博士学位論文

分散仮想環境の
リアルタイム性に関する研究

平成21年3月

茨城大学大学院理工学研究科
情報・システム科学専攻
河野　義広

論文要旨

近年，Second Life などで話題となっている分散仮想環境（以下 DVE と略記）において，サーバ・クライアント（以下 S/C と略記）型のトポロジが用いられることが一般的である．この型の DVE では，仮想状態の整合性の面で有利である．しかしながら，サーバへの負荷集中や，サーバを経由したデータ配信によるフィードバック遅れなどで応答性が犠牲となる．一方で Peer-to-peer（以下 P2P と略記）型の DVE が注目されている．P2P 型 DVE では，個々の peer が他の peer と直接通信を行うため，フィードバックの遅れが少なくて済む．しかしながら，各 peer の状態の同期保証が困難なため状態に不整合が生じる．これは整合性－応答性のトレードオフ問題として知られている．このように，リアルタイム性の高いアプリケーションにおいて，整合性と応答性はユーザの操作性や快適性といったユーザビリティに大きな影響を与える要因であるにも関わらず，現状ではアプリケーション開発者自身のノウハウとしてのみ蓄積されている場合が多い．

そこで本論文では，整合性と応答性の犠牲を最小限に抑えた最も効率的なリアルタイム型 DVE の設計方法について検討した．DVE を構築する際のトポロジは，S/C 型と P2P 型の 2 種類に大別される．ここで，各端末の役割に着目すれば，仮想空間の同期を制御する機構の有無でさらに分類できる．本研究では，仮想空間の同期制御機構が存在するものを Lockstep 型，そうでないものを Non-Lockstep 型とよぶ．こうすることにより，トポロジと役割の観点から DVE のモデルは，Lockstep 型 S/C，Non-Lockstep 型 S/C，Lockstep 型 P2P，Non-Lockstep 型 P2P の 4 種類に分類できる．ここで，Lockstep 型 DVE に分類される 2 種類のモデルでは，各端末の状態更新頻度が通信遅延，通信頻度に依存する欠点があるため，リアルタイム性の高いアプリケーションには適用し難いと考えられる．これらの事情を考慮し，本論文では，Non-Lockstep 型 DVE に着目し，整合性と応答性の観点からリアルタイム型 DVE の実現可能性について検討した．

はじめに，最も応答性に優れる Non-Lockstep 型 P2P について検討した．このモデルでは，Local 情報は即座に提示できる一方，Remote 情報は提示に時間が掛かるため，情報提示時刻に差異が発生し，共有オブジェクトの扱いが困難であることが知られる．そこで，応答性の良さをある程度維持したまま，仮想空間の整合性を維持する共有オブジェクトの管理手法を開発した．具体的には，共有オブジェクトの

管理手法に関して，各アバタ（DVE 上の仮想ユーザ）に共有オブジェクトを優先的に制御できる領域を割り当て，peer 間での共有オブジェクトに対する因果律の破綻を回避する方式を提案した．その応用例として，ネットワーク対戦型ダブルスエアホッケーに本方式を導入し，シミュレーション実験により本方式の有効性を検証した．これは，実際のオンラインゲームへの導入の可能性を示唆するものである．

次に，Non-Lockstep 型 S/C の代表例として，現在広く普及している Web 上の Ajax（Asynchronous JavaScript + XML）に着目し，ユーザが Web ブラウザを通して簡単に利用できるリアルタイム Web ゲームの実現可能性について検討した．リアルタイム Web ゲームの利点は，インストールの手間もなく，ファイアウォールなどに通信が制限されることもないため，実用的で利用ユーザ数の増加が期待できる．また，リアルタイム性を考慮して Non-Lockstep 型 DVE を採用したため，サーバはクライアント間のメッセージ仲介のみを行う．これは，一般性と融通性に配慮しつつ，サーバの負荷軽減とクライアントへの応答性を重視した設計方法といえる．また，仮想空間の整合性に配慮し，Ajax による予測処理を各クライアントに持たせた．その結果，本論文で提案した方式は，様々な種類のリアルタイム Web アプリケーションに適用可能であることが確認された．

以上より，本論文では DVE の設計方法と整合性・応答性の関連性に着目し，リアルタイム型 DVE の実現可能性について検討した．その結果，Non-Lockstep 型 DVE において，共有オブジェクトの排他制御が可能となる方式を，国内外を通じて初めて提案した．また，これと関連してリアルタイム型 DVE は，効果的な設計方法により実現可能であることを確認した．これらの点において，本研究は今後のリアルタイム型 DVE の設計方法に少なからず寄与するであろう．

Abstract

In general, a server-client (S/C hereafter) type of network topology may be employed in the Distributed Virtual Environment (DVE hereafter) such as Second Life. Although this type of DVE that offers the advantage of consistent states of objects in a virtual space, its drawbacks are well known; e.g., server overload and reduction of system throughputs. On the other hand, various peer-to-peer (P2P hereafter) network topologies are available for designing DVEs. P2P-type DVE can decentralize computing load of a server and decrease latency of turnarounds since each peer communicates directly with all of the other nodes. However, synchronization between nodes is not easy because such a network has no server for synchronization management. This issue is the so-called consistency-throughput trade-offs.

In this thesis, the author studied the most effective design strategy for real-time type DVE that maintains high quality of consistency and throughput. The topology of DVE is divided into S/C type and P2P type. Moreover, these DVE models are classified by existence of synchronizer of virtual space considering role of each terminal. In this research, the model where virtual space synchronizer exists is called Lockstep type. The other type is called Non-Lockstep type. Consequently, DVE models are classified to Lockstep type S/C, Non-Lockstep type S/C, Lockstep type P2P, and Non-Lockstep type P2P. Then, update interval depends on network latency and communication frequency in two kind of Lockstep type. Therefore, Lockstep type is difficult to be employed in real-time type DVE. In this thesis, the author studied feasibility for real-time type DVE from a perspective of consistency and throughput.

First, Non-Lockstep type P2P was discussed. In this model, while the information about a local avatar (virtual player in the DVE) can be quickly available at a local host, a remote avatar's information cannot be simultaneously displayed because of latency. Consequently, shared objects cannot be controlled jointly by all the users because of the temporal difference between the local and the remote avatars in terms of their status in reference to each other. Thus, the author proposed a way to improve consistency to maintain the shared object without sacrificing

throughput. Concretely, a priority field to control shared object is allocated to each avatar. Moreover, the author proposed the method to avoid absolutely inconsistent phenomena about shared object between peers. As an example of the application, a real-time networked doubles air-hockey was implemented for evaluation of validity of proposed method.

Secondly, as an example of the representative of Non-Lockstep type S/C, the author focused on the current Web Ajax (Asynchronous JavaScript + XML) system, the most widely used infrastructure of this type, and proposed a strategy for designing real-time Web games. When a game is actualized on the Web, users can easily play it through a browser, and this tends to lead to growth in the number of users. On the other hand, if HTTP is used to transmit the data, the transmission will not be limited by a firewall. Here, client side prediction by using Ajax can be used in real-time Web games to overcome the limitations of HTTP communication. As a result, proposed concept can be extended to various types of real-time Web interactive application.

Finally, this thesis studied feasibility for real-time type DVE from a perspective of consistency and throughput. As a result, in Non-Lockstep type DVE, the author proposed real-time exclusive control about shared object for the first time in the world. Moreover, it was confirmed that feasibility for real-time type DVE and its topology were irrelevant. Therefore, this research will contribute to the method of designing real-time type DVE in the future.

目 次

図 目 次

表 目 次

第1章　序論

1.1　分散仮想環境（DVE）の概要

今日，我が国のインターネットは急速な普及を遂げており，インターネット白書2007によると，インターネット利用人口は8,266万6,000人に上り，ブロードバンド世帯普及率は50.9% となり全世帯の半数を超えた[106]．現在，インターネットの利用はWorld Wide Web（以下Webと略記）がほとんどであり，利用サービスで多いのはアメーバブログ[20]，livedoor Blog[53]などのブログ，mixi[58]，MySpace[85]などのSocial Networking Service（以下SNSと略記）といったコミュニティサービス，Wikipedia[74]，価格.com[88]といった集合知を利用したサービスである．また，YouTube[82]，ニコニコ動画[94]などの動画共有サイト，Flickr[76]などの写真共有サイトも多くの利用者を有している．

このような状況の中，近年ではインターネット上で動作するオンラインゲームの市場が急速に普及してきている．オンラインゲームに代表されるコンピュータ上に仮想空間を構築し，これをネットワーク上の複数の端末で共有する技術を分散仮想環境（Distributed Virtual Environment；以下DVEと略記）とよぶ．現在，DVEに関する研究は盛んに行われ，オンラインゲーム[6], [18], [19], [33], [46], [48], [52], [78], [79], [95]〜[97], [99], [100], [108]だけでなく，遠隔教育[35]〜[38]，遠隔操作[54], [103], [105], [110]，遠隔音楽[61], [80], [81]，遠隔手術[104], [114]，軍事シミュレーション[54]など様々な分野に応用されている．DVEの最大の魅力は，遠隔地に点在する複数のユーザ同士が同一の時間と空間を共有し，互いにリアルタイムなインタラクションを可能とする点にある．

特に，マルチプレイヤ型のオンラインゲーム（Multi-player Online Game；以下MOGと略記）に注目すると，以下の3種類が特に利用者の多いジャンルである．

(1) 一人称シューティング（First-person shooters；FPS）

(2) リアルタイム戦略ゲーム（Real-time Strategy games；RTS）

(3) ロールプレイングゲーム（Role-Playing Game；RPG）

まず第一に，FPS は，最大で数十人程度のプレイヤが武装して撃ち合う非常にリアルタイム性の高いゲームであり，Counter-Strike [72]，Quake 4 [40]，Half-Life 2 [73]，Unreal Tournament 3 [27] が代表例として挙げられる．第二に，RTS とは命令および行動の順番が明確に決まっているターン型 の戦略ゲームとは異なり，プレイヤはリアルタイムに進行する時間に対応しつつ，戦略を立てながら敵と戦うゲームである．RTS の代表例は，Command & Conquer [26]，StarCraft II [10]，Age of Empires III [56] が挙げられる．

最後に，RPG とは「役割を演じるゲーム」の意で，プレイヤが一つの仮想人格を演じることでゲームに関わり，審判役との対話によりストーリーを作っていくゲームである．RPG の代表例としては，ラグナロクオンライン [32]，Diablo II [9]，Ultima Online [25]，Everquest II [71]，リネージュ II [60] が挙げられる．コンシューマ向けの RPG としては，ドラゴンクエスト（以下 DQ と略記）シリーズ [90]，ファイナルファンタジー（以下 FF と略記）シリーズ [92] がよく知られており，DQ IX [93]（2009 年発売予定），FF XI [91] がそれぞれ，オンライン対応となっている．また，オンライン RPG は，MORPG（Multi-player Online Role-Playing Game）と MMORPG（Massively Multi-player Online Role-Playing Game）に分類され，前者は比較的少数のプレイヤが集まり，独立したゲーム世界を作成してプレイするゲームであるのに対し，後者はサーバ内に存在するゲーム世界に対し不特定多数のプレイヤが接続するゲームである（Wikipedia より）[75]．なお，DQ IX は MORPG，FF XI は MMOPRG に分類される．

また，上記以外のジャンルとしては，Windows 専用ソフトウェアの真・三國無双 Online [84] や，任天堂 Wii 専用ソフトウェアの大乱闘スマッシュブラザーズ X [87] といったインターネット上のリアルタイムアクションゲームも注目を集めつつある．

その一方で，Second Life [51]，meet-me [89]，などの 3D 仮想空間を利用したコミュニケーションサービスも公開されている．特に，Second Life では仮想空内でユーザが創造した土地，建造物，衣服やアクセサリなどの様々なアイテムを売買することにより，実際の経済活動が行われている．また, [51] はトヨタ自動車，DELL などの自動車，PC といった製造メーカーでは広告スペースとしても利用されている．このように，DVE はオンラインゲームをはじめとして，インターネット上でさらなる広がりを見せつつある．

1.2　DVE における技術的課題

前節で示したように，DVE はインターネットを利用して急速に普及しつつある一方で，その実現性には課題が多い．DVE を構成するための重要な要素は，ⓐグラフィック描画性能，ⓑ入力デバイスの性能，ⓒ計算処理性能，ⓓ通信性能の 4 点であ

る [69]．このうち，ⓐ-ⓒは個々の端末で補うことができる要素であるのに対し，ⓓは利用ネットワーク回線の増強，あるいは端末間での協調が必要となる要素である．

DVE は，通信プロトコル，並列分散処理，グラフィック描画，マルチスレッド，データベース，ユーザインタフェースなど様々な技術の粋を結集した複雑なアプリケーションとなるため，効果的に実装することが極めて困難とされている．DVE の設計・開発における主な技術的課題には，以下の 6 点が挙げられる [69]．

課題 1. 通信頻度

DVE では通常，ネットワークを介して仮想状態の情報交換を行う．しかしながら，ネットワークは有限資源であるため，データの送受信には一定の時間間隔（通信間隔とよぶ）を空ける必要がある．そのため，ユーザは他のユーザに関する情報を高頻度に得ることができない可能性がある．なお，通信頻度の単位は [Hz] とし，端末が 1 秒間に通信可能な回数で示される．

課題 2. 公平性

DVE に参加する各ユーザの利用環境は必ずしも同一ではない．そのため，CPU やグラフィック性能の差による端末間での描画フレーム数（フレームレート）の差異，あるいは通信速度の差によるユーザの応答時間の遅れなどがユーザ間での公平性の面で課題となる．

課題 3. 分散インタラクション

ネットワークを介した情報共有には一定の待ち時間，すなわち通信遅延が存在する．通信遅延が存在する環境下において，複数のユーザが同時にインタラクションを行う場合，ユーザに対して一貫性のある仮想状態をリアルタイムに提示することは困難である．例えば，アバタ（DVE 上の仮想ユーザ）や共有オブジェクトといったオブジェクト同士の衝突判定，それに対する端末間での合意，並びにその解決策などである．

課題 4. 計算資源の管理

DVE では，多くの異なるタスクによって計算資源の競合が発生する．したがって，それらのタスクのほとんどでリアルタイムでの計算結果が要求されるため，CPU などの計算資源の管理が課題となる．

課題 5. エラー管理

DVE は複雑なシステムとなるため，エラーが発生した際の原因特定が困難である．例えば，中央サーバの異常終了によるシステム全体の停止，認証サーバの停止による後発ユーザのログイン不可状態，クライアント端末の障害による当該アバタの仮想空間からの離脱，などが挙げられる．

課題 6. スケーラビリティ

スケーラビリティとは，DVE に参加可能なユーザ数によって計られる尺度の 1 つである．スケーラビリティは，ネットワーク回線容量，CPU 性能，グラフィック性能，サーバ性能など様々な要因に依存する．

