山	31																															354	476	212	373	506	66	140	335	223	654	708	808	649	501	708	820	1,493
鳥取	32																																122	142	303	378	288	214	369	373	526	580	680	644	536	743	815	1,438
松江	33																																	263	389	257	410	335	456	495	404	458	558	523	603	810	693	1,316
岡山	34																																		161	294	146	72	228	231	442	496	596	542	394	601	713	1,354
広島	35																																			133	308	233	66	317	281	334	435	381	233	440	551	1,193
山口	36																																				440	366	199	450	148	202	302	266	346	553	437	1,060
徳島	37																																					75	269	157	588	642	742	583	435	642	754	1,428
高松	38																																						194	159	514	567	668	509	361	568	679	1,353
松山	39																																							251	347	401	501	315	167	374	485	1,159
高知	40																																								598	652	752	566	418	625	736	1,410
博多	41																																									54	154	118	199	406	289	912
佐賀	42																																										100	172	252	459	343	858
長崎	43																																											272	352	559	443	758
熊本	44																																												148	296	171	844
大分	45																																													207	319	992
宮崎	46																																														126	800
鹿児島	47																																															674	
那覇	48																																																

図 3.12　JPN48 ノード間距離 (km)

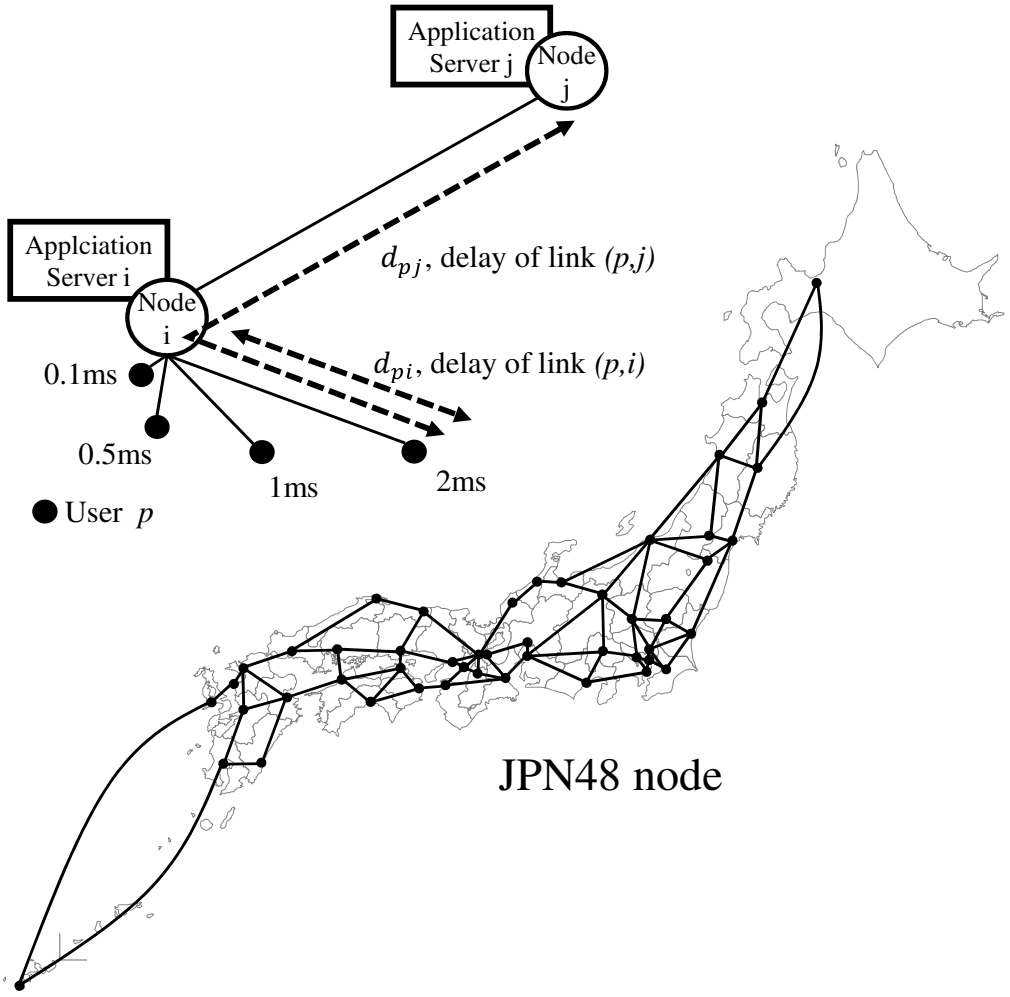


図 3.13 JPN48 ノードおよびユーザ端末とサーバ間トポロジ

JPN48 モデルにおける評価のパラメータを表 3.2 に示す.

表 3.2 JPN48 モデルにおける評価のパラメータ一覧

JPN48 モデルにおける評価
Y_{MAX}=48, M_i=192 (i=1〜48), 直接接続:d_{pi}=0.1ms，0.5ms，1ms，2ms, ノード経由接続:d_{pi}=0.1ms，0.5ms，1ms，2ms + (ノード間距離に比例した時間), u_{pi}=1, d_{ij}=JPN48 トポロジ上でのノード間最短経路に比例した時間

3.4.2 JPN48 モデルを用いた評価の評価結果

JPN48 モデルを用いた集中処理型と分散処理型の評価結果を表 3.3 に示す.

表 3.3 JPN48 model results

Scheme	Correction time	Results
Centralized (東京ノード)	$2D_U^{\mathrm{cen,max}}$	24.8
Centralized (和歌山ノード)	$2D_U^{\mathrm{cen,max}}$	18.94
Distributed	$2D_U^{\mathrm{dis,max}} + D_S^{\mathrm{dix,max}}$	18.6

分散処理型の評価では，全国のノードに配備する全てのサーバで M_i=192 として評価を行った．評価結果は，$D_U^{\mathrm{dis,max}}$=2.0，$D_S^{\mathrm{dis,max}}$=14.608 となり，ユーザ補正時間は $2D_U^{\mathrm{dis,max}} + D_S^{\mathrm{dis,max}} \simeq 18.6$ となる．

集中処理型の評価では，全国で 1 台のサーバで集中処理を行うため，集中処理サーバを M_i=192 として，それ以外のサーバは M_i=0 として評価を行った．集中処理サーバを東京ノードに配置する場合の評価結果は，$D_U^{\mathrm{cen,max}}$=12.434 となり，ユーザ補正時間は $2D_U^{\mathrm{cen,max}} \simeq 24.8$ となる．東京サーバの集中処理型と比較した分散処理型の遅延時間改善効果は約 25 %となる．また，集中処理サーバを遅延特性が最もよい和歌山ノードにサーバを配置する場合の評価結果は，$D_U^{\mathrm{cen,max}}$=9.47 となり，ユーザ補正時間は $2D_U^{\mathrm{cen,max}} \simeq 18.94$ となる．和歌山サーバの集中処理型と比較した分散処理型の遅延時間改善効果は約 2 %となる．

これらの結果から，JPN48 モデルにおいても集中処理型と比較し，提案方式は遅延特性に優れた特性であることわかる．加えて，全国に配備したサーバで分散処理を行うため，震災等によるビル被災時には，被災エリア以外ではサービスが継続できる等の信頼性面でのメリットも考えられる．

3.5 まとめ

提案方式の基本性能評価として，ネットワークトポロジとサーバ配備箇所による遅延特性の改善効果について評価を行った．サーバ間のネットワークトポロジとしては，スター型，フルメッシュ型，リング型，ライン型の 4 種類のトポロジを用いて，サーバの配備箇所による効果は，サーバ間の遅延時間を固定しユーザ端末とサーバ間の遅延時間を変化させることで，サーバの配備箇所による特性評価を行った．また，実際のネットワークにおける効果確認として，日本のネットワークトポロジを模擬した JPN48 ノードにサーバを

配備した場合の遅延特性の改善について評価を行った．

基本性能評価としては，フルメッシュ型やリング型のようにサーバ間で最短距離に近い距離のリンクを持っているトポロジが改善効果が高く，サーバ配備箇所としては，よりユーザに近いロケーションにサーバを配備すると改善効果が高いことが明らかとなった．分散処理型通信方式のサーバ選択問題の評価として，特定エリア内に一様分布した 200 台のユーザ端末について，定式化した最適化問題を解くことで，遅延時間を最小化するためのユーザ端末が選択するサーバが決定できることを示した．

実際のネットワークを擬似した評価として，JPN48 ノードをサーバが配備されている拠点として，1 台のサーバで集中処理する場合と，全 48 ノードのサーバで分散処理する場合を比較し，約 2-25 %の遅延時間の改善効果が確認された．

第 4 章

許容遅延時間の導入

4.1　許容遅延時間の導入とサーバ選択問題の定式化

リアルタイム型アプリケーションでは，エンド-エンドの 遅延時間によってはアプリケーションが成り立たないケースも考えられ，遅延許容時間を導入し許容遅延時間内でアプリケーションを利用可能とする．2.4.2 章の評価に加えて，アプリケーションの許容遅延時間として，ユーザ端末補正時間がとりうる値の最大値として D_{lim} を導入する．許容遅延時間を越えてサービス利用ができないユーザ端末数を N_{out} と表す．サーバ選択問題として，N_{out} を第一目的関数として，ユーザ端末補正時間を第二目的関数として最小化する最適化問題として定式化を行う．

q_p をユーザ端末 $p \in V_U$ に関するバイナリ変数として導入し，許容遅延時間を越えたユーザ端末を $q_p = 1$，許容遅延時間以内のユーザ端末を $q_p = 0$ で表す．遅延許容時間を超えてサービスが利用できないユーザ数 N_{out} を，q_p のユーザ端末 $p \in V_U$ に関する合計数とする．リンク $(p, i) \in E_U$ に関するバイナリ変数として z_{pi} を導入し，ユーザ端末補正時間がアプリケーションの許容遅延時間を満たすために，$(p, i) \in E_U$ が除外される場合は $z_{pi} = 0$ であり，除外されない場合は $z_{pi} = 1$ とする．$x_{pi} \cdot q_p = z_{pi}$ である．2.4.2 章で定式化した最適化問題に次式の条件を加えることで，遅延許容時間を考慮したサーバ選択問題として定式化する．

$$\text{Objective} \ \min \{N_{out} + \alpha(2D_U^{\max} + D_S^{\max})\} \tag{4.1a}$$

$$d_{pi}(u_{pi}x_{pi} - z_{pi}) \leq D_U^{\max} \quad ,\forall(p,i) \in E_U \tag{4.1b}$$

$$2D_U^{\max} + D_S^{\max} \leq D_{lim} \tag{4.1c}$$

$$N_{out} = \sum_{p \ inV_U} q_p \tag{4.1d}$$

$$z_{pi} \leq u_{pi}x_{pi}, \forall(p,i) \in E_U \tag{4.1e}$$

$$z_{pi} \leq q_p, \forall(p,i) \in E_U \tag{4.1f}$$

$$z_{pi} \leq u_{pi}x_{pi} + q_p - 1 \quad ,\forall(p,i) \in E_U \tag{4.1g}$$

$$z_{pi} \in \{0,1\}, \forall i \in E_U \tag{4.1h}$$

$$q_p \in \{0,1\}, \forall i \in V_U \tag{4.1i}$$

式 (2.1a) を，第一目的関数を遅延許容時間によって除外されたユーザ数 N_{out} とし，第二目的関数をユーザ端末補正時間 $2D_U^{\max} + D_S^{\max}$ として，これらの合計値を最小化する式 (4.1a) で置き換える．α は $2D_U^{\max} + D_S^{\max}$ が第二目的関数となるための十分小さな定数とする．式 (2.1e) は，$D_U^{\max}$ を許容時間以内のユーザ端末で最大化するため式 (4.1b) で置き換える．式 (4.1c) は，ユーザ補正時間 $2D_U^{\max} + D_S^{\max}$ が遅延許容時間 D_{lim} 以内であることを示す．式 (4.1d) は，N_{out} が q_p のユーザ $p \in V_U$ に関する合計であることを示す．式 (4.1e) - (4.1g) は，$x_{pi} \cdot q_p = z_{pi}$ を示す線形式を用いた表現である．式 (4.1h)，(4.1i) は，q_p，z_{pi} がそれぞれバイナリ変数であることを示す．

遅延許容時間を考慮したサーバ選択問題は，式 (4.1a)，式 (2.1b) - (2.1d)，式 (4.1b)，式 (2.1f)，式 (4.1c)，式 (4.1d)，式 (2.1g) - (2.1j)，式 (4.1e) - (4.1g)，式 (2.1k) - (2.1l)，式 (4.1h)，式 (4.1i) で定式化される最適化問題として扱うことができる．

表 4.1 に定式化した最適化問題のパラメータと決定変数を示す．

表 4.1　4 章で定式化した最適化問題のパラメータ，決定変数一覧

パラメータ	決定変数
Y_{MAX}, M_i, D_{lim}, d_{pi}, u_{pi}, d_{ij}	$D_U^{\max}$, $D_S^{\max}$, N_{out}, q_p, y_i, x_{pi}, z_{pi}

4.2　遅延許容時間を考慮したサーバ選択問題の評価

サーバのトポロジとしては，図 3.13 に示す JPN48 ノードをサーバが配備されている拠点とし，3.4.1 章と同じトポロジで評価を行う．定式化した式 (4.1a) において，α=0.0001 として評価を行う．遅延変動を考慮した評価のパラメータを表 4.2 に示す．

表 4.2　遅延許容時間を考慮した評価のパラメータ一覧

遅延許容時間を考慮した評価
Y_{MAX}=48，　D_{lim}=10-26ms，　M_i=192 (i=1〜48) 直接接続: d_{pi}=0.1ms，0.5ms，1ms，2ms， ノード経由接続: d_{pi}=0.1ms，0.5ms，1ms，2ms + (ノード間距離に比例した時間)， u_{pi}=1，　d_{ij}=JPN48 トポロジ上でのノード間最短経路に比例した時間

D_{lim} を変化させた場合の N_{out} とユーザ端末補正時間の比較を図 4.1 に示す．ユーザ端末補正時間は，集中処理型ではサーバ間通信がないため $2D_U^{\mathrm{cen,max}}$ となり，分散処理型では，$2D_U^{\mathrm{dis,max}} + D_S^{\mathrm{dis,max}}$ となる．

全ユーザのユーザ端末補正時間がアプリケーションの遅延許容時間よりも大きい場合は，ユーザ端末補正時間が小さくなるように，定式化された最適化問題の目的関数を決定付けているユーザ端末を除外し，ユーザ端末補正時間がアプリケーションの遅延許容時間以内となるまで遅延時間が大きいユーザ端末が除外される．これらのことから，集中処理型，分散処理型とも遅延許容時間が全ユーザのユーザ端末補正時間よりも小さい場合は，ユーザ端末補正時間≒遅延許容時間の関係となり図 4.1 においても，D_{lim} が 10ms から 20ms の間がそのような領域となっている．

遅延許容時間を越えて除外されるユーザ数は，同一のユーザ端末条件では，より小さいユーザ端末補正時間を実現できる方式のほうが，除外されるユーザ端末数が削減され，遅延時間がより大きいユーザ端末にも許容時間内でのエンド-エンド通信が可能となる．図 4.1 の D_{lim} が 10ms から 20ms の間では，集中処理型と比較し分散処理型のほう除外されるユーザ端末数 N_{out} が少なく，遅延特性に優れた方式であるといえる．

全ユーザのユーザ端末補正時間がアプリケーションの遅延許容時間よりも小さい場合は，遅延許容時間は定式化した最適化問題の目的関数の最小化に影響がなくなり，単純にユーザ端末補正時間での比較となる．図 4.1 において，分散処理型では D_{lim}=20ms を越えた領域からユーザ端末補正時間が一定となり，遅延許容時間がユーザ端末補正時間の決定に影響していないことを示している．集中処理型も除外されるユーザ数 N_{out}=0 となる D_{lim}=24ms を越えた領域から一定となり，分散処理型，集中処理型とも 3.4 章での評価

と同一条件となり，一定となるユーザ補正時間も表 3.3 の値で一定となっていることがわかる．

これらの結果から，本章の提案方式は遅延許容時間があるアプリケーションにおいて，より多くのユーザが利用可能で，かつ，遅延特性に優れた通信方式であるといえる．

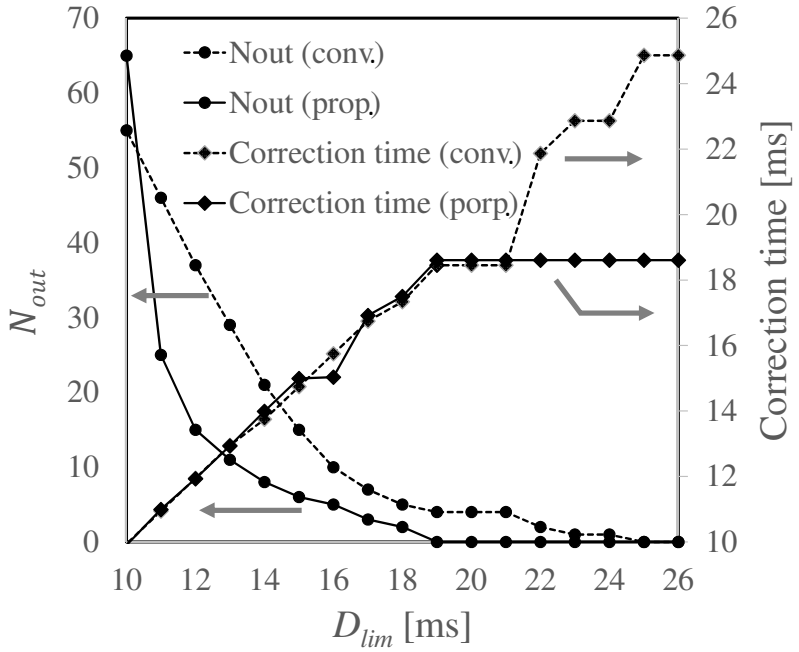


図 4.1　D_{lim} の増加に伴う N_{out} とユーザ端末補正時間の変化

4.3　まとめ

本章では，提案方式について遅延特性に厳しいリアルタイム型アプリケーションへの適用を考慮した拡張を行った．遅延許容時間と遅延許容時間を超えてアプリケーションが利用できないユーザ端末数を導入し，最適化問題の第一目的関数としてサービスが提供できないユーザ端末数，第二目的関数を補正時間として最適化問題を拡張した．

性能評価として，JPN48 ノードをサーバが配備されている拠点とし，遅延許容時間を変化させた場合の遅延許容時間を越えたユーザ端末数とユーザ端末補正時間の評価を行った．遅延許容時間について拡張を行った提案方式は，遅延許容時間を超えて利用できないユーザ端末数が集中処理型よりも少なく，集中処理型よりも少ない補正時間であることから，より多くのユーザが利用可能で，かつ，遅延特性に優れた通信方式であるといえる．

第 5 章

遅延変動率の導入

5.1　遅延変動率の導入とサーバ選択問題の定式化

4 章までの評価では，ユーザ端末とサーバ間の遅延時間が変動しない条件で評価を行ったが，実際のネットワーク環境では，トラヒック量によっては輻輳が発生し輻輳箇所を通過するトラヒックの遅延時間が変動する可能性がある．遅延変動を考慮したモデルとして，遅延変動率 f_{pi} を導入し，遅延変動を考慮した評価を行う．

定常時のリンク $(p,i) \in E_U$ の遅延時間を d_{pi} とした場合の遅延変動時のリンク (p,i) の最大遅延時間を遅延変動 f_{pi} を用いて $d_{pi}\times(1+f_{pi})$ と定義する．例えば，定常時の遅延時間が d_{pi}=1ms の場合に，遅延変動時の最大遅延が 2.5ms の場合は，f_{pi}=1.5 として，1ms×(1+1.5)=2.5ms となる．