上記の課題のうち，課題 1，3 はⓓ通信性能が大きく影響し，課題 4 はⓒ計算処理性能が影響する．また，課題 2，5，6 はⓐ-ⓓのすべてが影響する．

1.3　インターネットを取り巻く社会状況

1.1 節で示したように，インターネット利用者数の急増，音声・動画をはじめとする通信データの高度化に伴い，インターネット全体の通信トラフィックが増大している．そこで，FTTH や ADSL などの通信インフラの整備によるネットワーク回線の高速化・大容量化，あるいは通信データの高圧縮化といった対策が取られている．上記の対策は当然のことながら必要ではある．しかしながら，今後の通信トラフィックの増大，ネットワーク距離に比例した通信遅延が不可避であることを考慮すれば，通信遅延自体の本質的な解決策になり得るとは言い難い．

これらの状況を踏まえ，近年ではリッチインターネットアプリケーション（Rich Internet Application；以下 RIA と略記）という概念が注目されている．RIA とは，Web ブラウザなどのクライアントの機能を活かした，柔軟なインタフェースを持つ Web アプリケーションのことであり，Ajax（Asynchronous JavaScript + XML）[63]，Adobe Flash [1]，Microsoft Silverlight [57] などが代表例として挙げられる．

以上より，インターネットを取り巻く現在の社会状況としては，ネットワーク回線の増強やネットワーク設定の変更といった負担をユーザに掛けることなく，いかにしてリッチな体験を提供できるかが重要な課題となっている．

1.4　本論文の目的と構成

本研究では，1.2 節で示した DVE の設計・開発に関する課題のうち，現在の社会状況を踏まえ，通信性能が大きく影響するものを対象とする．さらに，本研究の立場としては，1.3 節の状況を考慮し通信遅延の軽減を試みるのではなく，それを不可避なものとして受け入れ，その影響をユーザに感じさせないことを目標とする．そこで，上記課題のうち，特に通信性能が影響する課題 1，3 を主な研究対象とし，各端末が互いに協調することで通信遅延の影響の軽減を試みる．課題 2 に関しては，通信性能に影響される不公平について多少検討する．

そこでまず，第 2 章では，オンラインゲームにおけるリアルタイム性を定義する．次に，オンラインゲームにおけるリアルタイム性確保のための要素技術について紹介する．さらに，これらの研究を踏まえ本研究の目的を明確にする．

続く第 3 章では，DVE のモデルの分類を行う．DVE を構築する際のトポロジは，サーバ・クライアント（以下 S/C と略記）型と Peer-to-peer（以下 P2P と略記）型の 2 種類に大別される．S/C 型では，仮想状態の整合性（consistency）の面で有利である．しかしながら，サーバへの負荷集中や，サーバを経由したデータ配信によるフィードバック遅れなどで応答性（throughput）が犠牲となる．一方，P2P 型では，個々の peer が他の peer と直接通信を行うため，フィードバックの遅れが少なくて済む．しかしながら，各 peer の状態の同期保証が困難なため状態に不整合が生じる．これは整合性－応答性のトレードオフ問題として知られている [69]．また，各端末の役割に着目すれば，仮想空間の同期を制御する機構の有無でさらに分類できる．本研究では，仮想空間の同期制御機構が存在するものを Lockstep 型，そうでないものを Non-Lockstep 型とよぶ．こうすることにより，トポロジと役割の観点から DVE のモデルは，Lockstep 型 S/C，Non-Lockstep 型 S/C，Lockstep 型 P2P，Non-Lockstep 型 P2P の 4 種類に分類できる．ここで，Lockstep 型 DVE に分類される 2 種類のモデルでは，各端末の状態更新頻度が通信遅延，通信頻度に依存する欠点があるため，リアルタイム性の高いアプリケーションには適用し難いと考えられる．これらの事情を考慮し，本論文では，Non-Lockstep 型 DVE に着目し，整合性と応答性の観点からリアルタイム型 DVE の実現可能性について検討する．

第 4 章では，最も応答性に優れる Non-Lockstep 型 P2P について検討する．このモデルでは，Local 情報は即座に提示できる一方，Remote 情報は提示に時間が掛かるため，情報提示時刻に差異が発生し，共有オブジェクトの扱いが困難であることが知られる [109], [110]．そこで，課題 3 の分散インタラクションに着目し，応答性の良さをある程度維持したまま，仮想空間の整合性を維持する共有オブジェクトの管理手法を開発する．具体的には，共有オブジェクトの管理手法に関して，各アバタに共有オブジェクトを優先的に制御できる領域を割り当て，peer 間での共有オブジェクトに対する因果律の破綻を回避する方式を提案する．その応用例として，ネットワーク対戦型ダブルスエアホッケーに本方式を導入し，シミュレーション実験により本方式の有効性を検証する．また，課題 2 の通信遅延による不公平に関して，共有オブジェクトの管理権の観点から検討する．

第 5 章では，Non-Lockstep 型 S/C の代表例として，現在広く普及している Web 上の Ajax に着目し，ユーザが Web ブラウザを通して簡単に利用できるリアルタイム Web ゲームの実現可能性について検討する．リアルタイム Web ゲームの利点は，インストールの手間もなく，ファイアウォールなどに通信が制限されることもないため，実用的で利用ユーザ数の増加が期待できる．しかしながら，Web アプリケー

ションではHTTP通信を利用する必要があるため，通信遅延やその変動（ジッタとよぶ），パケット損などの影響を受けやすく，リアルタイムゲームの実現が困難とされている．そこで，課題1の通信頻度に着目し，通信速度や通信頻度の制限があるHTTP通信において，リアルタイムWebゲーム実現のための課題解決を目的とする．

最後に第6章では，リアルタイム型DVEの実現可能性に関して得られた知見をまとめる．ここでは，整合性と応答性の犠牲を最小限に抑えた最も効率的なリアルタイム型DVEの設計方法についてまとめる．さらに，DVEのトポロジがリアルタイム性に与える影響についても述べる．

1.5 用語の定義

以下に本論文で用いる用語の定義を行う．

サーバ・クライアント（S/C）型
: DVEを構築する際のトポロジの1つである．S/C型では，各クライアントはサーバに接続し，仮想状態の取得を行う．通常，仮想状態はサーバで一元管理されるため，整合性に優れる．しかしながら，サーバへの負荷集中や，サーバを経由したデータ配信によるフィードバック遅れなどで応答性が犠牲となる．

Peer-to-peer（P2P）型
: DVEを構築する際のトポロジの1つである．P2P型では，個々のpeerが他のpeerと直接通信を行うため，フィードバックの遅れが少なく応答性に優れる．しかしながら，各peerの状態の同期保証が困難なため整合性が犠牲となる．

Lockstep型DVE
: 仮想空間の同期制御機構が存在するタイプのDVEである．MVCループの処理に着目すると，同期制御機構が全端末からの操作情報の受信後に仮想空間の再構築を行う．

Non-Lockstep型DVE
: 仮想空間の同期制御機構が存在しないタイプのDVEである．MVCループの処理に着目すると，各端末が独自に仮想空間の再構築を行う．

MVCループ
: Modeling-View-Controlループの略で，仮想空間の再構築の際に実行される一連の処理である．具体的な処理内容は，各アバタの最新情報に基づく仮想空間の再構築（Modeling），仮想空間の最新状態の提示（View），ユーザからの自

己アバタの操作情報の取得（Control）である．MVC ループは，一定の時間間隔で実行する必要がある．

フィールド型 DVE
: 仮想空間内でのアバタや共有オブジェクトの状態が位置や速度などの物理情報を示すような DVE である．

フィールド型仮想球技
: フィールド型 DVE の中で，各プレイヤが互いの勢力圏に影響を及ぼし合いながら，ボールなどの共有オブジェクトを奪い合うタイプの DVE である（例：サッカー，ホッケー）．

共有オブジェクト
: 複数の端末で制御可能なオブジェクトである．各アバタは，共有オブジェクトに接触した場合に限りその状態に変化を与えることができる．

共有オブジェクトの管理権
: 共有オブジェクトを制御する権利である．単に管理権ともよぶ．

Allocated Topographical Zone（AtoZ）
: 各アバタに対して割り当てられた共有オブジェクトを優先的に制御できる領域である．これはフィールド型 DVE において，どのアバタが最短時間で共有オブジェクトにアクセスできるかをフィールド全体で一義的に決定するための概念であり，各アバタの位置，速度，進行方向により，他のアバタとの間の相互関係を考慮に入れて決定される．この概念を Allocated Topographical Zone（地形的な割り当て領域の意味で，AtoZ）とよぶ．

Critical Case
: peer 間での共有オブジェクトに対する干渉順序が逆転する現象である．端末間での AtoZ 計算結果の差異により，複数の peer が同時に管理権を取得し，それらのアバタが同時に共有オブジェクトに接触した場合に発生する．

Phantom Case
: 自己アバタが共有オブジェクトに接触したにも関わらず，これを制御できない現象である．Local peer が共有オブジェクトの管理権を委譲した状況下で発生し得る．

膠着状態
: すべての peer で管理権を委譲し，どのアバタも共有オブジェクトを制御できない状態を指す．膠着状態は，Phantom Case の多発を招く可能性がある．

Dead Zone
: AtoZ 境界付近でアバタの存在確率が拮抗する領域である．Dead Zone では，通信遅延と更新間隔によって生ずる情報損により管理権が不明確となるため，管理権の計算を厳密に行う必要がある．

領域種別
: フィールド型 DVE における領域を分割した個々の成分である．領域全体は各アバタに属する AtoZ，各アバタ対で定義される Dead Zone に分類することができる．

Count Down Protocol（CDP）
: Critical Case の発生回避を目的とした Dead Zone 内での管理権決定プロトコルである．各 peer は，自己アバタが共有オブジェクトまで到達するのに要する最小フレーム数を互いに計算し合うことで実現する．

優先順位方式
: 膠着状態からの早期脱出を目的とする方式である．具体的には，共有オブジェクトの近傍に存在するアバタを対象とし，peer 間の優先順位による管理権決定を行う．これにより，膠着状態からの早期脱出を実現し，Phantom Case の発生を最小限に抑制することが可能である．

Dead Reckoning（DR）
: 各端末上で遅延を伴って受信されたデータを基に，他のアバタの現在の状態を時々刻々予測する手法である．DR を用い，データ受信間隔よりも短い時間間隔で他のアバタの状態を予測・提示することで，各端末上では通信遅延，更新間隔の粗さ，並びにそれらの変動をユーザに隠すことが可能となる．

リアルタイム Web ゲーム
: Web ブラウザ上で動作するリアルタイム性の高いゲームである．

階層構造モデル
: アプリケーション，情報抽出，ネットワークの 3 階層で構成されるシステム構成のモデルである．上記の各層はそれぞれ，アプリケーションプログラム，アプリケーション・ネットワーク間のインタフェース，端末間の通信機能を表す．

情報抽出レイヤ（Information Extraction Layer）
: アプリケーションとネットワークを円滑に接続するレイヤである．情報抽出レイヤでは，アプリケーションとネットワークを仲介し，アプリケーションやユーザにとって有用な情報を抽出する役割を担う．

DR プロトコル
: 本研究室で提案している独自の DR プロトコルである．本プロトコルでは，ローカル端末においてサンプリングされた位置データを基に速度，加速度等を計算し，それらを位置データとともにリモート端末に送信する方式を取っている．リモート端末側では，受信したデータを基にアバタ位置の予測を行う．これにより，ローカル端末での微小なサンプリング間隔による予測精度の向上，クライアントでの 自己アバタの履歴情報のみの保持，の利点が挙げられる．

通信遅延
: 端末間でのデータ送信に要する時間を指す．

遅延
: 通信遅延，端末での処理遅延を合計した待ち時間である．

通信頻度
: 他の端末との通信を行う頻度である．端末が 1 秒間に通信可能な回数で示される．単位は [Hz] とする．

更新頻度
: 仮想空間の状態が更新される頻度である．単位は [Hz] とする．

通信間隔
: 他の端末との通信を行う際の時間間隔である．単位は [ms] とする．

更新間隔
: 仮想空間の状態更新を行う際の時間間隔である．単位は [ms] とする．

クライアント間のデータ伝送遅延
: S/C 型 DVE において，一方のクライアントからのサーバへのデータアップロードから，他方のクライアントのサーバからのデータダウンロードまでの時間を指す．

第2章　関連研究

前章では，DVE の概要とその技術的課題について述べた．具体的には，通信遅延，通信頻度の制限によりゲームの整合性及び応答性が犠牲となる．これらは，リアルタイム性の高いオンラインゲームにおいて，ユーザの操作性や快適性といったユーザビリティに大きな影響を与える要因であるにも関わらず，現状ではアプリケーション開発者自身のノウハウとしてのみ蓄積されている場合が多い．本章ではまず，オンラインゲームにおけるリアルタイム性に関して，ユーザ同士の意思疎通の観点からその定義を行う．次に，オンラインゲームにおけるリアルタイム性確保のための要素技術について紹介する．さらに，これらの研究を踏まえ本研究の目的を明確にする．

2.1　オンラインゲームにおけるリアルタイム性

ここでは，本論文で対象とするリアルタイム性について述べる．リアルタイム性とは，一定時間内に応答があることを保証することで評価される指標である．本論文では，ユーザ同士の意思疎通に着目し，リアルタイム性の要求度合を以下の 3 種類に分類する．

- 高いリアルタイム性が要求されるモデル
- 中程度のリアルタイム性が要求されるモデル
- 低いリアルタイム性が要求されるモデル

高いリアルタイム性が要求されるモデルの例としては，1.1 節の FPS をはじめとして，格闘，レーシング，スポーツ，アクションゲームなどが挙げられる．このモデルでは，数 10[ms]〜100[ms] 程度の遅延が許容限界とされる [7]．ここでいう遅延とは，ユーザへの情報提示に要する時間であり，通信遅延，端末での処理遅延を合計した待ち時間である．

次に，中程度のリアルタイム性が要求されるモデルの例としては，1.1 節の RTS，RPG などが挙げられる．このモデルでは，100[ms]〜500[ms] 程度が遅延の許容限界であるとされる [8]．

最後に，低いリアルタイム性が要求されるモデルとは，インタラクティブな通信は行うが，厳しいリアルタイム性は要求されないモデルである [102]．例としては，将棋，囲碁などのターン型ゲームが挙げられる．このモデルでは，数秒程度の遅延までは許容可能である．

上記に関して，各モデルの遅延許容限界までの間に端末間のデータ送受信が実現されれば，応答性を犠牲にすることなく，仮想空間の整合性を確保することができる．しかしながら，その許容限界値を超える場合，端末間での整合性確保のための待ち時間を犠牲にしても，ユーザに何からの応答を返すことが重要となる．

2.2　DVE の各要素技術に関する研究

ここでは，DVE の各要素技術に関する研究紹介を行う．DVE 上の技術的課題をサーベイしている論文 [22], [23] によれば，DVE の要素技術を (1) 時間管理技術，(2) 情報管理技術，(3) システムアーキテクチャ技術に分類している．以下では，これらの技術に関して紹介する．

2.2.1　時間管理技術に関する研究

Lockstep Synchronization

Lockstep Synchronization は，整合性を確保する最も基本的な方法であり，応答性を犠牲にしてシステムのロールバックを回避する保守的なアルゴリズムである [29]．具体的には，DVE に参加する全ての端末において，仮想状態に関する計算が完了し，その結果が全ての端末上で確認されるまでの間，DVE の時間進行を止めておくことで整合性を確保する．しかしながら，この手法では整合性を重視する代償として，ゲーム進行速度が通信遅延の程度に影響されるという欠点がある．

Delayed Global Consisntency

Delayed Global Consisntency は，各ユーザに対し同一の仮想状態を異なる時刻に提示する技術である．例えば，Qin が開発した分散ホワイトボード [67] は，ユーザ間で同一の仮想状態が提示されれば，その提示時刻は必ずしも同一である必要はないという事実に基づいて開発されている．具体的には，各イベントにタイムスタンプを付加することで，各端末上で一貫性のある仮想状態が再構築可能である．ただし，提示時刻に差異が生じる．

Time Warp

Time Warp [44] は，Jefferson によって最初に提案され，DVE の整合性を維持するためにシステムのロールバックを行う手法である．この手法では，他の端末からのメッセージ受信と同時にそれを実行する．各メッセージにはタイムスタンプが付加されており，既に実行したメッセージよりも以前のメッセージを受信した場合，その時点まで時刻を戻しメッセージの再実行を行う．そのため，各端末では過去に実行した全てのメッセージの履歴情報を保持していなければならない．さらに，過去からのメッセージを受信したことで状況が変わってしまう場合，既に他の端末に送信したメッセージを取り消さなければならない [50].