4 章で定式化した最適化問題において，各式の d_{pi} を $d_{pi}(1+f_{pi})$ で置き換えることで，最大遅延時間での評価を行う．遅延時間の増加による影響を最小限にとどめるために，各ユーザ端末は最も遅延時間の少ないサーバを選択する．4 章で定式化した最適化問題では，サーバに近い場所に位置し目的関数に影響を及ぼさないユーザ端末は目的関数の最小化に影響のない範囲で，どこサーバを選択しても最適化問題の結果は変わらない．ユーザ端末が，最大遅延時間の最も小さいサーバを選択することで，遅延時間の増加時のユーザ端末補正時間の増加を抑止し，ユーザ端末とサーバ間で利用するリンク長を短くすることができる．

ユーザ端末 p とサーバ i 間リンク $(p,i) \in E_U$ の遅延時間 d_{pi} の全ユーザの総和 F_f と表し，最適化問題の第三の目的関数として最小化することで，4 章で定式化した最適化問題の条件を満足した上でユーザ端末を最も近いサーバを選択するサーバ選択問題として定

式化する．ユーザ端末とサーバ間リンクを最短なリンクで接続することで，有線リンクの場合は線路長の削減につながり，無線リンクの場合は安定した品質での接続が可能なネットワーク設計法として活用することができる．

4 章で定式化した最適化問題に次式の条件を加えることで，遅延変動率および遅延時間の総和を考慮した最適化問題として定式化する．

$$\text{Objective} \ \min \{N_{out} + \alpha(2D_U^{\max} + D_S^{\max}) + \beta F_f\} \tag{5.1a}$$

$$d_{pi}(1 + f_{pi})(u_{pi}x_{pi} - z_{pi}) \leq D_U^{\max} \quad ,\forall (p,i) \in E_U \tag{5.1b}$$

$$F_f = \sum_{(p,i)\in E_U} x_{pi} d_{pi}(1 + f_{pi}) \tag{5.1c}$$

式 (4.1a) は，第一目的関数を遅延許容時間によって除外されたユーザ数 N_{out} とし，第二目的関数をユーザ端末補正時間 $2D_U^{\max} + D_S^{\max}$ とし，第三目的関数を全ユーザのユーザとサーバ間リンクの遅延時間の総和 F_f として，式 (5.1a) で置き換える．α は $2D_U^{\max} + D_S^{\max}$ が第二目的関数となるための十分小さな定数であり，β は F_f が第三目的関数となるための十分小さな定数である．式 (4.1b) は，遅延変動を考慮して $D_U^{\max}$ を決定するため，式 (5.1b) で置き換える．式 (5.1c) は，遅延変動率 f_{pi} を考慮した条件でのユーザ端末とサーバ間リンクの遅延時間の総和が F_f であることを表す．

遅延許容時間と遅延変動率を考慮したサーバ選択問題は，式 (5.1a)，式 (2.1b) - (2.1d)，式 (5.1b)，式 (2.1f)，式 (4.1c)，式 (4.1d)，式 (5.1c)，式 (2.1g) - (2.1j)，式 (4.1e) - (4.1g)，式 (2.1k) - (2.1l)，式 (4.1h)，式 (4.1i) で定式化される最適化問題として扱うことができる．

表 5.1 に定式化した最適化問題のパラメータと決定変数を示す．

表 5.1　5 章で定式化した最適化問題のパラメータ，決定変数一覧

パラメータ	決定変数
Y_{MAX}, M_i, D_{lim}, d_{pi}, u_{pi}, d_{ij}	$D_U^{\max}$, $D_S^{\max}$, N_{out}, F_f, q_p, y_i, x_{pi}, z_{pi}

5.2　遅延変動を考慮したサーバ選択問題の評価

5.1 章で定式化した最適化問題を用いることで，遅延変動耐性が向上しユーザ端末とサーバ間のリンク長が最小化されていることを評価する．評価は，遅延変動率 f_{pi}

と遅延時間の総和 F_f を考慮したサーバ選択問題と遅延変動を考慮しないサーバ選択問題の 2 つの設計法で決定したサーバ選択の結果を比較する．比較は，遅延時間 d_{pi} が増加した場合の遅延許容時間を超えるユーザ数で行う．また，ユーザ端末が選択したサーバとのユーザ端末-サーバ間リンク長の評価については，遅延時間が伝送距離と比例している前提で，ユーザ端末とサーバ間リンクの遅延時間の総和を比較することで，より短いリンク長のサーバを選択しているかを評価する．

定式化した式 (5.1a) において，α=0.0001，β=0.0000001 として評価を行う．D_{lim}=2ms，全サーバで M_i=200 として評価を行う．サーバのトポロジとしては，関東エリアにサービス提供していると仮定し，図 5.1 に示す JPN48 関東エリアのノードをサーバが配備されている拠点とする．

サーバ間のネットワークは，全サーバが論理的にはメッシュに接続されているとして，図 5.1 の JPN モデルから直接のリンクを持たないサーバ間はノードを経由して最短ルートで接続されているものとする．ユーザ端末の分布は，図 5.1 に示すように，経度 139 度から 140.5 度，緯度 35.2 度から 36.8 度のエリア内に 200 台のユーザ端末が一様分布していると仮定する．ユーザ端末は全ノードと直接接続されているリンクが存在し，ユーザ端末と各ノード間は，図 5.1 の座標上の直線距離で接続されているものとする．ユーザ端末とサーバ間リンクの遅延時間は，ユーザの緯度・経度情報と各ノードの緯度・経度情報から地理上の直線距離を求めるべきであるが，本評価では簡単化のため座標軸の 1 目盛を 1ms の遅延時間として，座標軸上の 2 地点間の距離を遅延時間とする．ユーザ端末の分布としては，X 軸の経度 139 度から 140.5 度が 1.5ms で 300km 相当，Y 軸の緯度 35.2 度から 36.8 度が 1.6ms で 320km 相当のエリアに 200 台が一様分布していると仮定される．

本評価では，JPN48 関東エリアのノードに配備されたサーバが 8 台，ユーザ端末数は 200 台となる．また，8 台のサーバがフルメッシュで接続されているためサーバ間リンク数は 28 となり，各ユーザ端末は全ノードと直接リンクを持っているため，ユーザ端末とサーバ間リンク数は 1600 となる．遅延変動を考慮した評価のパラメータを表 5.2 に示す．

表 5.2 遅延変動を考慮した評価のパラメータ一覧

遅延変動を考慮した評価
Y_{MAX}=8， D_{lim}=2ms， M_i=200 (i=1〜200)，
d_{pi}=ユーザ端末とサーバの座標上の直線距離に比例した時間， u_{pi}=1，
f_{pi}=1.5 (大宮ノードと横浜ノードを経由するリンク)，
f_{pi}=0 (大宮，横浜を経由しないリンク)
d_{ij}=JPN48 トポロジ上でのノード間最短経路に比例した時間

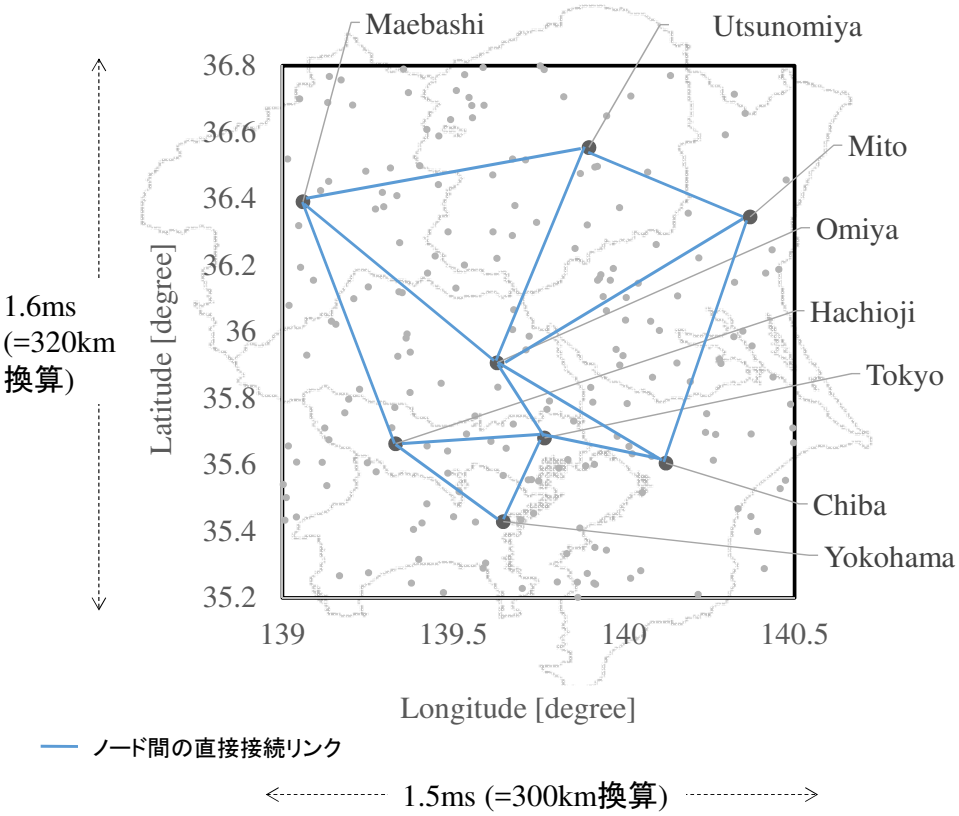
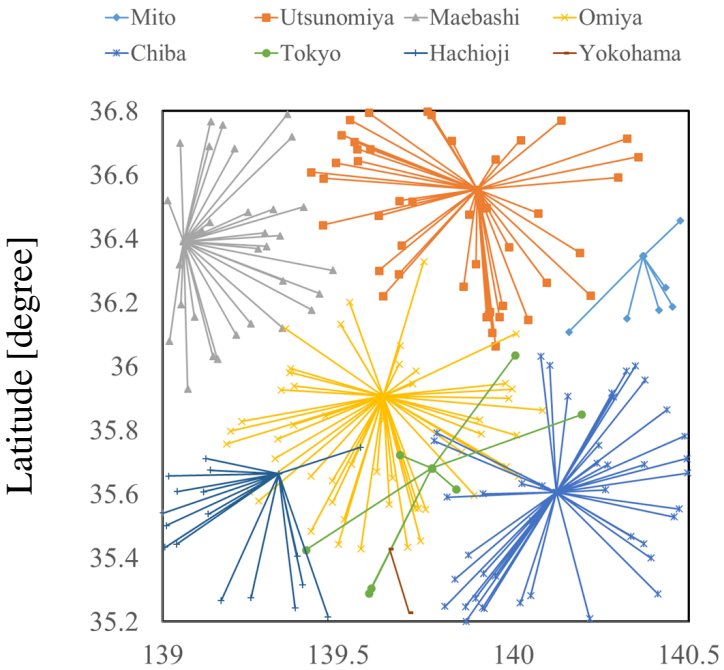
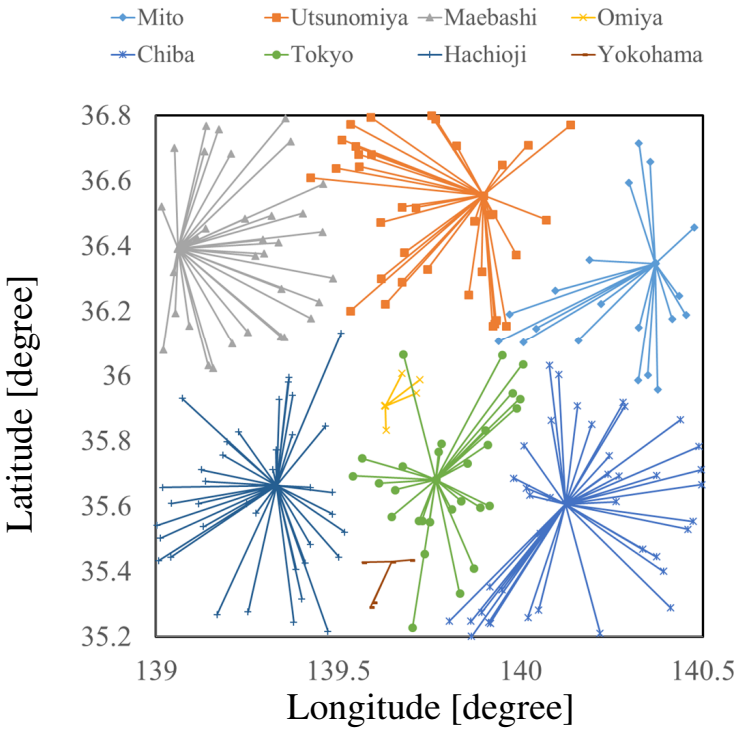


図 5.1　JPN48 ノード関東エリア上でのユーザ端末の分布

JPN48 関東エリアの 8 つのノードのうち大宮ノードと横浜ノードが輻輳していると仮定する．図 5.2 (a) に，4.1 章で定式化した遅延変動を考慮しない解法で決定したユーザ端末が選択するサーバについて，ユーザ端末と選択されたサーバを線で結んだ図を示す．図 5.2 (b) に，大宮ノードと横浜ノードを経由するリンクを f_{pi}=1.5 として，5.1 章で定式化した遅延変動を考慮した解法で決定した結果を示す．



(a) Servers accommodating users
determined by optimization problem at section 4.1



(b) Servers accommodating users
determined by optimization problem at section 5.1

図 5.2　4.1 章で定式化したサーバ選択法と 5.1 章で定式化した遅延変動を考慮したサーバ選択法によって決定されたユーザ端末の選択するサーバ

図 5.2 (a) では，非常に近い場所に位置する複数のユーザ端末が，異なるサーバを選択している．特に，ユーザ端末とサーバを接続する線が交差する箇所もあり，3.3 章での評価どおり目的関数を決定づけないユーザ端末については，目的関数の値が増加しない範囲で，どのサーバが選択されてもよいため，用いたソルバのアルゴリズムによって選択されるサーバが決定しているためである．また，横浜ノードのユーザ端末数が非常に少なくなっているのも同様な理由と想定される．

図 5.2 (b) では，第三目的関数として，遅延時間の総和 F_f を最小化しているため，ソルバのアルゴリズムによって決定していた選択サーバが，最大遅延時間が最も小さいサーバが選択され，図 5.2 (a) でのユーザ端末とサーバを接続する線が交差する箇所が，ほぼなくなっているのがわかる．また，大宮サーバと横浜サーバを選択しているユーザ端末がサーバ近傍に限られて少なくなっているのは，ユーザ端末とサーバ間リンクの遅延時間が遅延変動時の最大遅延時間で評価されているためである．

ユーザ端末が選択するサーバを決定した時点から実際の遅延時間が増加した場合の遅延特性への影響評価として，図 5.2 の 2 つの結果について，ユーザ端末とサーバ間の遅延時間が増加した場合に，選択サーバを決定した際の遅延許容時間を超えるユーザ端末数を比較する．図 5.3 に，大宮ノードと横浜ノードを選択しているユーザ端末の遅延時間の増加に伴い，遅延許容時間 D_{lim}=2ms を超えるユーザ数の変化を示す．

式 (5.1a) で設計した結果では，設計時の遅延変動率 f_{pi}=1.5 を超えて，x 軸が 2.0 の d_{pi} が 2 倍増加する場合でも，選択サーバを決定した際の遅延許容時間 D_{lim}=2ms を超えるユーザ端末数である y 軸が 0 となっている．x 軸が 2.0 を超えても，式 (4.1a) で設計した結果と比較し，遅延許容値 D_{lim}=2ms を超えるユーザ端末数が少なく抑えられている．

遅延時間の増加に伴い，サーバから遠いユーザ端末から D_{lim} を超えることになるが，同一の遅延時間でも図 5.2(b) の結果のほうが，最大遅延時間を考慮したサーバの選択を行っているため，D_{lim} を超えるユーザ端末数が少なく，遅延時間が増加した場合の影響が小さいことがわかる．

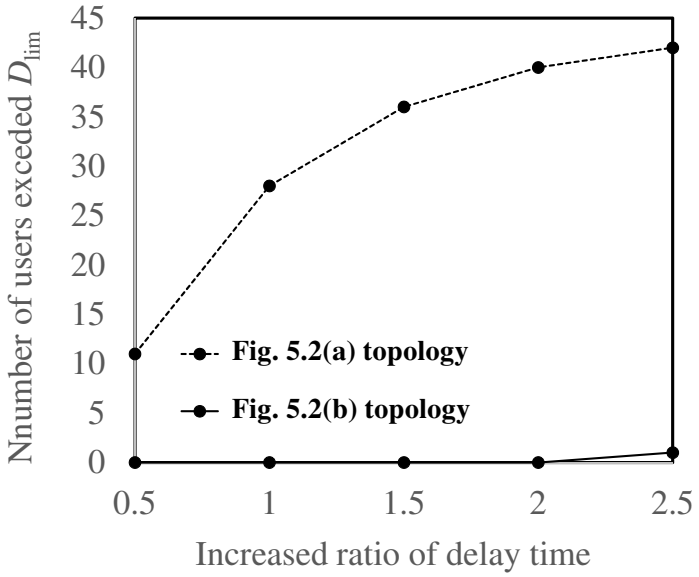


図 5.3　横浜と大宮ノードを経由するリンクの遅延増加に伴う $D_{\lim}$ を超えるユーザ端末数

図 5.4 に，図 5.2 (a) と (b) のユーザ端末の選択したサーバ群におけるユーザ端末とサーバ間遅延時間の総和の比較を示す．図 5.4 では，図 5.2(a) の F_f の値を 1.0 とした場合の比率で比較を行う．図 5.4 に示すように，最適化問題に第三目的関数として F_f を導入し最小化することで，F_f の値が約 20 %削減されており，ユーザ端末とサーバ間のリンク長を最短で接続するサーバの選択が可能となる．

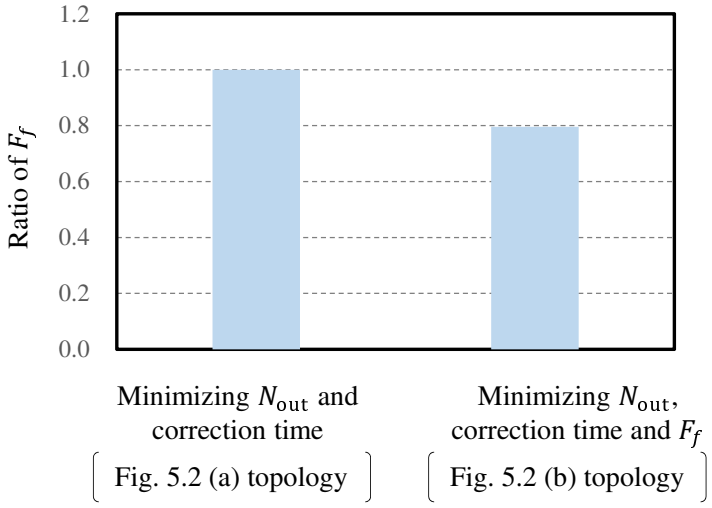


図 5.4　4.1 章で定式化したサーバ選択問題と 5.1 章で定式化した遅延変動を考慮したサーバ選択問題による F_f の比率

5.3　まとめ

本章では，提案方式についてネットワーク輻輳時の遅延変動を考慮した拡張を行った．遅延変動率を導入し遅延変動時の最大遅延時間を遅延時間として扱うとともに，第三の目的関数としてユーザ端末とサーバ間遅延時間の総和を最小化する拡張を行った．これらの拡張により，下記の優先順位で選択するサーバを決定するサーバ選択問題として定式化を行った．

1. 許容遅延時間を超えるユーザ端末を最小化
2. 遅延時間を最小化するためにユーザ端末補正時間を最小化
3. 遅延変動時の最大遅延時間が最も小さいサーバを選択

性能評価として，JPN48 関東エリアのノードをサーバが配備されている拠点として，一様分布した 200 台のユーザ端末の選択サーバを本章で定式化したサーバ選択問題で決定した結果と，前章で定式化したサーバ選択問題で決定した結果の比較評価を行った．遅延変動について拡張を行った提案方式は，ユーザ端末の遅延時間を増加させた場合に，許容遅延時間を越えるユーザ端末数が少なく，輻輳等で遅延が増加した場合の影響が少ないことが確認された．