2.2.2 情報管理技術に関する研究

Dead Reckoning

Dead Reckoning（以下 DR と略記）[68] [111] とは，各端末上で遅延を伴って受信されたデータを基に，他のアバタの現在の状態を時々刻々予測する手法である．DR を用い，データ受信間隔よりも短い時間間隔で他のアバタの状態を予測・提示することで，各端末上では通信遅延，更新間隔の粗さ，並びにそれらの変動をユーザに隠すことが可能となる．これまでに，実時間システムへの適用により，リアルタイム性やユーザの作業効率の向上，情報の滑らかな提示に効果があることが，幾つかの研究で明らかにされている [68] [86] [5] [13].

Relevance Filter

通常，DVE 上で送受信されるパケット数は，ユーザ数の増加に伴い指数関数的に増加する．そこで，ユーザが知る必要のないデータをフィルタリングすることで，DVE 全体のトラフィック量を削減する方法が知られる [4], [12], [70].

具体的には，各端末は自己アバタの周囲にアプリケーションの種類，通信遅延などに依存した領域を形成する．この領域は関心のある領域の意味で Area of Interest（以下 AOI と略記）とよばれる．AOI 内にアバタが存在する端末のみとデータ送受信を行う． したがって，AOI 内に互いのアバタが存在しなければ，インタラクションが発生する可能性もないため，両者の間でデータ送受信の必要はない.

データ圧縮

データ圧縮 [3] とは，データの効率的な保持，並びに通信データの軽量化を目的とした手法である．データ圧縮は，可逆圧縮と不可逆圧縮に大別される [69]．可逆圧縮

は，パケットのエンコード方式を工夫することで，情報を損なうことなく通信データ量を削減する方法である．一方，不可逆圧縮は，パケットからの不必要と考えられるデータを削除する方法である．

パケットバンドリング

パケットは，アプリケーションが利用するデータ部分のペイロードに対し，IP（Internet Protocol）などの宛先情報であるヘッダを付加した構成となっている．そこで，数フレーム分のパケットを 1 つにまとめることで，ヘッダ部分を省略しトラフィック量の削減が可能である．この手法は，パケットバンドリング [39] とよばれ，実際のオンラインゲームでも利用されている．

2.2.3　システムアーキテクチャ技術に関する研究

ネットワークアーキテクチャ

ここでは，S/C 型，P2P 型などのトポロジに関する技術を紹介する．S/C 型の例としては，NPSNET [54]，VLNET [65]，RING [28]，VELVET [21] が挙げられる．文献 [21], [54] はマルチサーバ型 DVE における負荷分散を目的としており，各サーバの管理領域をセルとよばれる六角形の領域によって分割する．同時に，各サーバからクライアントへのデータ配信にはマルチキャストが利用される．

それ以外の例として，山河らの研究 [77] では文献 [21], [54] と同様に仮想空間を六角形に分割し，Xcast（eXplicit Multi-Unicast）[11] によりデータ配信を行う．また，自己アバタが次フィールドへ移動する際，事前にフィールド情報を先読みすることで，ユーザに対する応答性の向上，並びに DVE 全体の通信トラフィックの平滑化を計っている．

一方，P2P 型の例としては，DIVE [14]，ANGEL [101]，MiMaze [24] などが挙げられる．文献 [101] では，P2P 型オンラインゲームにおけるノード間の遅延を最適化するためのミドルウェアの実装及び評価を目的とする．そこで，全結合型の P2P ネットワークを一部の peer のみが部分的に結合するハイブリッド型 P2P に再構成することで，DVE 全体の応答性を向上させている．

通信プロトコル

通信プロトコルとしては，DVE の標準プロトコルとして策定された DIS（Distributed Interactive Simulation）プロトコル [41]，QoS（Quality of Service）確保のための RSVP （Resource ReSerVationProtocol）[83] が挙げられる．DIS は，軍事シ

ミュレーションなどの DVE での利用を目的として定義された通信プロトコルであり，送受信パケットにはプレイヤの ID，位置，向き，速度，並びに加速度などの情報が含まれる．その後，DR を目的として NPSNET [54] で利用されている．一方，RSVP は，送信先までの帯域を予約し，ネットワーク上の通信路の品質保証を行うプロトコルである．具体的には，IP，または UDP（User Datagram Protocol）の上位プロトコルとして機能し，2 台のコンピュータが通信経路上における CPU，バッファ，帯域などのリソースを確保し，QoS を保証する．

2.3 まとめ

本章では，オンラインゲームにおけるリアルタイム性について述べた．さらに，DVE の要素技術について紹介した．ここで，各要素技術の目的についてまとめると，

- 応答性を犠牲にして整合性を確保する技術
  例：Lockstep Synchronization，Delayed Global Consisntency
- 通信遅延を軽減する技術
  例：データ圧縮，パケットバンドリング
- 応答性を犠牲にすることなく通信遅延の影響を軽減する技術
  例：Dead Reckoning

に分類できる．そこで本論文では，高いリアルタイム性が要求されるモデルに対して，1.4 節で示した各端末の協調動作により通信遅延の影響の軽減を目的とする．

第3章　DVEの分類方法

高いリアルタイム性が要求されるDVEにおいて，整合性と応答性は極めて重要な要素である．そこで本章では，これらの要素に着目し，DVEのモデルをトポロジと各端末の役割の観点から分類し，それらの特徴について述べる．また，本研究で対象とするDVEモデルについても述べる．

3.1　トポロジと役割に基づく分類

DVEを構築する際のトポロジは，S/C型とP2P型の2種類に大別される．ここで，各端末の役割に着目すれば，仮想空間の同期を制御する機構の有無でさらに分類できる．本研究では，仮想空間の同期制御機構が存在するものをLockstep型，そうでないものをNon-Lockstep型とよぶ．こうすることにより，トポロジと役割の観点からDVEのモデルは，Lockstep型S/C，Non-Lockstep型S/C，Lockstep型P2P，Non-Lockstep型P2Pの4種類に分類できる．

上記の各モデルにおいて，仮想空間の再構築の際には，MVC（Modeling-View-Control）ループ[107]とよばれる一連の処理を一定の時間間隔で実行する必要がある（図3.1）．具体的には，各アバタの最新情報に基づく仮想空間の再構築（Modeling），仮想空間の最新状態の提示（View），ユーザからの自己アバタの操作情報の取得（Control）という処理である．これらの処理に着目すると，Lockstep型では同期制御機構が全端末からの操作情報の受信後に仮想空間の再構築を行うのに対し，Non-Lockstep型では各端末は独自に仮想空間の再構築を行う．そこで，MVCループに着目した各モデルのTime chartを図3.2-3.5に示す．なお，簡単のために二者間での通信を想定する．

図3.2は，Lockstep型S/CのTime chartを示している．ここでは，クライアントA，Bがそれぞれのアバタの操作情報をサーバに送信している．サーバでは，各クライアントから受信した情報をもとに，仮想空間の再構築を行い，その結果をクライアントに返している．ここで，図中のtは時刻，その添え字はフレーム番号を表す．したがって，このモデルでは各クライアントが共通の時間軸を利用している．

図3.3は，Non-Lockstep型S/CのTime chartを示している．ここでは，クライアントA，Bのそれぞれが独自にMVCループを実行している．一方，サーバの役

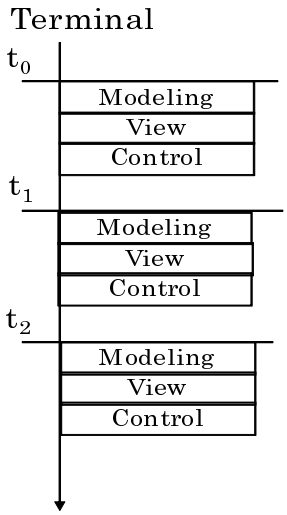

図 3.1: MVC ループの概要

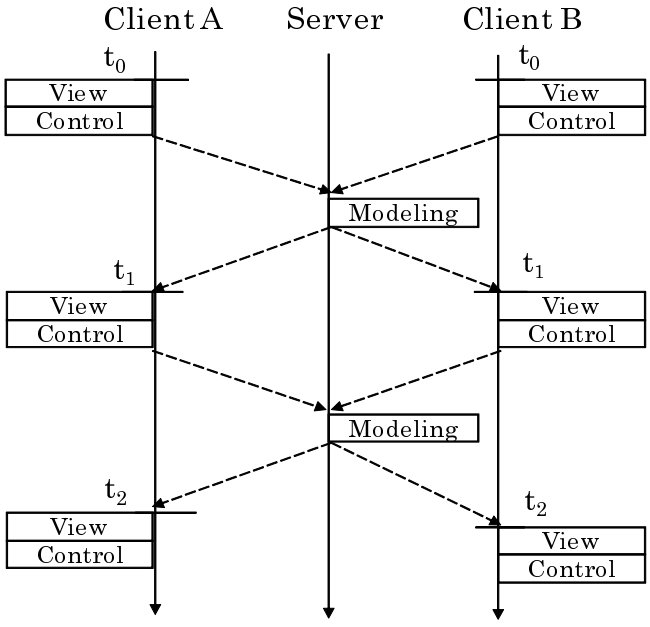

図 3.2: Lockstep 型 S/C の Time chart

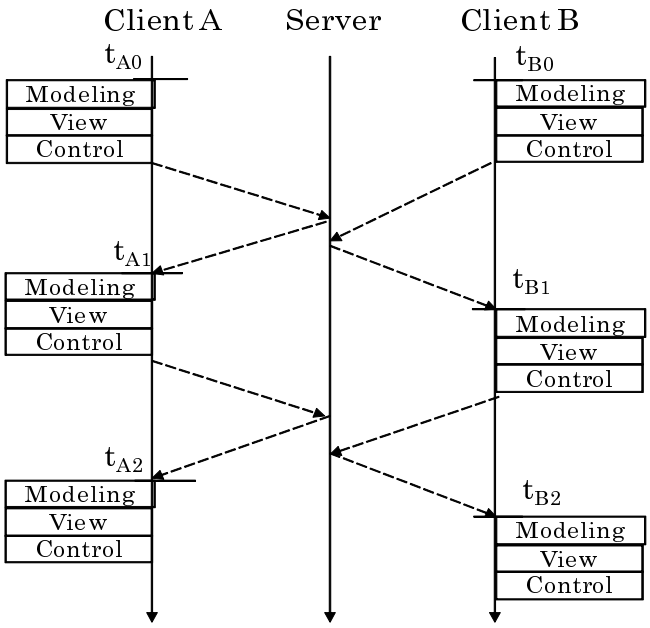

図 3.3: Non-Lockstep 型 S/C の Time chart

割は，各クライアントからの情報を他のクライアントに配信するのみである．このとき，各クライアントは一定時間毎にサーバへの問い合わせを行い，自己アバタの最新情報をアップロードするとともに，他のクライアントからの最新情報をダウンロードする．したがって，このモデルでは各クライアントが異なる時間軸を利用しており，図中の t の添え字はクライアント毎に異なる．

図 3.4 は，Lockstep 型 P2P の Time chart を示している．ここでは，Peer A，B のそれぞれが独自に MVC ループを実行している．ただし，各 peer は，他の peer からの情報を受信するまで仮想空間の状態更新を行わず，受信後にその更新を行う．こうすることにより，全 peer は同一の情報を持ち得るため，状態更新のためのアルゴリズムが peer 間で一致していれば，各 peer は一貫性のある仮想状態を得ることができる．

図 3.5 は，Non-Lockstep 型 P2P の Time chart を示している．ここでは，図 3.4 と同様に peerA，B のそれぞれが独自に MVC ループを実行している．ただし，各 peer は，他の peer からの情報の受信を待たず，一定の時間間隔で自己アバタの情報を他の peer に送信する．したがって，このモデルでは Non-Lockstep 型 S/C と同様に各 peer が異なる時間軸を利用しているため，DVE の整合性が犠牲となる．