第 6 章

逐次参加型のユーザ端末参加方法

6.1　逐次参加型のユーザ端末参加方法の定式化

ネットゲームや電話会議などユーザ端末数が時間的に変化する逐次参加型アプリケーションへの適用を考慮する．逐次参加型アプリケーションでは，新規ユーザ端末が参加するたびに，利用中のユーザ端末が異なるサーバへ選択先が切り替わると，利用の中断等発生する可能性があり，利用中ユーザ端末へ影響なく参加ユーザが最適なサーバを選択することが必要となる．上記の条件を満たす逐次参加型のユーザ端末参加方法として，アプリケーション利用中のユーザ端末は，既に利用しているサーバにを選択する制約を与え，モデル化を行う．

2.4.2 章の条件に加えて，利用中ユーザ端末を表すノードの集合を V_U^1 とし，参加する新規ユーザ端末の集合を V_U^2 と表す．利用中ユーザ端末が選択しているサーバの集合を V_S^1 とし，利用中ユーザ端末が選択するサーバの集合を V_S^2 と表す．利用中ユーザ端末とサーバ間リンクの集合を E_U^1 とし，参加する新規ユーザ端末とサーバ間リンクの集合を E_U^2 と表す．利用中でかつ新規参加のユーザ端末は存在しないことから，$V_U^1 \cup V_U^2 = V_U$，$E_U^1 \cup E_U^2 = E_U$，$V_U^1 \cap V_U^2 = \emptyset$，$E_U^1 \cap E_U^2 = \emptyset$ となる．2.4.2 章の定式化において，x_{kl} をリンク $(k,l) \in E$ に関する利用中を示すバイナリ変数，y_i をサーバ $i \in V_S$ に関する利用中を示すバイナリ変数として導入した．逐次参加型では，利用中のユーザ端末は，既に利用しているサーバを選択するため，ユーザ端末 $p \in V_U^1$ とサーバ $i \in V_S$ 間リンク $(p,i) \in E_U^1$ に対しては，x_{pi} が決定している値として x_{pi}^{given} と表す．利用中のユーザ端末が選択しているサーバ $i \in V_S^1$ は，y_i を決定している値として y_i^{given} と表す．$i \in V_S^2$ に対しては，y_i^{given} の値が与えられていないが，$y_i^{given} = 0$ と扱う．2.4.2 章の最適化問

題に次式の条件を加えることで，利用中ユーザ端末が既に利用しているサーバを選択する制約条件が課された最適化問題として定式化する．

$$\sum_{i \in V_S} u_{pi} x_{pi} = 1, \forall p \in V_U^2 \tag{6.1a}$$

$$x_{ij} \geq x_{ij}^{given}, \forall (i,j) \in E_U^1 \tag{6.1b}$$

$$y_i \geq y_i^{given}, \forall i \in V_S \tag{6.1c}$$

$$\tag{6.1d}$$

式 (2.1b) は，選択するサーバが決定していないユーザ端末を対象とするため，式 (6.1a) に置き換える．利用中ユーザ端末とサーバ間リンクに関する変数 x_{ij} を決定している値 x_{ij}^{given} で置き換えるため式 (6.1b) の制約条件を与える．利用中のユーザ端末が選択しているサーバに関しては，y_i が決定している値として y_i^{given} で置き換えるため式 (6.1b) を加える．

逐次参加型のユーザ端末参加方法は，式 (2.1a)，式 (6.1a)，式 (2.1c)- (2.1l)，式 (6.1b)，式 (6.1c) で定式化される最適化問題として扱うことができる．また，逐次参加型のユーザ端末参加方法は，以下の Step 1 - Step 4 の処理で表すことができる．

- Step 1
  - 初期化, $V_U = 0$, $E_U = 0$
- Step 2
  - 新規ユーザ端末の集合 V_U^2 の参加．V_U^2 から E_U^2 を生成．
  - $V_U^2 = \emptyset$ であれば終了，$V_U^2 \neq \emptyset$ であれば Step3 へ進む．
- Step 3
  - 式 (2.1a)，式 (6.1a)，式 (2.1c)- (2.1l)，式 (6.1b)，式 (6.1c) で定式化した最適化問題を解く．
- Step 4
  - 最適化問題の解として求められた (k,l) の値を x_{kl}^{given} に代入する．
  - 最適化問題の解として求められた y_i の値を y_i^{given} に代入する．
  - $V_U^1 \leftarrow V_U^1 \cup V_U^2$
  - $E_U^1 \leftarrow E_U^1 \cup E_U^2$
  - Step 2 に戻る．

表 6.1 に定式化した最適化問題のパラメータと決定変数を示す．

表 6.1 6.1 章で定式化した最適化問題のパラメータ，決定変数一覧

パラメータ	決定変数
Y_{MAX}, M_i, y_i^{given}, d_{pi}, u_{pi}, d_{ij}	$D_U^{\max}$, $D_S^{\max}$, y_i, x_{pi}

6.2 ユーザ参加時間の短縮と定式化

ユーザ端末数が時間的に変化する逐次参加型アプリケーションでは，新規ユーザ参加時の選択サーバを決定する計算時間は，参加時の待ち時間となるためサービス性の観点からは迅速な選択サーバの決定が求められる．6.1 章の条件を踏まえて，新規ユーザ端末の参加待ち時間を短縮化するユーザ端末の参加方法を導入する．

利用中ユーザ端末については，既にユーザ端末補正時間である $2D_U^{\max} + D_S^{\max}$ とサーバ i を選択しているユーザ端末数 M_i が決定しているため，6.1 章で定式化した最適化問題において，ユーザ端末を新規参加ユーザ端末に関連する $p \in V_U^2$，$(p,i) \in E_U$ 2 に関するパラメータのみで計算できるように拡張することで，利用中ユーザ端末数に依存しない高速な計算を実現する．

利用中のユーザ端末から決定されている $D_U^{\max}$ を $D_U^{\max,\mathrm{given}}$ と表し，サーバ i のユーザ端末収容数 $\sum_{p \in V_U^1} u_{pi}x_{pi}$ を M_i^{given} と表す．6.1 章の最適化問題に次式の条件を加えることで，利用中ユーザ端末が既に利用しているサーバを選択する制約条件が課された最適化問題として定式化する．

$$\sum_{p \in V_U \ 2} u_{pi}x_{pi} \leq (M_i - M_i^{given}), \forall i \in V_S \tag{6.2a}$$

$$u_{pi}x_{pi}d_{pi} \leq D_U^{\max}, \forall (p,i) \in E_U^2 \tag{6.2b}$$

$$D_U^{\max,\mathrm{given}} \leq D_U^{\max} \tag{6.2c}$$

式 (2.1c) は，サーバ i が受け入れられる最大の新規ユーザ端末について，利用中ユーザ端末分を差し引いた値で計算を行うため式 (6.2a) で置き換える．式 (2.1e) は，新規ユーザ端末のみで計算ができるように，式 (6.2b) と式 (6.2c) で置き換える．

ユーザ参加時間の短縮化を導入した逐次参加型のユーザ端末参加方法は，式 (2.1a)，式 (6.1a)-(6.1c)，式 (6.2a)，式 (2.1d)，式 (6.2b)，式 (6.2c)，式 (2.1f)- (2.1l) で定式化される最適化問題として扱うことができる．

表 6.2 に定式化した最適化問題のパラメータと決定変数を示す．

表 6.2　6.2 章で定式化した最適化問題のパラメータ，決定変数一覧

パラメータ	決定変数
Y_{MAX}, M_i, M_i^{given}, $D_U^{\mathrm{max.given}}$, y_i^{given}, d_{pi}, u_{pi}, x_{pi}^{given}, d_{pi}, u_{pi}, d_{ij}	$D_U^{\max}$, $D_S^{\max}$, y_i, x_{pi}

6.3　逐次参加型ユーザ端末参加方法の遅延特性の評価

逐次参加型のユーザ端末補正時間は，一斉参加型と比較して，利用中のユーザ端末はサーバの選択先を変えないため遅延特性の改善性能が劣化する可能性があり，遅延特性について評価を行う．

5.2 章の評価と同様に，サーバは JPN48 関東ノードに配備されており，図 5.1 に示すエリア内に 200 台のユーザ端末が一様分布しているモデルで，遅延許容時間 D_{lim}=2ms，サーバのユーザ端末収容数は全サーバで M_i=200 として評価を行う．

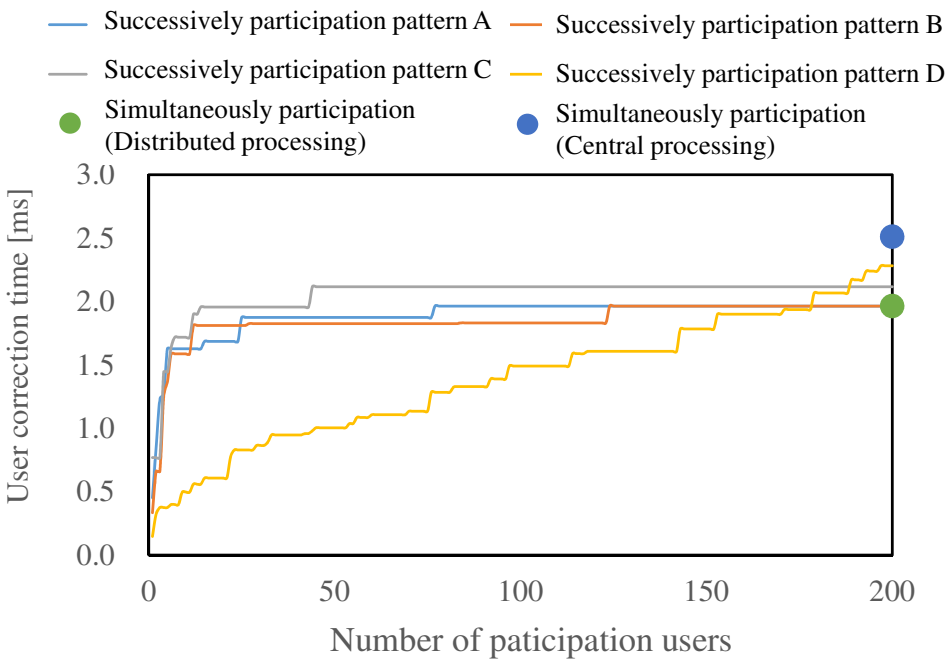
遅延特性の性能評価結果として，図 6.1 (a) に，以下の 6 種類の参加パターンにおけるユーザ端末補正時間を示す．逐次参加型は． 6.2 章で定式化された参加方法で評価を行う．表 6.4 に，以下の 6 種類の参加パターンにおける 200 台目のユーザ端末参加時の遅延特性比 R の値を示す．逐次参加型では，1 台毎に参加した際のユーザ端末補正時間の変化を示す．図 6.1(b) は，各参加パターンにおけるサーバごとのユーザ収容数である．

逐次参加型ユーザ端末参加方法の評価におけるパラメータを表 6.3 に示す．

表 6.3　逐次参加型ユーザ端末参加方法の評価におけパラメーター一覧

逐次参加型ユーザ端末参加方法の評価
Y_{MAX}=8，　M_i=200(i=1〜200)， d_{pi}=ユーザ端末とサーバの座標上の直線距離に比例した時間，　u_{pi}=1 d_{ij}=JPN48 トポロジ上でのノード間最短経路に比例した時間

- 逐次参加型，ランダム参加 (Successively participation pattern A, B)
- 逐次参加型，遅延の大きい順に参加 (Successively participation pattern C)
- 逐次参加型，遅延の小さい順に参加 (Successively participation pattern D)
- 200 台一斉参加，分散処理型 (Simultaneously participation, Distributed processing)
- 200 台一斉参加，集中処理型 (Simultaneously participation, Central processing)



(a) Correction time dependence on number of participation users

(b) Number of users belonged in each location servers

図 6.1　ユーザ参加モデルによるユーザ端末補正時間とサーバごとのユーザ収容数

表 6.4　ユーザ参加モデルによる 200 台目のユーザ端末参加時の R の値

ユーザ参加モデル	R (遅延特性比)
逐次参加型，ランダム参加 (A)	0.78
逐次参加型，ランダム参加 (B)	0.78
逐次参加型，遅延の大きい順に参加 (C)	0.84
逐次参加型，遅延の小さい順に参加 (D)	0.91
一斉参加型	0.78

図 6.1 (a) および表 6.4 のとおり，遅延特性比は一斉参加型の $R = 0.78$ に対して，逐次参加型は，同一のユーザ端末でも参加順序によって $R = 0.78 - 0.91$ と悪化するケースがある．特に，ユーザ端末-サーバ間リンクの遅延時間が小さいユーザ端末から参加した場合 (pattern D) は，遅延時間の最も大きいユーザ端末が利用中の 199 台のユーザ端末の選択サーバが決定している制約の多い条件で，収容サーバが決定されるためである．逐次参加型のいずれのパターンにおいても，ユーザ端末補正時間は集中処理型よりも低い値となっており，分散処理型通信方式の有効性が確認された．

6.4　逐次参加型ユーザ端末参加方法の計算時間短縮化の評価

逐次参加型ユーザ端末参加方法と計算時間短縮化を導入した 6.2 章で定式化したユーザ参加方法を用いた選択サーバを決定する計算時間の短縮効果を評価する．5.2 章の評価と同様に，図 5.1 に示すエリア内に 200 台のユーザ端末が一様分布しているモデルで，遅延許容時間 D_{lim}=2ms，サーバのユーザ端末収容数は全サーバで M_i=200 として評価を行う．逐次参加型では，199 台の利用中ユーザ端末が存在する状態に，200 番目のユーザ端末が参加した際の選択サーバを決定する計算時間を比較する．

図 6.2 に，200 台のユーザ端末が一斉に参加した際の集中処理型での処理時間を 1 とした場合の以下の方式における 200 台目のユーザ端末参加時の計算時間の比率を示す．最適化計算を実施した計算機のハードウェア条件は，Intel (R) Xeon (R) CPU E5-2609 2.5GHz 8 コア，64GB メモリ，使用した整数計画ソルバーは，CPLEX [78] を用いて，処理時間は計算機のシステム時刻より算出した．最適化計算は同一の条件で 5 回実施し，以下の各方式の記述における (　) 内が CPU のシステムクロック時間 (tick) と実行時間である．実行時間が短いため，システムクロック時間で処理時間の比率を計算した．

- 200 台一斉参加，集中処理型
  - (2.12ticks×5 回，0.00sec×3 回，0.01sec×2 回)
- 200 台一斉参加，分散処理型
  - (261.2ticks×5 回，0.35sec×3 回，0.37sec，0.43sec)
- 逐次参加型 200 台目参加，6.1 章で定式化された参加方法
  - (0.79ticks×5 回，0.00sec×5 回)
- 逐次参加型 200 台目参加，ユーザ参加時間を短縮化した 6.2 章で定式化された参加方法
  - (0.13ticks×5 回，0.00sec×5 回)

一斉参加型と比較し，逐次参加型は利用中ユーザ端末 199 台は選択するサーバが決定しているため，選択するサーバを計算をするユーザ端末数の量が削減され，1/100 以下まで計算時間が短くなっている．一斉参加型においては，集中処理型は選択するサーバが決定していることから，選択サーバを決定する計算量が削減され計算時間が，分散処理型と比較し 1/100 以下まで短くなっている．逐次参加型においては，ユーザ参加時間を短縮化した 6.2 章で定式化したユーザ参加方法の値が，6.1 章で定式化した参加方法の値と比較し，70 %以上の削減が実現されている．

今回の評価結果から，ユーザ参加時間の短縮化を導入した参加方法の適用により，選択サーバを決定するための計算処理の大幅な削減が行われ，新規ユーザ追加時の待ち時間の削減が期待できると考えられる．

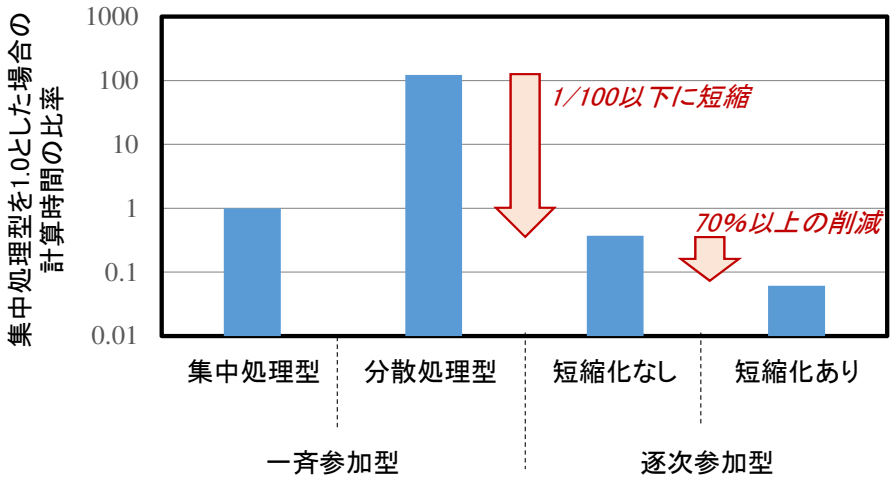


図 6.2　一括参加型と逐次参加型の計算時間の比較

6.5　まとめ

本章では，提案方式についてユーザ端末が時間経過とともに適宜追加されるアプリケーションへの適用を考慮した逐次参加型のユーザ参加方法について拡張を行った．逐次参加型のユーザ参加方法では，変数として扱っていた利用有無を表すパラメータを利用中ユーザ端末については，決定値として扱うことで，選択サーバを変更しないユーザ参加方法について，サーバ選択問題の定式化を行った．

新規ユーザ参加時の待ち時間を短縮するユーザ端末参加方法として，利用中ユーザ端末から決定している値を用いて，最適化問題の対象範囲を新規ユーザ端末に限定することで，利用中ユーザ端末数に依存しないユーザ参加方法についてサーバ選択問題の定式化を行った．

逐次参加型ユーザ端末参加方法の遅延特性について，一斉参加型との比較評価を行い，同一のユーザ端末条件でも参加する順序によりユーザ端末補正時間は変わるものの，逐次参加型のいずれのパターンにおいても，ユーザ端末補正時間は集中処理型よりも低い値となっており，逐次参加型のユーザ参加方法について拡張を行った提案方式の有効性が確認された．

逐次参加型に加えて，ユーザ参加時間の短縮化を導入した参加方法による選択サーバを決定する計算時間の短縮効果について評価を行った．逐次参加型は利用中ユーザ端末が選択するサーバが決定しているため，その分の計算量が削減され，一斉参加型と比較し 1/100 以下に処理時間が短くなっており，ユーザ端末の参加時の待ち時間が非常に短縮化されたユーザ参加方法であることが確認された．また，ユーザ参加時間を短縮化したユーザ参加方法の導入により，さらに 70 %以上の計算時間の削減が実現されることを示した．

第 7 章

結論と今後の研究課題

7.1 結論

本研究では，電話会議システム，対戦型ネットゲームやネットセッション等の多地点間で通信を行うリアルタイム型アプリケーションに適用する分散処理型通信方式を提案した．提案方式では，ユーザ端末とのネットワーク遅延の少ないロケーションに配備された複数のサーバで分散処理を行い，サーバ上の仮想時刻で実際のイベント発生順序を再現することで，ユーザ端末のイベント順序性を保証する．エンド-エンドの 遅延特性を最小化するためのユーザ端末が選択するサーバの決定問題について計算複雑度の評価から，本問題が NP 困難であることを明らかにした．なお，本研究では，このサーバ選択問題を解くために線形計画法を用いた．