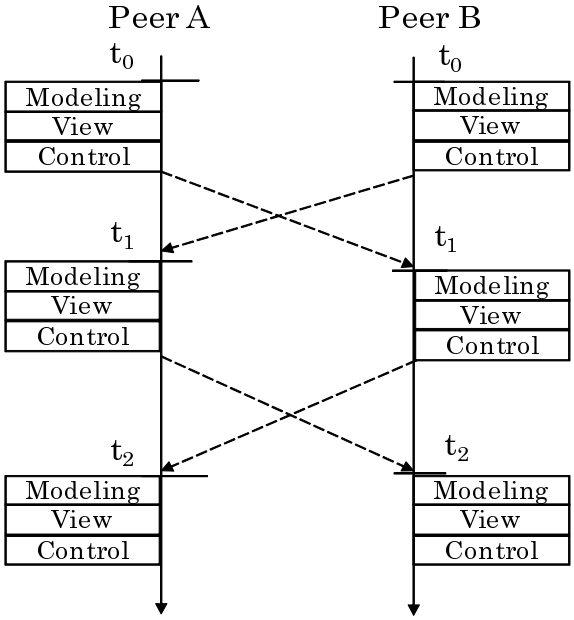

図 3.4: Lockstep 型 P2P の Time chart

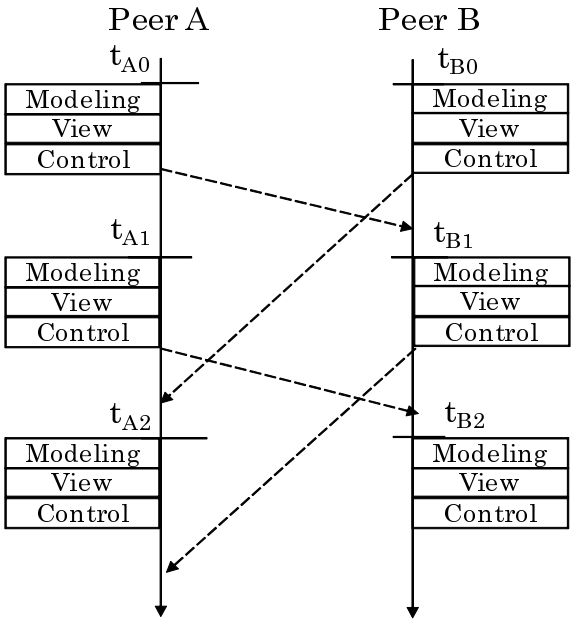

図 3.5: Non-Lockstep 型 P2P の Time chart

表 3.1: 各モデルの特徴

	通信遅延	サーバ処理	トラフィック量
Lockstep 型 S/C	往復	通信+管理	ユニキャスト：$O(n)$ マルチキャスト，Xcast：$O(n)$
Non-Lockstep 型 S/C	往復	通信	ユニキャスト：$O(n)$ マルチキャスト，Xcast：$O(n)$
Lockstep 型 P2P	片道	N/A	ユニキャスト：$O(n^2)$ マルチキャスト，Xcast：$O(n)$
Non-Lockstep 型 P2P	片道	N/A	ユニキャスト：$O(n^2)$ マルチキャスト，Xcast：$O(n)$

3.2 各モデルの特徴

前節で示した各モデルの特徴を表 3.1 にまとめる．表 3.1 は，各モデルに関して，通信遅延，サーバでの処理内容，及びトラフィック量の特徴を示している．

通信遅延に関していえば，S/C 型はサーバとの往復分の通信を要し，一方の P2P 型は他の peer までの片道分の通信でデータの送信が可能である．サーバでの処理内容としては，Lockstep 型 S/C では，各クライアントからのデータ受信，仮想空間の再構築，各クライアントへのデータ送信がある．それに対し，Non-Lockstep 型 S/C では，各クライアントとのデータ送受信のみである．なお，P2P 型の 2 種類は，サーバが存在しないため処理はない．また，システム全体のトラフィック量に関していえば，参加ユーザ数を n とし，ユニキャストによる通信を想定した場合，S/C 型では $O(n)$，P2P 型では $O(n^2)$ となる．ただし，マルチキャスト，あるいは Xcast を利用した場合，いずれのモデルもトラフィック量は $O(n)$ となる．

これまでの議論をまとめると，表 3.2 より，Lockstep 型 DVE は，整合性が容易に確保できる一方で，応答性は通信遅延に依存する欠点がある．それに対し，Non-Lockstep 型 DVE では，応答性が通信遅延に依存しない利点がある一方，整合性の確保は困難である．

3.3 まとめ

前節までの議論より，Lockstep 型 DVE に分類される 2 種類のモデルでは，各端末の状態更新頻度が通信遅延，通信頻度に依存する欠点があるため，リアルタイム性の高いアプリケーションには適用し難いと考えられる．これらの事情を考慮し，本

表 3.2: 各モデルの特徴まとめ

	整合性確保	応答性
Lockstep 型 S/C	容易	通信遅延依存
Non-Lockstep 型 S/C	困難	通信遅延非依存
Lockstep 型 P2P	容易	通信遅延依存
Non-Lockstep 型 P2P	困難	通信遅延非依存

論文では，Non-Lockstep 型 DVE に着目し，整合性と応答性の観点からリアルタイム型 DVE の実現可能性について検討する．

そこでまず，続く第 4 章では Non-Lockstep 型 P2P について検討し，第 5 章では Non-Lockstep 型 S/C について検討する．

第4章　Non-Lockstep型P2Pモデル

本章では，最も応答性に優れるNon-Lockstep型P2Pについて検討する．このモデルでは，Local情報は即座に提示できる一方，Remote情報は提示に時間が掛かるため，情報提示時刻に差異が発生し，共有オブジェクトの扱いが困難であることが知られる[109], [110]．そこで，1.2節の課題3の分散インタラクションに着目し，応答性の良さをある程度維持したまま，仮想空間の整合性を維持する共有オブジェクトの管理手法を提案する．また，課題2の通信遅延による不公平に関して，共有オブジェクトの管理権（共有オブジェクトを制御する権利）の観点から検討する．

4.1　フィールド型仮想球技

本研究では，フィールド型DVE（仮想空間内でのアバタや共有オブジェクトの状態が位置や速度などの物理情報を示すようなDVE）を対象とする．特に本章では，共有オブジェクトに対する分散インタラクションを検討するため，ボールなどの共有オブジェクトが存在するサッカーやホッケーといったフィールド型仮想球技を対象とする．

フィールド型仮想球技における共有オブジェクトの管理手法に関する研究としては，Shared Soccer[52]が挙げられる．文献[52]では，P2Pによるサッカーゲームに関して，共有オブジェクトとなるボールの管理権の同期機構を提案している．しかしながら，ボールの管理権に関して，peer間で同期を取る必要があるため，応答性が犠牲となってしまう．そこで本章では，Non-Lockstep型P2Pにおける共有オブジェクトの管理手法を提案する．

Non-Lockstep型P2Pのフィールド型仮想球技を構成するには共有オブジェクトの管理権，並びに各peerでの物理状態（共有オブジェクトと全アバタの状態）の一貫性が最も重要な課題となる．このため

(1) 共有オブジェクトの管理権を有するpeer（管理者とよぶ）の認識を全peerが共通に持つことを保証する（管理権の整合性の保証）

(2) 上記の管理権の決定はできる限り効率良くなされなければならない（管理権の効率性）

(3) 全 peer ですべてのアバタの物理状態が一致する（アバタ状態の整合性の保証）

(4) ネットワーク遅延により上記 (1) や (3) の共通認識が時間的にずれてしまうことの軽減（遅延による影響の軽減）

の 4 項目を考える必要がある．このための基本的概念として AtoZ，Dead Zone，相補予測の 3 つを提案している [98]．本論文では，4.2.1，4.2.2 で筆者らが既に提案している AtoZ，Dead Zone について概説する [98], [78], [47]．さらに 4.2.3 で Critical Case の発生を回避するためのプロトコルを提案する．そして 4.3 で仮想球技の応用例として実装した，ネットワーク対戦型ダブルスエアホッケーにより，提案プロトコルの有効性を実験的に検証する．

4.2　提案手法

4.2.1　AtoZ (Allocated Topographical Zone)

ここで与えられた条件は，各 peer には 1 名のプレイヤが存在し，それぞれが自分のアバタのみを直接的に制御できること，及び共有オブジェクトは絶えず唯一つのアバタにより排他的に管理可能な状態に置かれること，の 2 点である．この管理権を有するアバタは絶えず動的に変わり得る．ここでは，peer とアバタが一対一で対応するため，Local peer （自分がいる peer）の管理権取得と自己アバタの管理権取得は等価である．

このような条件下で前節における項目 (1) と (2) を実現するために，共有オブジェクトの扱いに関する各 peer 間の分散処理プロトコルを考える．そこでまず，各アバタが優先的に共有オブジェクトを管理できる，空間領域という概念を導入する．これはフィールド型 DVE においてどのアバタが最短時間で共有オブジェクトにアクセスできるかをフィールド全体で一義的に決定するための概念であり，各アバタの位置，速度，進行方向により，他アバタとの間の相互関係を考慮に入れて決定される．この概念を Allocated Topographical Zone（地形的な割り当て領域の意味で，AtoZ）とよぶ（図 4.1）．なお，図 4.1 の T，t_i は時刻を示す（i はフレーム番号）．ここで，i 番目のアバタの状態値を p_i，全アバタ状態の集合を P=p_1,p_2,$\cdots p_j$,$\cdots$,p_n，アバタの添え字集合を N=1，2，$\cdots$，n とすれば，p_i の AtoZ は具体的に

$$AtoZ(p_i) \quad = \quad \{\boldsymbol{x} | \boldsymbol{x} \epsilon Z, Access(\boldsymbol{x}, p_i) = \min_{j \epsilon N}(Access(\boldsymbol{x}, p_j))\} \tag{4.1}$$

で定義できる．ただし，$AtoZ(p_i)$ はアバタ p_i に属する AtoZ 領域，$\boldsymbol{x}$ は位置情報，Z は領域全体を示している．また，$Access(x, p_i)$ はアバタ p_i が $\boldsymbol{x}$ にアクセスするまで

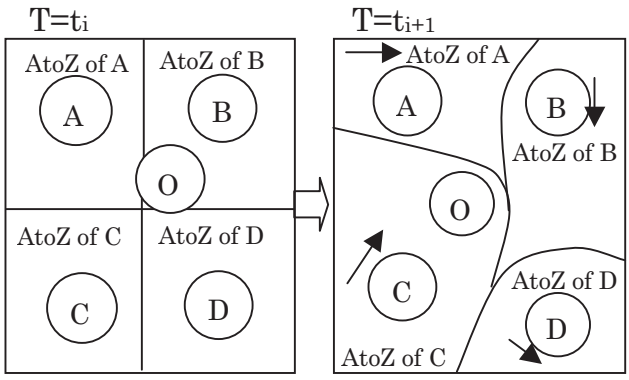

図 4.1: AtoZ の概要

の時間である．したがって，アバタ p_i の AtoZ は，p_i が全アバタ中，他よりも早くアクセスできる点 $\boldsymbol{x}$ の集合となる．なお，$Access$ 関数の計算は，文献 [98] に示すマハラノビスの距離を利用する．

ここで各 peer がそれぞれのアバタの AtoZ を共通に認識するという仮定で，現時点で共有オブジェクト O が位置する AtoZ の属するアバタがその共有オブジェクトの管理権を持つようにすれば，管理権の整合性の保証（項目 (1)），並びにその効率性（項目 (2)）の条件を満たすことができる．ただし各 peer がそれぞれの AtoZ 領域を共通に認識するという仮定が成り立つためには，項目 (3) が満たされ，かつ AtoZ の決定方式を全 peer が共有する（計算式とその計算精度が一致する）必要がある．

項目 (3) は各アバタを制御する各 peer が自分のアバタの物理状態情報を他の全 peer に絶えず送信し続け，自分以外のアバタの物理状態情報を絶えず受信し続けることで可能となる．したがってこれらをまとめると，以下の手順を各 peer が同時並列的に実行すればよい．

〈 Non-Lockstep 型 P2P での AtoZ による分散処理プロトコル 〉

Step 1. 各 peer は自己アバタの状態を他の全 peer に定期的に送信し，他の全 peer から自己以外の各アバタ状態を定期的に受信する．

Step 2. 各 peer で Step 1 に基づく仮想空間内の全アバタに属する AtoZ 領域を独自に計算する．

Step 3. Step 2 の結果，共有オブジェクト O が自己アバタに属する AtoZ に存在するなら Step 3-1 を実行，そうでないなら Step 3-2 を実行する．

Step 3-1. Local peer が O の管理権を持つものとして，自己アバタと O の相互作用を計算し仮想空間の状態を更新してその結果を他の全 peer に送信する．

Step 3-2. O が属するアバタの peer から O に関する更新情報を受信し仮想空間の状態を更新する．

上記の Step 1〜Step 3-2 の処理を各 peer が一定の時間間隔で実行し，フレームの更新を行う．フレームの更新時には，仮想空間の状態（各アバタ，共有オブジェクトの位置情報），提示画面，共有オブジェクトの管理権が更新される．ここで，Step 1 及び 3 で発生すると考えられる遅延による時間ずれの問題は前節の項目 (4) と等価である．これについては [98] で提案した相補予測により対処する．

4.2.2 Dead Zone

Critical Case と Phantom Case

前節により AtoZ に基づいて管理権を決定することで共有オブジェクトの管理が可能となる．しかしながら，AtoZ の共通認識には，まだ通信遅延分の曖昧さ（遅延による影響の軽減）の問題が残っている．つまり，共有オブジェクトに対して複数のアバタが同時にアクセスする際，通信遅延による微小なアバタ状態の差異が peer 間に生じ，それにより AtoZ の計算に差異が発生することがある．

〈 Critical Case 〉

上記の結果，複数の peer が同時に管理権を取得し，それらのアバタが同時に共有オブジェクトに接触した場合，peer 間で共有オブジェクトに対する干渉順序の逆転が生じる．この現象を Critical Case とよぶ．すなわち，管理権の競合が発生すると，Critical Case 発生の危険性が高くなるといえる．Critical Case の発生は，共有オブジェクトへの干渉に対する peer 間の因果律（causality）に破綻を来たし，その結果，共有オブジェクトに対するデータの不整合が生じる．したがって，Critical Case は絶対に回避しなければならない．

〈 Phantom Case 〉

Local peer が共有オブジェクトの管理権を委譲した状況では，自己アバタが共有オブジェクトに接触したにも関わらず，これを制御できないという現象が発生し得る．この現象を Phantom Case とよぶ．すなわち，管理権の譲歩が発生すると，Phantom Case 発生の可能性があるといえる．さらに，すべての peer で Phantom Case が発生した場合，どのアバタも共有オブジェクトを制御できない状態に陥ってしまう．この状態を膠着状態とよぶ．

Phantom Caseの発生は，共有オブジェクトの因果律に破綻は来たさない．しかしながら，共有オブジェクトに対する干渉が滞ることで，膠着状態に陥りやすく，ゲームの応答性が犠牲となる．したがって，Phantom Caseはできる限り排除する必要がある．

Dead Zone

Critical Case/Phantom Caseの発生は，それぞれのアバタ対のAtoZ境界付近への集中により起こる．このためAtoZ境界付近でアバタの存在確率が拮抗する領域を，Dead Zoneとし，管理権の計算を厳密に行うことにする．つまり，遅延と更新間隔によって生ずる情報損により管理権が不明確となる領域と考えて，Dead Zoneは次式で定義できる．

$$DZ(p_i, p_j) \ = \ \{\boldsymbol{x} | \boldsymbol{x} \epsilon Z, \frac{|Access(\boldsymbol{x}, p_i) - Access(\boldsymbol{x}, p_j)|}{\triangle t} < ([d/\triangle t] + 1) \qquad (4.2)$$