提案方式におけるサーバ選択問題について，遅延特性の厳しいアプリケーションへの適用を考慮した遅延許容時間やネットワーク輻輳を考慮した遅延変動について拡張を行うことで，様々な条件への適用を可能とした．

提案方式について，シンプルなトポロジでの基本性能評価，実際のネットワークを擬似した JPN48 を活用した評価を実施し，下記の数値を最小化する遅延特性に優れた通信方式であるといえる．また，ネットワーク輻輳による遅延変動も考慮でき，輻輳箇所や輻輳による遅延変動が明らかな場合は，ネットワーク輻輳も考慮したユーザ端末とサーバ間ネットワークの設計法として適用可能である．

- 許容遅延時間を超えるユーザ端末
- エンド-エンドの通信遅延時間

ユーザ端末が時間経過とともに増加する逐次参加型のユーザ参加方法について，利用中ユーザ端末の選択サーバを決定値として扱うサーバ選択問題として，ユーザ参加時間を短

縮化する拡張を行った．評価結果から，集中処理型と比較し遅延特性に優れ，ユーザ参加時の待ち時間の軽減が可能な処理量が大幅に削減可能な方式であることが確認された．

本研究により，ネットワーク内に様々なアプリケーションを配備する環境において，提案方式を活用することで，様々な用途に活用できる遅延特性に優れた通信環境を実現することが可能となり，仮想化技術の様々なアプリケーション適用への適用を広げることができる．さらに，リアルタイム型アプリケーションを提供するネットワーク設計やシステム構築においても，アプリケーションを配備する拠点やユーザ端末とサーバ間回線の設計法としての活用が期待される．

7.2　今後の研究課題

本研究における今後の課題としては，大きく 2 つに分類される．

第一の課題は，サーバ上での処理性能に関する課題である．提案方式は分散処理を行うサーバにおいて，仮想時刻で各ユーザ端末のイベントと分散処理するサーバからのイベントを順序補正をする必要があるが，ユーザ端末ごとに遅延時間が異なるため，それぞれ異なる待ち時間でイベント処理を行う必要がある．これらの処理は，ユーザ端末数や分散処理するサーバ数の増加に伴い処理量が単純増加するため，効率的な待ち時間の挿入処理を行わないとサーバあたりの収容可能なユーザ端末数に制限がでる等の処理性能に関する課題が発生する可能性がある．

第二の課題は，アプリケーションの分散処理に関する課題である．提案方式では，複数のサーバで分散処理を行うためアプリケーションによっては複数のサーバで分散処理をするためのソフトウェアの修正が必要となる可能性がある．電話会議システムのように，特定ユーザ端末の音声データをミキシングし，さらに，ミキシングされた複数の音声データをミキシングすることで，複数の音声データの合成ができるような状態を持たずに多段の処理が可能なアプリケーションは，提案方式は非常に親和性が高い．一方，アプリケーションが状態をもち，複数のユーザ端末で状態を共有するようなアプリケーションでは，複数のサーバで分散処理を行う提案方式では，サーバ間での状態整合を考慮しなければならない可能性がある．分散処理するサーバの仮想時刻上では，全ユーザ端末のイベントが全サーバで同時に実行されるため，同一の状態となるはずだが，サーバの処理性能の差や時刻のずれ等によって，状態の不整合が発生する可能性がある場合は，アプリケーションの特性に応じたソフトウェアの修正が必要と考えられる．

今後は，上記の課題に対して，実際のアプリケーションを提案方式で実装することで，より具体的な処理性能と状態同期における課題が具体化されると期待される．

提案方式の発展として，ユーザ端末が移動体端末の場合や信頼性向上のための冗長性としての分散処理について検討を進めることで，より広範囲な応用が可能となると考えられる．ユーザ端末が移動体端末の場合は，移動体端末とサーバとの距離が動的に変化するので，本検討を動的なロケーション変化に対応するサーバ選択方法として応用することが期待される．本検討におけるサーバを無線基地局として考えると，無線基地局の最大端末収容数を考慮した上で，遅延の小さい無線基地局を選択する方式等への応用が期待される．また，信頼性向上のための冗長性向上としては，本方式の逐次参加型の拡張を改良することで，サーバ故障時に当該サーバを選択していたユーザ端末を正常なサーバに高速に切り替える方式への応用が期待される．

参考文献

[1] “Network Functions Virtualisation,” http://www.etsi.org/technologies-clusters/technologies/nfv, 参照 August 2015.

[2] “ネットワーク機能の仮想化 (NFV) の概要,” NTT 技術ジャーナル, vol. 26, no. 1, pp. 70-73, Jan. 2014.

[3] “NetroSphere 構想の実現に向けた取り組み,” NTT 技術ジャーナル, vol. 27, no. 8, pp. 70-73, Aug. 2015.

[4] “NetroSphere 構想とネットワークアーキテクチャ,” NTT 技術ジャーナル, vol. 27, no. 8, pp. 8-13, Aug. 2015.

[5] “NetroSphere 構想実現に向けた技術開発の取り組み,” NTT 技術ジャーナル, vol. 28, No. 2, pp. 8-13, Feb. 2016.

[6] 四七秀貴，増田暁生，川端明生, “NetroSphere の実現に向けた取り組み,” Business Communication, vol. 52, no. 12, pp. 4-5, Dec. 2015.

[7] 川端明生，堀米英明，小谷忠司, “NetroSpherePIT の最新動向,” Business Communication, vol. 53, No. 3, pp. 20-21, Mar. 2016.

[8] “Software-Defined Networking: The New Norm for Networks,” https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, 参照 April 2017.

[9] R.M. Fujimoto, “Parallel and distributed simulation systems,” New York, John Wiley & Sons, 2000.

[10] 前川守, 清水謙多郎, 濱野純，“分散処理システムにおける同時実行制御,” 情報処理 vol. 28, no. 4, pp. 387-394, Apr. 1987.

[11] R. M. Fujimoto, “Parallel Discrete Event Simulation,” *Communications of the ACM*, vol. 33, no. 10, pp. 30-53, Oct. 1990.

[12] K. M. Chandy and J. Misra, “Distributed simulation: A case study in design and verification of distributed Fgrams,” *IEEE Transactions on Software Engineering*, vol. 5, no.5, pp. 440-452, Sep. 1979.

[13] R.E. Bryant, “Simulation of Packet Communication Architecture Computer Systems,” *Massachusetts Institute of Technology Cambridge*, MA, USA, 1977.

[14] N. James and M. Claypool, “The Effects of Latency on Online Madden NFL Football,” *Proc. of the 14th international workshop on Network and operating systems support for digital audio and video*, ACM, pp. 146-151, Jun. 2004.

[15] D. R. Jefferson,“Virtual Time,” *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 3, pp. 404-425, Jul. 1985.

[16] P. F. Reynolds Jr, “A Spectrum of Options for Parallel Simulation,” *Proc. of the 20th conference on Winter simulation*, ACM, pp. 325-332, Dec. 1988.

[17] K. M. Chandy, and M. Jayadev, “Distributed Simulation: A Case Study in Design and Verification of Distributed Programs,” *Software Engineering, IEEE Transactions on*, vol. 5, No. 5, pp. 440-452, Sep. 1979.

[18] W. L. Bain, and D. S. Scott, “An Algorithm for Time Synchronization in Distributed Discrete Event Simulation,” *Proc. of the SCS Multiconference on Distributed Simulation*, SCS Simulation Series vol. 19, pp. 30-33, 1988.

[19] W. K. Su and C. L. Seitz, “Variants of The Chandy-Misra-Bryant Distributed Discrete Event Simulation Algorithm,” *Proc. of the SCS Multiconference on Distributed Simulation*, Society for Computer Simulation Vo. 21, pp. 38-43. 1989.

[20] W. Cai and S. J. Turner, “An Algorithm for Distributed Discrete-Event Simulation: The Carrier Null Message Approach,” *Proc. of the SCS Multiconference on Distributed Simulation*, SCS Simulation Series Vol. 22, pp. 3-8, 1990.

[21] R. C. DeVries, “Reducing Null messages in Misra’s Distributed Discrete Event Simulation Method,” *Software Engineering, IEEE Transactions on*, vol. 16, no. 1, pp. 82-91, Jan. 1990.

[22] B. R. Preiss, W. M. Loucks, J. D. Maclntyre, and J. A. Field, “Null Message Cancellation in Conservative Distributed Simulation,” *Advances in Parallel and Distributed Simulation*, SCS Simulation Series vol. 23, no. 1, pp. 33-38. 1991.

[23] M. L. Yu, S. Ghosh, and E. DeBenedictis, “A Non-Deadlocking Conservative Asynchronous Distributed Discrete Event Simulation Algorithm, *Proc. of the 1991 Western Simulation Multiconference*, pp. 23-25, Jan. 1991.

[24] T. D. Blanchard, T. W. Lake, and S. J. Turner, “Cooperative Acceleration: Robust Conservative Distributed Discrete Event Simulation,” *ACM SIGSIM Simulation Digest*, ACM, vol. 24, no. 1, pp. 58-64, Jul. 1994.

[25] 高井峰生, 山城登久二, 成田誠之助, “離散事象シミュレーションにおける効率的な

メッセ時送出則," 情報処理学会論文誌, vol. 37, no. 3, pp. 430-438, Mar. 1996.

[26] 高井峰生, 山城登久二, 成田誠之助, "離散事象並列シミュレーションにおける保守的同期手法の評価," 情報処理学会論文誌, vol. 37, no. 11, pp. 2010-2019. Nov. 1996.

[27] J. Misra, "Distributed Discrete Event Simulation," *ACM Computing Surveys*, vol. 18, no. 1, pp. 39-65, Mar. 1986.

[28] D. M. Nicol, "The Cost of Conservative Synchronization in Parallel Discrete Event Simulations," *Journal of the ACM*, vol. 40, no. 2, pp. 304-333, Apr. 1993.

[29] H. T. Kung, and J. T. Robinson, "On Optimistic Methods of Concurrency Control," *ACM Transactions on Database Systems (TODS)*, vol. 6, no. 2, pp. 213-226, Jun. 1981.