ただし，式中の[ ]はガウス記号となる．ここで，$DZ(p_i, p_j)$はアバタp_i，p_j間のDead Zone，$d[ms]$は通信遅延，$\triangle t[ms]$は更新間隔となっており，$([d/\triangle t] + 1)$で遅延フレーム数を算出できる．その他は式4.1と同様である．なお，通信遅延$d[ms]$は，あらかじめpingにより測定する．その際，設定する通信遅延には，ジッタ分の時間を上乗せする．そこで，Dead Zoneの幅は，両アバタの位置と最高速度，及び通信遅延，更新間隔に依存していると考えられる．したがって，Remote peer（他のpeer）にデータが届くまでに各アバタが最速で進行した場合でも，Critical Caseが発生しないような領域のみをAtoZとし，どのAtoZにも属さない領域をDead Zoneとする．したがって，領域全体は各アバタに属するAtoZ，各アバタ対で定義されるDead Zoneに分類することができる．このように領域が分割された個々の成分を領域種別という．

4.2.3 Count Down Protocol

上述の理由により，本章では，Critical Caseの発生回避を目的とする．また，Phantom Caseの発生は，応答性の犠牲を防ぐため，最小限に抑制することを目的とする．一方，共有オブジェクトがAtoZ内に存在する場合は，管理権が明確となるため，Critical CaseとPhantom Caseのいずれも発生し得ない．したがって，これらのケースは，各アバタと共有オブジェクトのDead Zoneへの集中により発生する．そこで，Dead Zone内での管理権決定のため，以下のプロトコルを提案する．

Count Down Protocol

Count Down Protocol（以下 CDP と略記）は，共有オブジェクトが Dead Zone に進入した際の管理権決定プロトコルである．CDP を Dead Zone 内での管理権決定にのみ適用することで，AtoZ 内での管理権の譲歩を排除すると同時に，管理権決定に要する処理量を最小限に抑制することができる（管理権条件の 2）．

このプロトコルの基本的な考え方は，他の peer から共有オブジェクト制御の事実を受信するまでの間，Local peer での共有オブジェクトの管理権取得を抑制することである．そこで，各 peer は，自己アバタが共有オブジェクトまで到達するのに要する最小フレーム数（これをカウントダウン値とよぶ．以下 CD 値と略記）を互いに計算し合うことで実現する．本プロトコルの詳細を以下に示す．

⟨ Count Down Protocol ⟩

Step 1. 各 peer は，自己アバタの CD 値を計算し，送信情報に付加する．CD 値は，自己アバタの位置・最高速度，共有オブジェクトの位置・速度をもとに計算される．

Step 2. 他の peer より受信した CD 値から遅延フレーム数を引いた値（補正 CD 値とよぶ）が 0 以下ならば，Step 3-1 に進む．すべての peer に対して正ならば，Step 3-2 に進む．なお，ジッタやパケットロスの影響により他の peer の CD 値を受信できなかった場合，現在保持する CD 値から 1 を減算する．

Step 3-1. 他のアバタが共有オブジェクトに到達している可能性があるため，管理権の取得を放棄する．

Step 3-2. いずれのアバタも共有オブジェクトに到達している可能性はないため，自己アバタの CD 値が 1 かつ，他の全アバタの補正 CD 値よりも小さいならば，管理権を取得する．このとき，CD 値が同じ値ならば，あらかじめ決められた優先順位の高い peer が管理権を取得する．

ここで，2 つの peer の場合における CDP の概要を図 4.2 に示す．このとき，peer A，B 間の片道遅延には p フレームの遅延が発生している．ここでは，簡単のために peer 間で対称の遅延を想定する．ただし，非対称の遅延にも容易に拡張可能である．

図 4.2 において，peer A が保持するアバタ B の最新の CD 値は，現時刻 t_i から遅延フレーム数 p を差し引いた時点での値となる（i はフレーム番号）．また，peer B が保持するアバタ A の CD 値も同様である．したがって，両 peer が保持する補正 CD 値は以下のように表される．

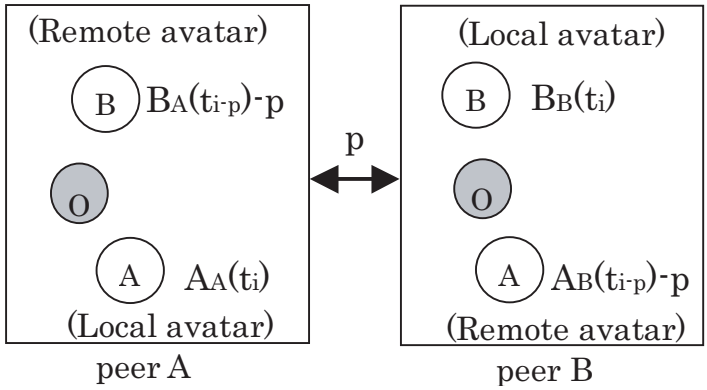

図 4.2: Count Down Protocol の概要

(peer A が保持する補正 CD 値)

$$A_A(t_i),\ B_A(t_{i-p}) - p \tag{4.3}$$

(peer B が保持する補正 CD 値)

$$A_B(t_{i-p}) - p,\ B_B(t_i) \tag{4.4}$$

ただし，各式は各アバタの補正 CD 値を示し，その添え字は補正 CD 値を保持する peer を表す.

上記の CDP において，各 peer は絶えず CD 値を送信し続ける必要がある．これは，共有オブジェクトが Dead Zone に進入した際，各 peer は管理権決定のため即座に全 peer の CD 値が必要となるためである．この CDP を Dead Zone 内の管理権決定に採用することで，Critical Case の発生を回避できると考えられる．この理由は，以下の 2 点より明らかである.

1. 各 peer は，自己アバタの CD 値が 1 の場合のみ管理権を取得できる.

2. 各 peer は，補正 CD 値から他の peer の管理権取得の可能性を正確に検出できる.

以上より，各 peer は他の peer の管理権取得を事前に検出できるため，Critical Case の発生を未然に防止できる．ただし，アバタ同士が接近するときには各々の peer がともに，他のアバタに対する 0 以下の補正 CD 値を持ち得る．すなわち，図 4.2 の場合，以下の式を満たすことになる.

$$B_A(t_{i-p}) - p \leq 0 \text{ かつ } A_B(t_{i-p}) - p \leq 0 \tag{4.5}$$

これにより，管理権の譲歩が続きゲームが膠着状態に陥りやすく，Phantom Case の発生が予想される．そこで，膠着状態を検出し，Phantom Case の発生を最小限に抑制する機能が必要となる．なお，Step 3-2 で用いる優先順位は，あらかじめ測定した各 peer から他の全 peer までの通信遅延の合計によって決定する．これは，ネットワーク距離の総和が少ない peer が管理権を得ることで，他の peer への情報配信が円滑になり，膠着状態から迅速に脱出できるためである．ここでは，ゲーム中の著しいネットワーク距離の変化はないものとする．このように決定された優先順位を用いて，Phantom Case の抑制機能を考える．

優先順位方式

Phantom Case の発生を抑制するためには，早期に膠着状態を検出し，任意のアバタが共有オブジェクトの制御を行う必要がある．そのため，管理権を取得すべきアバタの近傍に，共有オブジェクトが存在すべきである（管理権条件の 2）．そこで，共有オブジェクトの近傍アバタ（CD 値と遅延フレーム数をもとに選出）を対象とし，優先順位による管理権決定を考える．

ここで CDP の性質上，他のアバタの補正 CD 値は考え得る最小値に見積もられる．つまり，Local peer において自己アバタを近傍アバタと判断するならば，すべての Remote peer においても自己アバタは近傍アバタであると判断されることになる．したがって，各 peer は自己アバタを近傍アバタと判断した場合，peer 間の優先順位により管理権を一意に決定することが可能である．この管理権決定法を優先順位方式（priority method）とよぶ．本機能の詳細を以下に示す．

〈 優先順位方式 〉

Step 1. 各 peer は，送受信される管理権フラグ（管理権取得の有無を示すフラグ）をもとにゲームの膠着状態を検出する．ここで，一定時間以上，どの peer も管理権を取得しなければ膠着状態であると見なす．膠着状態が検出された場合，補正 CD 値をもとに近傍アバタを選出する．

Step 2. 各 peer は，自己アバタが近傍アバタかつ，自己アバタの優先順位が近傍アバタ内で最も高いと判断したならば，Step 3-1 に進む．そうでなければ，Step 3-2 に進む．

Step 3-1. 管理権の予約を行う．ここでは，膠着状態検出の直前に他の peer が管理権を取得した可能性があるため，近傍アバタ内の最大片道遅延分の待ち時間を設け待機後，管理権を取得する．

Step 3-2. 管理権を取得せず，近傍アバタ内の最大往復遅延分の待ち時間を設け待機する．

Step 4. いずれかの peer が管理権を取得し，膠着状態からの脱出が確認された場合，待ち時間を 0 に初期化する．また，管理権の予約があるならば破棄する．

上記の優先順位方式を CDP の追加機能として用いることで，Critical Case の発生を回避すると同時に，Phantom Case の発生を最小限に抑制することができる．この方法では，膠着状態の検出後，即座に管理権を決定でき，一定時間後に膠着状態からの脱出が可能となる．そのため，待ち時間を最小化し Phantom Case の発生を最小限に抑制できる．ただし，すべての peer が自己アバタを近傍アバタと判断しなかった場合，いずれの peer も管理権を取得できず管理権の譲歩が続くと考えられる．その場合，実際にはどのアバタも近傍アバタではないことを意味するため，待ち時間を初期化しゲームを続行することで対処する．なお，優先順位方式で用いる待ち時間は，ジッタを含めた通信遅延とする．

また，上記の CDP では，各 peer は自己アバタと他のアバタとの相互作用のみで管理権の決定が可能である．したがって，各 peer は管理権決定のための処理を分散でき，計算量を $O(n)$ に抑えることができる（n はプレイヤ数）．

4.3 シミュレーション実験

CDP の有効性を検証するための実験を行った．実験では，管理権委譲のための閾値調整法（Tuning Method）との比較実験を行った．また，実際のオンラインゲームとして適用可能か検証するため，被験者実験を行った．

4.3.1 実験タスク

実験タスクとして，ネットワーク対戦型ダブルスエアホッケーを試作した（図 4.3）．この図において左側は実際のゲーム画面，右側は AtoZ の様子を表した画面である．ここで，白色パックが共有オブジェクトであり，パックを打つ白黒のマレットが各アバタとなる．また，AtoZ 画面では，白円がパックを表し，その他の楕円は各マレットを表す．また，中央の黒い帯は各アバタの AtoZ 同士を分離する Dead Zone を表している．そして，Dead Zone によって分離された領域が各アバタの AtoZ となる．ここで，試作したエアホッケーではフィールド内を互いに自由に移動可能とした．これは，フィールド型仮想球技全般における AtoZ，Dead Zone，Critical Case，Phantom Case の評価を行うためである．ここで，Critical Case が発生した場合，パックに対

する干渉順序が peer 間で逆転し，パックの位置が著しくずれてしまう可能性がある．また，Phantom Case が発生した場合，マレットはパックを打つことができず，パックをすかすことになる．

また，ゲームにおける仮想空間，及びシステムの設定条件を以下に示す．ここで，各 peer は全結合型のネットワークトポロジを構成し，ユニキャストを用いて通信を行う．なお，各 peer はゲーム開始前に NTP などにより時刻同期を取得する．

(i) 仮想空間

- 対戦台の大きさ: 縦 2.89 [m], 横 1.48 [m]
- 台の動摩擦係数: 3.00×10^{-5}
- パック半径: 4.1 [cm]
- マレット半径: 8.5 [cm]
- パック速度の上限: 3.0 [m/s]
- マレット速度の上限: 1.0 [m/s]
- マレットとパックの反発係数: 0.8
- パックと壁の反発係数: 0.8

(ii) システム

- 同期取得のための時刻情報の通信: TCP
- ゲーム中の情報通信: UDP
- 更新間隔: 25 [ms]

マレット自動化アルゴリズム

閾値調整法との比較実験では，客観的な評価を行う必要がある．そこで，プレイヤによるマレット操作等の不確定要素を排除するため，マレットの動作を自動化したシミュレーション実験を行う．マレットの自動化アルゴリズムを以下に示す（図 4.4）．

① マレット（自己アバタ）とパック（共有オブジェクト）の距離を求める．次に，最速で進行したパックがその距離を進行するまでのフレーム数 x を求める．

② x フレーム後のパックの位置を予測する．

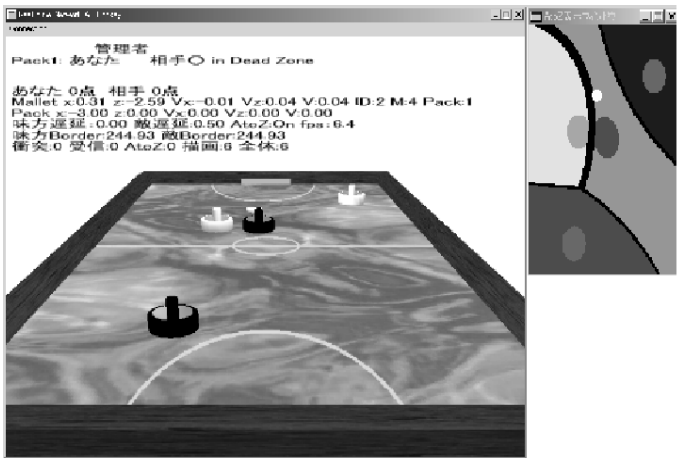

図 4.3: エアホッケーのゲーム画面

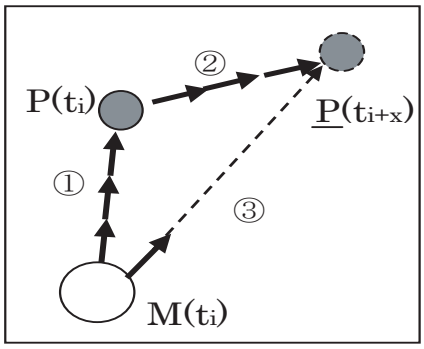

図 4.4: マレット自動化アルゴリズム

③ x フレーム後のパック位置に向かって最速で進行するようにマレットの速度を決定する.

図 4.4 において，$M(t_i)$，$P(t_i)$，$\underline{P}(t_{i+x})$ はそれぞれ，時刻 t_i のマレット位置，時刻 t_i のパック位置，時刻 t_{i+x} のパック予測位置を表す．なお，マレット自動化アルゴリズムでは，パックに適当な初速度を与える．

4.3.2 実験条件

この実験の評価項目は

(i) 全 peer での管理権の整合性の維持

(ii) Critical Case の発生頻度

(iii) Phantom Case の発生頻度

(iv) Dead Zone 内での管理権取得の割合

である．項目 (i) は，管理権の状態を求めることで評価できる．項目 (ii)，(iii) は，共有オブジェクトが Dead Zone 内に存在する場合の Critical Case/Phantom Case の発生確率の調査により行う．これは Critical Case/Phantom Case は，共有オブジェクトが Dead Zone 内に存在し，複数のアバタが同時に接触した場合にのみ発生するためである．項目 (iv) は，各 peer の優先順位別に，CDP による管理権取得の割合，優先順位方式による管理権取得の割合の調査により，ゲームの公平性を評価する．本章では，Critical Case 発生の回避，Phantom Case 発生の抑制が主な課題である．ただし，ゲームにおいて公平性は重要な要素となるため，参考として固定の優先順位による公平性への影響を調査する．ここで，模擬遅延は文献 [15] で提供されている NistNet を用いて，pareto 分布に基づく遅延を発生した．また，敵同士の通信遅延として，以下の 3 種類を用意した．ここでは，LAN などのイントラネットを想定したほぼ遅延なしの環境，インターネットを想定した平均通信遅延 57[ms] の環境，さらにそれらの中間として平均通信遅延 28[ms] の環境を採用した．なお，57[ms] の通信遅延は，国内のあるサイトに対して ping による測定結果を採用した．ここでは，味方同士はネットワーク距離が近く，敵同士はネットワーク距離が遠いという仮定を置く．なお，味方同士での通信遅延は設定されていないが，更新間隔の影響により互いに他の peer の情報は，1 フレーム遅れで届くことになる．

- 遅延なし
- 平均 28.0 [ms]，標準偏差 4.0 [ms]
- 平均 57.0 [ms]，標準偏差 8.1 [ms]

以上の条件で，マレット自動化アルゴリズムを用いてダブルスエアホッケーの実験を行った．実験は，閾値調整法，優先順位方式なしの CDP，CDP の 3 種類のプロトコルと遅延（3 種類）の組み合わせ 9 パターンに対して，それぞれ 100,000 フレーム分（各 peer で 25,000 フレームずつ）のデータを採取した．ここで，実験で設定したパックとマレットの半径より，1 個のパックに対して同時にアクセス可能な最大マレット数は 4 個であると判断できる．したがって，実験では最も Critical Case が発生しやすいと考えられる状況で最小規模の環境を想定しており，この環境で Critical Case が発生しなければ，CDP の有効性を示すことができる．なお，シミュレーション実験の評価を行うため，得点は入らないように設定した．

ここで，閾値調整法について簡潔に説明する．閾値調整法では，以下の式に従って Dead Zone 内の管理権決定を行う．

$$\begin{aligned} &local\ avatar \cdots\ if\ |local(t_i) - O(t_i)| + th(t_i) \\ &\qquad\qquad < |remote_j(t_i) - O(t_i)| \forall j, \\ &remote\ avatar \cdots\ otherwise. \end{aligned} \tag{4.6}$$

ここで，$local(t_i)$，$remote_j(t_i)$，$O(t_i)$ はそれぞれ，時刻 t_i における local avatar の位置，j 番目の remote avatar の位置，共有オブジェクトの位置と定義する．また，$th(t_i)$ は時刻 t_i における閾値であり，この値を大きく設定することで管理権の競合を回避できる．閾値調整法では，$th(t_i)$ の値を管理権の状態（競合，譲歩，整合）によって調整することで，Critical Case/Phantom Case の削減を行う．なお，実験で使用した閾値調整法における閾値 $th(t_i)$ の調整は以下のとおりである．

0. 初期値：4cm

1. 管理権競合時：$th(t_i) = th(t_{i-1}) + (4 \times Competition\_repeats)$（上限:40cm）

2. 管理権譲歩時：$th(t_i) = th(t_{i-1}) + (4 \times Concession\_repeats)$（下限:4cm）

3. 管理権整合時：$th(t_i) = th(t_{i-1})$

ここで，Competition_repeats と Concession_repeats はそれぞれ，管理権の競合と譲歩が続いているフレーム数である．実験結果を図 4.5-4.8 に示す．

また被験者実験では，被験者の操作でも Critical Case が発生しないかどうかを調査する．ここでは，上記の模擬遅延の中で最大である 57[ms] の遅延において，CDP を適用し実験を行う．それ以外の条件は，シミュレーション実験と同じである．また被験者実験の評価項目は，Critical Case/Phantom Case の発生確率である．

4.3.3 考察

まず，各図について説明する．図 4.5 は各プロトコルにおける管理権の状態を示す．管理権の状態には，競合，譲歩，整合の 3 つの状態があり，競合の状態が増加すると Critical Case が発生しやすくなり，譲歩の状態が増加すると Phantom Case が発生しやすくなる．次に，図 4.6，4.7 はぞれぞれ，Dead Zone 内での Critical Case，Phantom Case の発生確率を示す．そして，図 4.8 は，Dead Zone 内での管理権取得の割合を示す．この図では，各 peer の優先順位別に，CDP と優先順位方式における管理権取得の割合を示している．

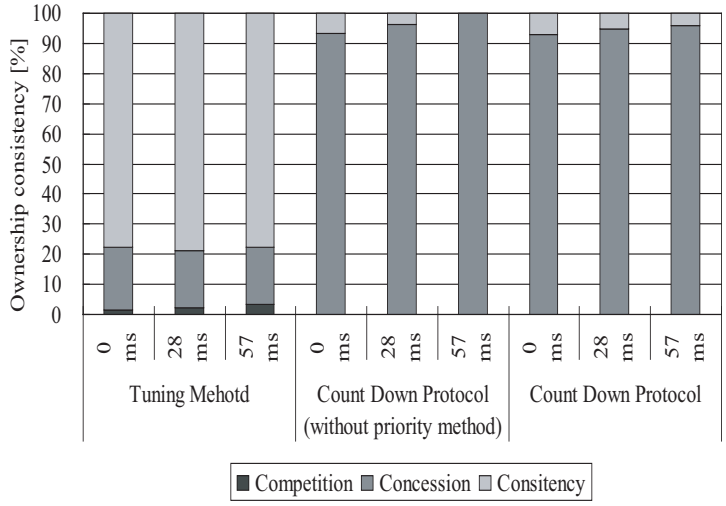

図 4.5: 各プロトコルにおける管理権の状態

図 4.5 を見ると，閾値調整法では遅延の増加に伴って管理権競合の状態が増加している．一方，譲歩の状態は遅延が増加しても大きな変化は見られない．閾値調整法では，各 peer は管理権フラグや領域種別の情報を他の peer に送信し，各 peer はそれらの情報をもとに閾値 $th(t_i)$ を調整することで，管理権競合の回避を行っている．しかしながら，通信遅延の影響によりそれらの情報の伝達に遅れが生じるため，各 peer は競合の状態を即座に検出することができない．そのため，$th(t_i)$ の値が引き上げられず，競合が多発する．一方，CDP では，競合の状態はほとんど発生しなかった．これは，各 peer が Dead Zone 内では常に最悪のケース（Critical Case の発生）を想定し，次の状態を決定するためである．その一方で，ゲーム全体の 90%以上が譲歩の状態であった．CDP では，Critical Case 回避のため共有オブジェクト制御の直前にのみ管理権を取得できる．また，シミュレーション実験を行った環境のようにアバタ同士が接近する状況では，Critical Case 発生の危険性がより高くなる．したがって，Dead Zone 内では，Critical Case 回避のため譲歩の状態が支配的となる．これにより，ゲームが膠着状態に陥りやすく，優先順位方式の利用が必要不可欠となる．優先順位方式を利用することで，競合の状態を増加させず，譲歩の状態を減少させることができる．

図 4.6，4.7 より，閾値調整法では通信遅延の増加に伴って Critical Case が増加している．一方，Phantom Case は遅延が増加しても大きな変化は見られない．これは，管理権の状態と同様に，通信遅延の影響により $th(t_i)$ の値が引き上げられず，Critical Case が発生しやすくなるためである．したがって，閾値調整法では Critical Case の発生を回避することはできない．

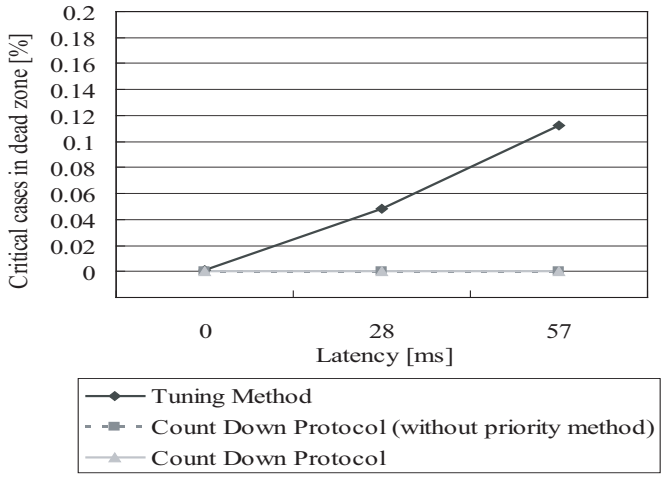

図 4.6: Dead Zone 内での Critical Case 発生確率

次に CDP では，遅延が増加しても Critical Case は全く発生しなかった．このことより，最も Critical Case が発生しやすい状況において，CDP の有効性を示すことができたといえる．ただし，優先順位方式を利用しない場合，遅延の増加に伴い Phantom Case が急激に増加し，膠着状態に陥りやすくなってしまう．

優先順位方式を利用した場合，遅延が増加しても Phantom Case の発生を閾値調整法の 2 倍程度に抑制することができた．なお，図 4.5 からは優先順位方式を利用しても，譲歩の状態の大きな減少は見られないが，いずれかの peer が管理権を取得することで膠着状態から脱出でき，Phantom Case の発生が抑制されている．すなわち，この Phantom Case 発生確率の違いは，Critical Case の発生を回避するための必要最小限のリスクである．以上より，CDP は遅延が増加しても Critical Case の発生を回避するプロトコルであるといえる．さらに，優先順位方式と組み合わせることで，遅延が増加しても Phantom Case の発生を最小限に 抑制することが可能となる．

また，味方同士には通信遅延が設定されていないため，通信遅延の増加による影響（Critical Case/ Phantom Case の増加等）は，すべて敵同士の通信遅延の増加によるものである．

図 4.8 より，優先順位の高い peer の方が Dead Zone 内での管理権取得の割合も高いことが分かる．また，この差は通信遅延の増加に伴いより顕著となっている．したがって，固定の優先順位を用いて管理権の決定を行った場合，通信遅延の増加に伴い，peer 間でより大きな不公平が生じることが分かる．ゲームにおいて，このような不公平は避けなければならないため，今後はゲームの公平性を保証することが

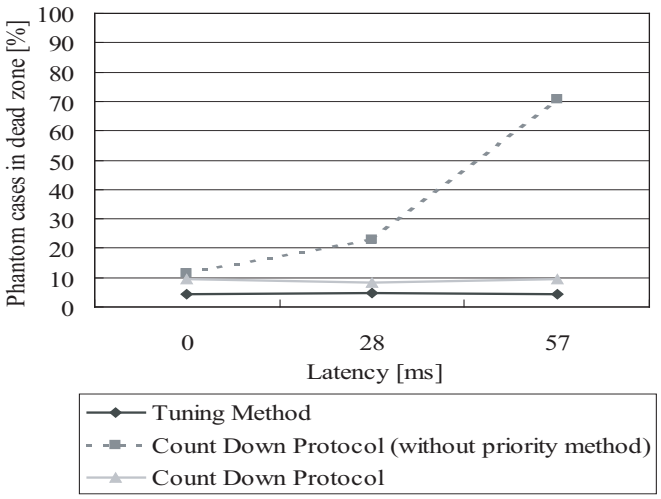

図 4.7: Dead Zone 内での Phantom Case 発生確率

課題となる．

また，被験者実験の結果，Critical Case は一度も発生しなかった．そして，Phantom Case はシミュレーション実験の結果より少なく，0.83%しか発生しなかった．したがって，CDP は実際のオンラインゲームにも十分適用可能であると考えられる．

4.4　まとめ

本章では，Non-Lockstep 型 P2P において，1.2 節の課題 3 の分散インタラクションに着目し，応答性の良さをある程度維持したまま，仮想空間の整合性を維持する共有オブジェクトの管理手法を提案した．具体的には，各 peer が共有オブジェクトまで到達するのに要する最小フレーム数を互いに計算することで，Critical Case の発生を未然に防止する CDP を提案した．その応用例として，ネットワーク対戦型ダブルスエアホッケーに CDP を導入し，マレット自動化アルゴリズムを用いたシミュレーション実験により本プロトコルの有効性を検証した．

その結果，最重要課題である Critical Case の発生回避が確認できた．このことより，Non-Lockstep 型 DVE において，共有オブジェクトの排他制御が可能となる方式を，国内外を通じて初めて提案した．さらに，優先順位方式により Phantom Case の発生を最小限に抑制することが可能となった．しかしながら，課題 2 の公平性に関しては，peer 間で不公平が生じる結果となった．

最後に，実際に被験者による実験を行ったところ，Critical Case は一度も発生し

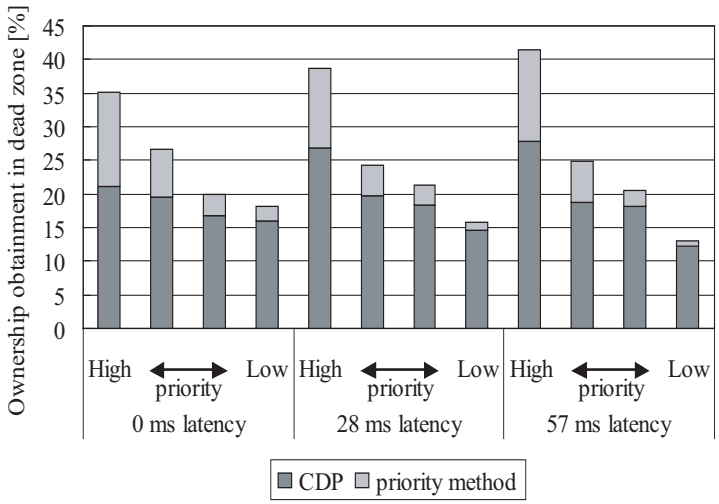

図 4.8: Dead Zone 内での管理権取得の割合

なかった．これらの結果は，実際のオンラインゲームへの導入の可能性を示唆するものである．

第5章　Non-Lockstep型S/Cモデル

本章では，1.2 節で示した課題 1 の通信頻度に関して，Non-Lockstep 型 S/C の代表例として，現在広く普及している Web に着目し，ユーザが Web ブラウザを通して簡単に利用できるリアルタイム Web ゲームの実現可能性について検討する．また，Web 上でのリアルタイム型 DVE の意義，Web 上で利用可能なリアルタイム性確保技術について述べる．その上で，整合性と応答性の犠牲を最小限に抑えた最も効率的なリアルタイム Web ゲームの基盤技術の検討を行う．

5.1　リアルタイムWebゲーム

5.1.1　リアルタイムWebゲームの意義

現在，1.1 節で示したように Web 上のサービスは急速な増加傾向にあり，今後のさらなる発展が予想される．このような状況の中で，Web ブラウザ上で動作する Web ゲームも増加している．Web ゲームの利点は，インストールの手間もなく，ファイアウォールなどに通信が制限されることもないため，実用的で利用ユーザ数の増加が期待できる点にある．

ここで，Web ゲームの例として，ディノパライソ [59] という無料オンラインキャラ育成ゲームが報告されている．しかしながら, [59] はマルチプレイヤ対応であっても，キャラクタが１日１回だけ行動するようなターン型ゲームであり，リアルタイム性は極めて低い．一方，Java ベースの MMORPG として RuneScape [42] も報告されているが，あまりリアルタイム性の高いゲームではない．一方，岡本らによるシステム [62] も同様のシステムとして試作している．しかしながら，これは Web ゲーム作成のためのプラットフォームを重視したものであり，リアルタイム性に着目したものでは無く，その評価を行ってもいない．

また，Shockwave ベースのリアルタイム多人数シューティングゲームとして Tank Ball 2 [55] が報告されている．しかしながら，このゲームにおいて戦車と弾の衝突判定は最も重要な要素であるにも関わらず，仮想空間の整合性，及び公平性は保証されていない．これは，整合性－応答性のトレードオフ問題として知られる [69].