[30] Y. B. Lin, B. R. Loucks, W. M. , and E. D. Lazowska, "Selecting the Checkpoint Interval in Time Warp Simulations," *ACM SIGSIM Simulation Digest*, vol. 23, no. 1, pp.3-10, Jul. 1993.

[31] A. C. Palaniswamy, and P. A. Wilsey, "An Analytical Comparison of Periodic Checkpointing and Incremental State Saving," *ACM SIGSIM Simulation Digest*, vol. 23, no. 1, pp. 127-134, Jul. 1993.

[32] H. Avril, and T. Carl, "Clustered Time Warp and Logic Simulation," *Parallel and Distributed Simulation, 1995. (PADS'95), in Proc.., Ninth Workshop on (Cat. No.95TB8096)*, pp. 112-119, Jun. 1995.

[33] B. Samadi, "Distributed Simulation, Algorithms and Performance Analysis (Load Balancing, Distributed Processing)," Doctoral Dissertation, University of California, Los Angeles, 1985.

[34] 小林真也, 福田宗弘, "階層型マルチリングによる GVT 決定アルゴリズム," 情報処理学会論文誌, vol. 41, no. 11, pp. 3161-3172. Nov. 2000.

[35] A. D. Kshemkalyani, M. Singhal, "Distributed Computing: Principles, Algorithms, and Systems," Cambridge University Press, 2011.

[36] G. F. Coulouris, J. Dollimore, and T. Kindberg, "Distributed Systems: Concepts and Design (5th Edition)," Addison-Wesley, 2011.

[37] Y. M. O. Gazali and S. Hassan, "Survey on The Event Orderings Semantics Used for Distributed Systems," *International Journal of Computer Science and Information Technology*, vol. 2, no. 3, pp. 150-158, Jun. 2010.

[38] L. Lamport, "Time, Clocks, and The Ordering of Events in Distributed System," *Communication of the ACM*, vol. 21, no. 7, pp. 558-565, Jul. 1978.

[39] X. Défago, A. Schiper, and P. Urbán, "Total Order Broadcast and Multicast

Algorithms: Taxonomy and survey," *ACM computing Surveys*, vol. 36, no. 4, pp. 372-421, Dec. 2004.

[40] C. J. Fidge, "Timestamps in Message-Passing Systems that Preserve The Partial Ordering," *Proc. of the 11th Australian Computer Science Conference*, vol.10, no.1, pp. 56-66, 1988.

[41] A. Schiper, J. Eggli, and A. Sandoz, "A New Algorithms to Implement Causal Ordering," *Proc. of the third International Workshop on Distributed Algorithms, Springer, Berlin, Lecture Notes in Computer Science*, vol.32, pp. 219-232, Sep. 1989.

[42] K. Birman, A. Schiper, and P. Stephenson, " Lightweight Causal and Atomic Group Multicast," *ACM Transactions on Computer Systems*, vol. 9, no. 3, pp. 272-314, Aug. 1991.

[43] S. Meldal, S. Sankar, and J. Vera, "Exploiting Locality in Maintaining Potential Causality," *Proc. of the ACM Symposium on Distributed Computing*, pp. 231-239, Jul. 1991.

[44] M. Raynal, A. Schiper, S. Toueg, "Causal Ordering Abstraction and A Simple Way to Implement It," *Information Processing Letters*, vol. 39, no. 6, pp. 343-350, Sep. 1991.

[45] M. Singhal, and A. Kshemkalyani, "An Efficient Implementation of Vector Clocks," *Information Processing Letters*, vol. 43, no.1, pp. 47-52, Aug. 1992.

[46] R. Scgwarz, and F. Mattern, "Detecting Causal Relationships in Distributed Computations: In serch of Holy Grail," *Distributed Computing*, vol. 7, no. 3, pp. 149-174, Mar. 1994.

[47] T. D. Chandra, and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems.," *Journal of the ACM*, vol. 43, no. 2, pp. 225-267, Mar. 1996.

[48] B. Charron-Bost, S. Toueg, and A. Basu, "Revisiting Safety and Liveness in The Context of Failures," *11th International Conference (CONCUR 2000). Lecture Notes in Computer Science*, vol. 1877. pp. 552-565, Aug. 2000.

[49] X. Défago, A. Schiper, and P. Urbán, "Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey," *ACM Computing Surveys*, vol. 36, no. 4, pp. 372-421, Dec. 2004.

[50] R. Baldoni and S. Cimmino and C. Marchetti, "A Classification of Total Order

Specifications and its Application to Fixed Sequencer-Based Implementations," *Journal of Parallel and Distributed Computing*, vol. 66, no. 1, pp. 108-127, Jan. 2006.

[51] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, Jun. 2010.

[52] "Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE1588, 2002.

[53] 節政暁生, "ネットワークゲームの仕組みとゲームデザイン," CEDEC2010 基調講演, Aug. 31. 2010, http://www.4gamer.net/games//105/G010549/20100905002/, 参照 Aug. 2015.

[54] "NETDUETTO," http://netduetto.net/, 参照 Aug. 2015.

[55] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for Vm-based Cloudlets in Mobile Computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct. 2009.

[56] W. Shi, and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78-81. 2016.

[57] "エッジコンピューティング構想," NTT 技術ジャーナル, vol. 26, no. 4, pp. 56-57, Apr. 2014.

[58] "IoT 時代を拓くエッジコンピューティングの研究開発," NTT 技術ジャーナル, vol. 27, no. 8, pp. 59-63, Aug. 2015.

[59] "Mobile Edge Computing A key technology towards 5G," http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf, 参照 Apr. 2017.

[60] N. Takahashi, H. Tanaka, and R. Kawamura," Analysis of Process Assignment in Multi-tier Mobile Cloud Computing and Application to Edge Accelerated Web Browsing," *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on*, pp, 233-234, Mar. 2015.

[61] 土屋利明, "NGN における品質基準と制御," NTT 技術ジャーナル vol. 20, no. 6, pp. 42-43, Jun. 2008.

[62] 茂木俊一, 一岡義宏, 田中健二, 西永望, 勝本道哲, "広帯域ネットワークを用いた遠隔同時音楽演奏の実験とその考察," マルチメディア通信と分散処理研究会, vol. 54, pp. 49-54, Jun. 2002.

[63] Y.W. Bernier, "Latency Compensating Methods in Client/Server *i* n-game Protocol Design and Optimization," *Proc. of the 2001 Game Developers Conference*,

pp. 73-85, Mar. 2001.

[64] P. Lothar and L.C. Wolf, "On The Impact of Delay on Real-Time Multiplayer Games," *Proc. of the 12th international workshop on Network and operating systems support for digital audio and video*, ACM, pp. 23-29, May 2002.

[65] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The Effect of Latency on User Performance in Warcraft III," *Proc. of the 2nd workshop on Network and system support for games*, ACM, pp.3-14, May 2003.

[66] T. Henderson and S. Bhatti, "Networked Games: A QoS-Sensitive Application for QoS-Insensitive Users?," *Proc. of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?*, ACM, pp. 141-147, Oct. 2003.

[67] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," *Proc. of 3rd ACM SIGCOMM workshop on Network and system support for games*, ACM, pp. 144-151, Sep. 2004.

[68] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games," *Proc. of 4th ACM SIGCOMM workshop on Network and system support for games*, ACM, pp. 1-7, Oct. 2005.

[69] K. T. Chen, P. Huang, and C. Lei, "How Sensitive Are Online Gamers to Network Quality?," *Communications of the ACM*, vol. 49, no. 11 pp. 34-38, Nov. 2006.

[70] "東日本電信電話株式会社 ビジネスイーサワイドの SLA," http://www.ntt-east.co.jp/business/service/sla/, 参照 Aug. 2015.

[71] "NTT コミュニケーションズ Arcstar IP-VPN の SLA," http://www.ntt.com/vpn/ip-vpn/data/merit.html, 参照 Aug. 2015.

[72] 徳永博之，関野公彦，久保田創一，佐藤栄，"リアルタイムグループウェアにおけるイベント順序制御の一考察," 報処理学会研究報告マルチメディア通信と分散処理，vol. 35, pp. 171-176, Apr. 1997.

[73] Y. Ishibashi and S. Tasak, "A Group Synchro-ization Mechanism for Live Media in Multicast Communications," *Proc. of the IEEE Globecom'97*, pp. 746-752, Nov. 1997.

[74] T. A. Funkhouser, "RING: A Client-Server System for Multiuser Virtual Environments," *Proc. of the Symposium on Interactive 3D Graphics*, pp. 85-92, Apr. 1995.

[75] S. Ba, A. Kawabata, B. C. Chatterjee, and E. Oki, "Computational Time Complexity of Allocation Problem for Distributed Servers in Real-Time Applications,"

Proc. of the 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Oct. 2016.

[76] R.M. Karp, *Reducibility among Combinatorial Problems*. Springer, 1972.

[77] “GNU Linear Programming Kit (GLPK),” http://www.gnu.org/software/glpk/, 参照 Aug. 2015.

[78] “IBM ILOG CPLEX,” http://www.ibm.com/software/commerce/optimization/cplex-optimizer, 参照 Aug. 2015.

[79] E. Oki, Linear Programming and Algorithms for Communication Networks, CRC Press, Boca Raton, 2012.

[80] 大木英司，通信ネットワークのための数理計画法, コロナ社，2012.

[81] “Japan Photonic Network Mode,” http://www.ieice.org/cs/pn/jpn/jpnm.html, 参照 Mar. 2016.

論文目録

1. 学術雑誌論文
   - 川端明生, 大木 英司, “リアルタイム型アプリケーションに適用する分散処理型通信方式,” 電子情報通信学会論文誌 (B)，vol. J99-B, no. 4, pp. 356-367, Apr. 2016. (査読あり)
2. 国際会議論文
   - Akio Kawabata, Bijoy Chand Chatterjee, and Eiji Oki, “Distributed Processing Communication Scheme for Real-time Applications Considering Admissible Delay,” *Proc. of the IEEE International Workshop on Communications Quality and Reliability (CQR 2016)* , May 2016. (査読あり)