そこで本章では，整合性と応答性の犠牲を最小限に抑えた最も効率的なリアルタイム Web ゲームの基盤技術の検討を目的とする．これは，今後のリアルタイム Web ゲームの方向性を示すための重要かつ不可欠な調査である．ここで，本研究では一般性と融通性に配慮しつつ，サーバの負荷軽減とクライアントへの応答性を重視し，サーバはクライアント間のメッセージ仲介のみを行うものとする．したがって，通常の Web サーバ程度の性能と機能で十分実現可能であり，運用コストの大幅な削減が期待できる．ただし，各クライアントの状態を同期させるには，各クライアント側で対処する必要がある．

5.1.2　リアルタイム Web ゲームの要件

リアルタイム Web ゲームの前提条件は「Web の利用」,「Web サーバの機能はクライアント間のメッセージ仲介のみであること」の 2 点である．

そこで，上記の前提条件を考慮したリアルタイム Web ゲームの要件は，以下の 3 点となる．

(1) 現状のシステムで利用可能であること

(2) 整合性：端末間で可能な限り同一のゲーム状態が提示されること

(3) 応答性：遠隔アバタの更新頻度が通信頻度に影響されないこと

本章では Web を前提とするため，HTTP 通信を利用したアプリケーションが上記 3 つの要件を満たさねばならない．このうち，要件 (1) はリアルタイム Web ゲームを広く普及させるための最重要項目である．具体的には，一般的なスペックの PC，ネットワーク回線，並びに通常の Web ブラウザにおいて利用可能であることが重要である．

要件 (2)，(3) について，HTTP 通信では，データ伝送時の時間遅れやその変動，パケット損などの影響を受けやすく，リアルタイムゲームの実現が困難とされている．そこで，通信速度や通信頻度の制限がある HTTP 通信において，リアルタイム Web ゲーム実現のための課題解決を目的とする．

そこでまず，要件 (2) について，本章では Web サーバはクライアント間のメッセージ仲介のみを行うため，仮想空間の整合性はクライアント側で確保する必要がある．しかしながら，通信遅延などの影響により，端末間で提示される情報に差異が生ずるため，アバタ情報の同期は困難である．そこで，各端末上で遠隔アバタに対して予測処理により対処することから,「端末間のアバタ位置情報の差異がゲームの破綻しない範囲であること」を整合性の要件とする．なお，ゲームが破綻する条件は，ゲームの種類に依存するため個別に定義する必要がある．

次に，要件 (3) について，リアルタイム性の高いゲームにおいて，円滑なゲーム遂行のために人間が許容できる応答速度は，100[ms] 以下であることが実験的知見として得られている [102] [7]．そこで本研究でもこれを採用し，「100[ms] 以内に相手アバタの位置情報が更新されること」，すなわち，「1 秒間に 10 フレーム以上，相手アバタの最新状態を描画できること」を応答性の要件とする．ここで，ゲームロジックと通信部分が同期している場合，サーバとの通信が完了するまで仮想空間の状態を更新できない．そのため，仮想空間の更新頻度がサーバとの通信頻度に依存してしまう．

5.1.3 リアルタイム Web ゲームの実現技術

本節では，リアルタイム Web ゲームの実現に利用可能な技術を検討する．具体的には，Non-Lockstep 型 Web ゲームを実現するために，ゲームロジックと通信部分の分離が可能な技術について検討する．

上記の条件を満たす技術としては，Ajax [63] や Flash [1] が知られる．Ajax は，JavaScript の HTTP 通信機能の利用により，Web ブラウザ上で Web サーバとの非同期通信とインタフェースの構築を実現する技術の総称である．さらに，プラグインを必要とせずに Web ブラウザのみで動作するため，Goole Maps [31]，Gmail [30] などでも利用されており，Google の Web アプリケーションを通じてその実用性が世間に認知されつつある．一方，Flash は，ベクター画像と ActionScript とよばれるスクリプト言語の組み合わせにより，アニメーション，ゲームなどのインタラクティブなコンテンツを作成するためのソフトウェアである．さらに，ActionScript の利用により，Ajax と同様に XML を用いた非同期通信が可能である．ただし，Flash プラグインのインストールが必要となる．

その一方で，通信プロトコルの観点から Comet [16] とよばれる技術が注目されている．Comet は，HTTP 通信を利用して擬似的にサーバプッシュ型の Web アプリケーションを実現する技術である．通常，Ajax などでは一定時間毎のサーバへの問い合わせにより，サーバとのデータの送受信を実現する．それに対し，Comet では，サーバはクライアントからの要求に対し即座に応答せずに，サーバ上で状態更新が行われたタイミングで応答を返す．これにより，サーバとクライアント間の通信回数の削減，並びにクライアント間のデータ交換に要する遅延の軽減が可能となる．しかしながら，サーバは各クライアントとの接続を維持する必要があるため，サーバのリソースが消費され続けるという欠点がある．

ここで，非同期通信を実現するための技術として見れば，Ajax と Flash の間に大差はない．その一方で，フレームワークに関しては，Ajaxian.com [2] の調査によれば，prototype.js [66]，jQury [45]，OpenLaszlo [49] が注目を集めている．このよう

に，Ajax で利用可能なフレームワークは数多く存在しており，今後の普及・発展が予想される．なお，Comet の利用に関しては，通常の Web サーバとは異なる要求を満たす必要があるため，本研究においては利用の対象外とする．

以上の理由より，本研究では Ajax [63] を採用し，ゲームロジック部分と通信部分の分離を行う．また，ゲームロジック部分を JavaScript で記述することにより，予測処理を各クライアントに持たせることが可能である．ここで，Ajax の利点を以下に示す．

- サーバとの非同期通信
- クライアント側でのゲームロジックの実行
- Web ページの部分更新

以上の利点により，各クライアントは仮想空間の再構築に必要な MVC ループを独自に実行できるため，サーバの負荷が大きく軽減される．したがって，5.1.2 節の要件 (3) を満たすことが可能となる．

さらに，クライアント側の予測処理として DR を利用することにより，HTTP のような通信頻度の制限がある環境においても，遠隔アバタの状態をリアルタイムに提示することが可能となる．

5.2　システム設計

5.2.1　階層構造モデル

これまでのアプリケーションでは，通信方式などのネットワーク部分を考慮したプログラミングが必要とされた．そこで本節では，新たな階層構造モデルを提案する（図 5.1）．具体的には，アプリケーションとネットワークを円滑に接続する情報抽出レイヤ（Information Extraction Layer）を提案する．情報抽出レイヤでは，アプリケーションとネットワークを仲介し，アプリケーションやユーザにとって有用な情報を抽出する役割を担う．ここで，各レイヤの役割を以下に示す．

- アプリケーション
  - アプリケーションプログラム
- 情報抽出
  - アプリケーション，ネットワークとのインタフェース

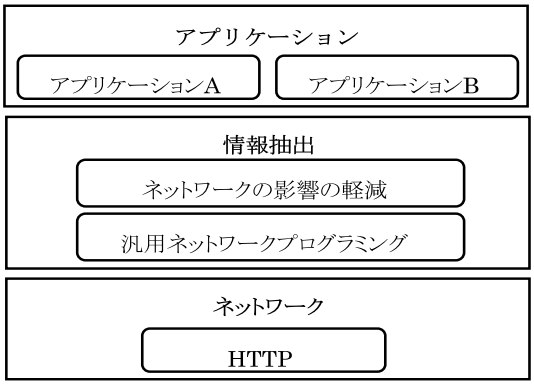

図 5.1: 階層構造モデル

  - ネットワークの影響の軽減
  - ネットワークからの情報抽出

- ネットワーク
  - 端末間の通信に関わる機能
  - サーバによるデータ集計を含む

情報抽出レイヤは，通信遅延，通信頻度の制限といったネットワークの影響をアプリケーションに対して隠蔽することを主目的とする．また，煩雑なネットワークプログラミングの汎用化を行い，開発効率の向上も目的とする．そこで，情報抽出レイヤの 1 機能として，遠隔アバタ情報の予測手法である DR を導入する．

5.2.2 システムアーキテクチャ

図 5.2 は，5.1.2 で示した要件を満たすために設計されたシステムアーキテクチャであり，図 5.1 の階層構造モデルの具体的実装例である．

図 5.2 において，各端末は GUI Handler，Modeler，Viewer，Dead Reckoning Provider（DRP），Network Handler の 5 つのコンポーネントを持つ．

ここで，MVC ループは各クライアントのアプリケーションレイヤで管理する．また，Ajax の利用により MVC ループの操作は，データ通信と非同期に実行可能である（図 5.3）．図 5.3 において，各端末は遠隔端末のデータ送信間隔で受信したデータに基づき，MVC ループ内で遠隔端末の情報を取得する．

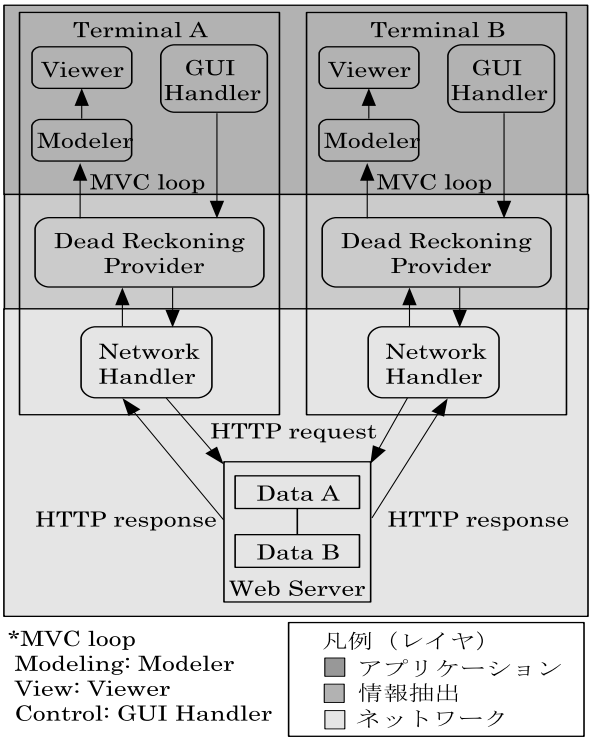

図 5.2: システムアーキテクチャ

5.2.3　システムコンポーネント

Web Server

Web サーバは，端末間の通信を中継する役割を担う．仮想空間の状態情報は，データサイズの軽減とパース時間の短縮化 [43] のため，XML ではなく JavaScript Object Notation（JSON）[17] 形式を利用する．

Web サーバは，HTTP リクエストの際に各端末からアバタ情報を受信し，そのレスポンスとして全端末の最新情報を連結したデータを各端末に送信する．

GUI Handler

GUI Handler は，ユーザの入力を DRP に送信する役割を担う．DRP で計算されたデータは，Modeler に提供される．

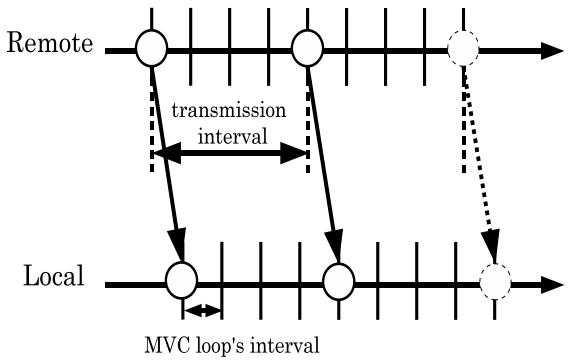

図 5.3: MVC ループとネットワーク伝送のタイムチャート

Modeler

Modeler は，DRP からのデータを受信し，そのデータを基に再構築した仮想空間を Viewer に提示する役割を担う．

Viewer

Viewer は，Modeler から提供された仮想空間の情報をユーザに提示する役割を担う．本研究では，Viewer として Web ブラウザを利用する．

Dead Reckoning Provider と Network Handler

DRP では，DR を利用し 5.1.2 で示された仮想空間に関する課題を解決する役割を担う．具体的には，ユーザ操作に関する情報を受信し，Modeler に適切な情報を提供する．また，上記の MVC ループとは非同期に Network Handler から遠隔アバタに関するデータを受信する．

ここで，Network Handler からの受信データは通信頻度の制限などにより，通信頻度の低下が発生し得る．これは，円滑なゲーム進行に悪影響を及ぼす．そこで，送信データにタイムスタンプを付加し，それを基に DR を利用し遠隔アバタのデータ補間を行う [68].

以上より，DRP の目的は，ゲーム開発者がネットワークの影響を考慮することなく，適切な情報を Modeler に提供することである．さらに，ゲーム開発者はスタンドアロンプログラムの作成と同じように，ネットワークを意識することなく簡単にリアルタイム Web ゲームを開発できる．また，Network Handler は，Ajax を利用し Web サーバとの通信を行う．

5.3 Dead Reckoning Protocol

本節では，DVE に関する要素技術をリアルタイム Web ゲームに適用することを目的とする．本研究室は既に DVE の要素技術として，独自の DR プロトコル [34] を提案している．PHBDR（Position History Based Dead Reckoning）[68] をはじめとする一般的な DR では，過去に受信した位置データを基に遠隔アバタの速度，加速度等を計算し，予測を行っている．それに対し，本研究室で提案している独自の DR プロトコルでは，ローカル端末においてサンプリングされた位置データから速度，加速度等を計算し，それらを位置データとともに遠隔端末に送信する方式を取っている．遠隔端末側では，受信したデータを基にアバタ位置の予測を行う．これにより，ローカル端末での微小なサンプリング間隔による予測精度の向上，クライアントでの自己アバタの履歴情報のみの保持，の利点が挙げられる．

DR プロトコルは，通信間隔の間の遠隔アバタ情報の補間を可能とする．したがって，端末間のデータ送受信間隔が HTTP 通信による制限を被ったとしても，ネットワークの影響を受けることなくユーザへのリアルタイムな情報提示が可能となると考えられる．そこで本節では，リアルタイム Web ゲームへの DR プロトコルの適用について説明する．図 5.4 は，図 5.2 における DR コンポーネントの詳細である．

図 5.4 において，各レイヤ（アプリケーション，情報抽出）は実線で示しており，各コンポーネント（DRP，GUI Handler，Modeler）は破線で示す．また，$A(t_i)$ と $A'(t_i)$ はそれぞれ，時刻 t_i におけるアバタ A の位置情報，及びその速度を表す．また，$A(t_{i+1})$ は時刻 t_{i+1} におけるアバタ A の予測位置を表す．図 5.4 に示すように，DR プロトコルは次の 3 つのステップからなる．

(1) ローカル端末でのサンプリング

まず，ローカル端末（図 5.4 では端末 A）はローカルにおいてサンプリング可能な短い時間間隔 ε で自己アバタのデータを記録する．このデータは，ローカル端末での履歴情報を更新するために利用される．また，サンプリングの際に利用するデータフォーマットを図 5.5a に示す．

(2) ローカル端末での速度の計算

次に，ローカル端末は履歴情報を基に差分を計算し，そのデータを更新間隔 u でリモート端末（図 5.4 では端末 B）に送信する．このときの送信データフォーマットを図 5.5b に示す．なお，t_i におけるアバタ A の速度は次式で定義される．

$$\frac{A(t_i) - A(t_{i-\varepsilon})}{\varepsilon} \tag{5.1}$$

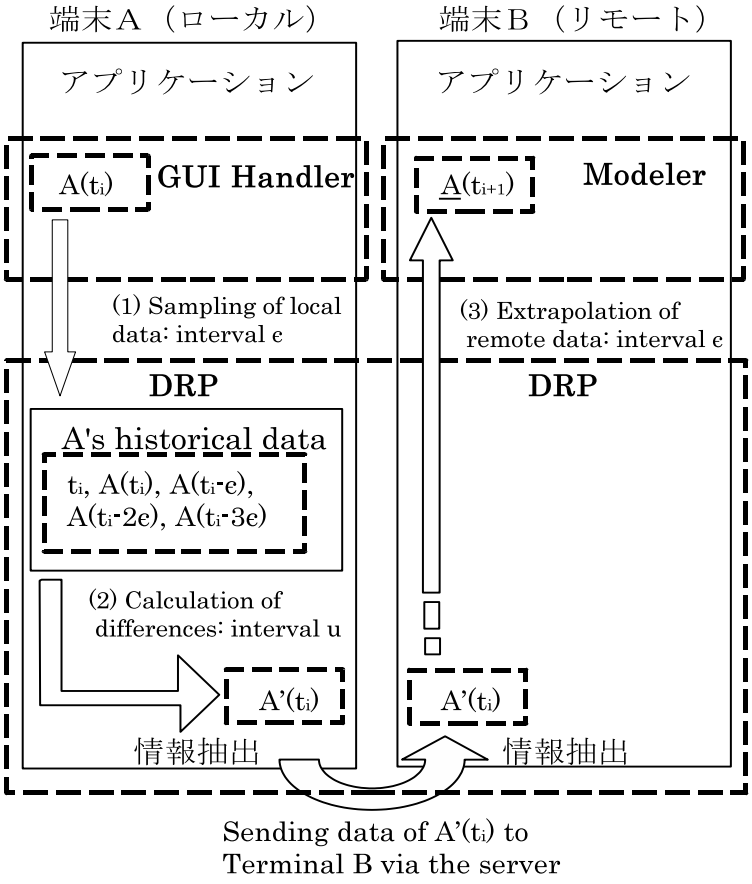

図 5.4: DR プロトコルの概要

(3) リモート端末でのローカルアバタのデータ補間

そして，リモート端末はローカルアバタのデータを更新間隔 ε で補間していく．ここで，$\varepsilon \ll u$ である．

5.4 評価実験

5.4.1 実験タスク

リアルタイム Web ゲームの実現性を評価するための実験タスクとして，DR プロトコルを導入したアバタ操作ゲームを試作した．本タスクでは，ユーザは Web サイトにログインすることで自己アバタを操作でき，Web ブラウザに自己アバタととも

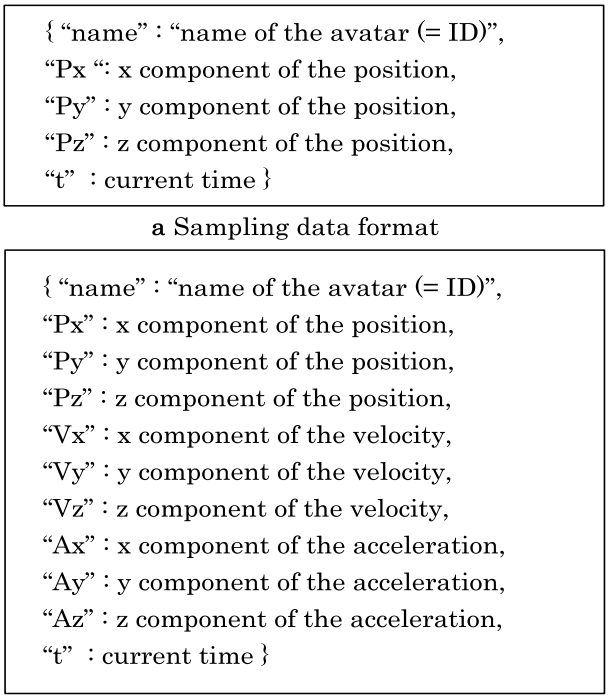

図 5.5: DR プロトコルで利用する JSON フォーマット

に相手アバタの様子が表示される．図 5.6 にシステム構成図，図 5.7 に実行画面を示す．図 5.6 では，各クライアントは独自に MVC ループを実行し，自己アバタの位置，並びにその差分情報をサーバにアップロードする．サーバ側では，各クライアントの最新情報を保持し，クライアントへのレスポンスとして返信する．

また，Web ゲームにおける仮想空間，及びシステムの設定条件を以下に示す．

- 仮想空間
  - 仮想空間の大きさ：無制限
  - アバタの大きさ：32×32[pixel]
  - アバタ速度の上限：64[pixel/s]
- システム
  - Web サーバとの通信：HTTP
- Web ブラウザ

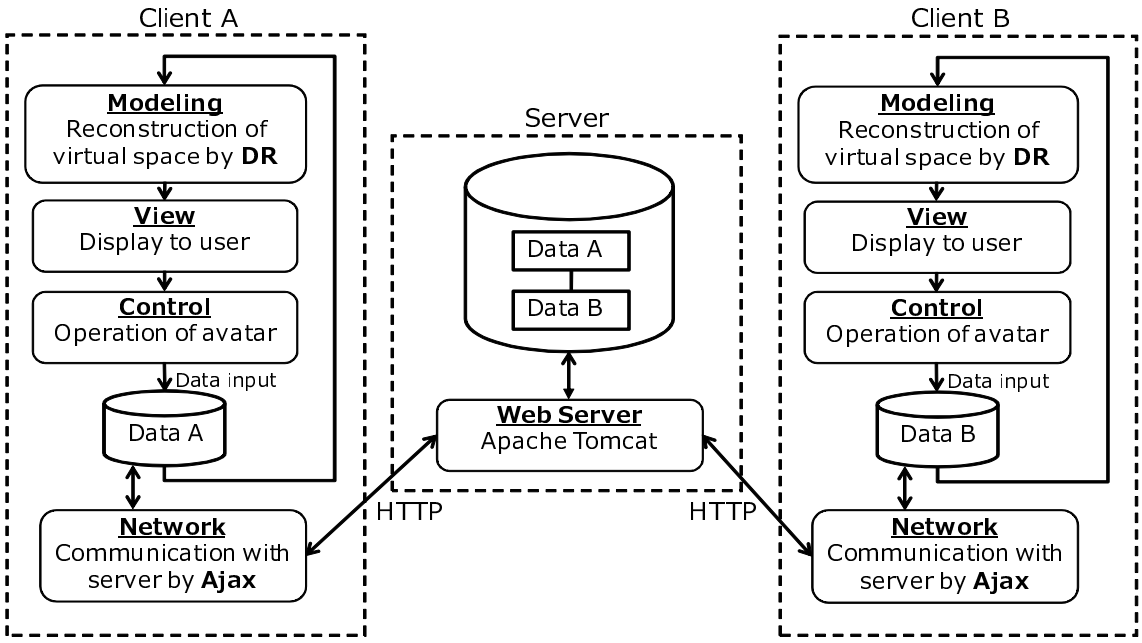

図 5.6: システム構成図

– Firefox 2.0.0.14

5.4.2 実験の目的と条件

実験の目的は，リアルタイム Web ゲームの実現性を評価することである．具体的には，HTTP 通信やサーバとの通信頻度の影響により，端末間で提示されるアバタ位置情報の差異を評価する．また，DR プロトコルを利用することでそれらの影響がどの程度軽減されるかについても評価する．

さらに，Web ブラウザ上で動作する本システムにおいて，MVC ループの間隔やサーバとの通信が一定の周期で実行されるかについても評価する．

ここで，本実験の評価項目は，

(1) アバタ位置情報の差異

(2) MVC ループの間隔

(3) Web サーバとの通信間隔

(4) 相手アバタの更新頻度

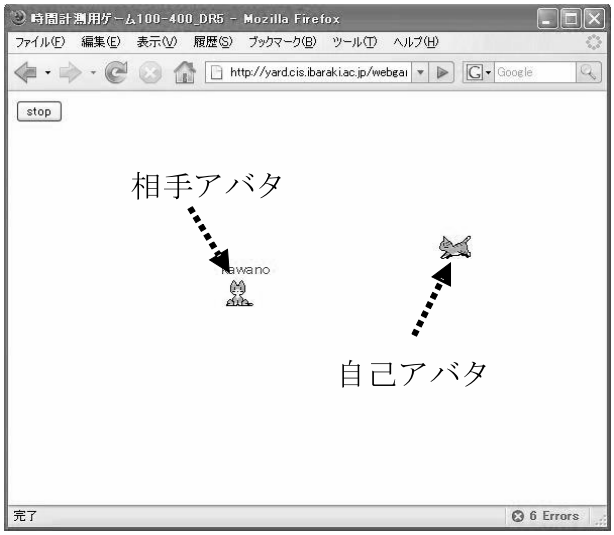

図 5.7: Web ゲームの実行画面

(5) Web サーバとの通信遅延

(6) クライアント間のデータ伝送遅延

である．まず (1) は，MVC ループのタイミングで更新される各アバタの位置情報を比較することで評価する．本実験における整合性の要件は，タスクの種類とアバタの大きさを考慮して次式で定義する．

$$|A_A(t_i) - A_B(t_i)| + |B_A(t_i) - B_B(t_i)| \leq 32 \tag{5.2}$$

ただし上式の各項は各端末に提示されるアバタの状態（単位：pixel）を示し，その添え字は提示される端末，括弧内は時刻を表す．すなわち，端末 A，B におけるアバタ A の位置情報の差異（左辺第一項）と，アバタ B のそれ（左辺第二項）の合計が 32[pixel]（アバタキャラクタのピクセルサイズ）以下であることを整合性の要件とする．本方式では，前提条件としてサーバの機能はメッセージ中継のみとするため，アバタ同士の接触判定は各端末上で行う必要がある．そのため，(1) が上式を満足できない場合，端末間での接触判定に不整合が生じゲームが破綻する可能性がある．次に (2) は，設定した MVC ループの間隔と実際の MVC ループの間隔を比較し評価する．同様に (3) は，設定した通信間隔と実際の通信間隔を比較し調査する．(4) は，その際に相手アバタの情報が更新されているかを調査することで評価する．(5) は，サーバとの通信に要した時間を評価する．そして (6) は，クライアントがサーバに自己アバタのデータをアップロードした時刻と，もう一方のクライアン

トがサーバから相手アバタのデータをダウンロードした時刻の差を調査する．この項目は，クライアント間のデータ伝送遅延といえる．

上記の評価項目を調査するための実験を行った．本実験では，実際のインターネットを利用し，都内のデータセンタ内に設置したWebサーバと都内のプロバイダを介して通信を行った．実験では，比較的短時間で終了し，タスクの内容が分かり易く，かつ更新間隔の影響を受け易い単純なタスクとして，1対1のアバタ追従タスクを採用した．ここで，片方の被験者には「アバタを周期的に左右に移動させるのみ」という単純なタスクを指示し，もう一方の被験者には相手アバタの追従を指示した．これは，できる限り人為的な影響を排除し，DRの純粋な効果を測るためである．実験パターンを以下に示す．

- DRプロトコルの有無
- MVCループの間隔：50，100[ms]
- サーバとの通信間隔：100 - 500[ms]（刻み幅は，100[ms]間隔）

上記の組み合わせ20パターンに対して，各パターンにつき5回試行した．また，各試行の実行時間は1分間とした．なお，本実験では，DRプロトコルの最も基本的な予測モデルとして，等速度運動を仮定した予測モデルを利用した．これにより，リアルタイムWebゲームにおけるDRの基本的特徴を検証する．実験結果を図5.8-5.10，表5.1-5.6に示す．

5.4.3 考察

図5.8より，DRを利用しない場合，通信間隔の増加に伴い，アバタ位置情報の差異が増加している．特に，通信間隔が300[ms]以上の場合，1アバタあたりの位置情報の差異が20[pixel]以上となるため，式5.2の要件を満足できない．その一方で，DRを利用した場合，アバタ位置情報の差異は2.8〜8.0[pixel]の低い値で安定しており，通信間隔が増加しても式5.2の要件を満足していた．ここで，本システムでは，Webサーバへのデータのアップ／ダウンロードによりクライアント間のデータ伝送を実現している．そのため，Webサーバへの通信間隔の増加によって現れた，アバタ位置情報の端末間での差異は，DRを利用することにより軽減できている．

表5.1，5.2より，いずれの場合においても，設定した間隔と同程度の平均的時間間隔でMVCループが実行されている．ただし，その変動は大きく，特に通信間隔が短いほど大きい．この問題は，Webブラウザにおける JacaScriptの実装による．具体的には，本実験で利用しているFirefoxのタイマ処理に関する内部動作に起因する．本システムでは，JavaScriptのsetInterval関数によりMVCループの間隔とWeb

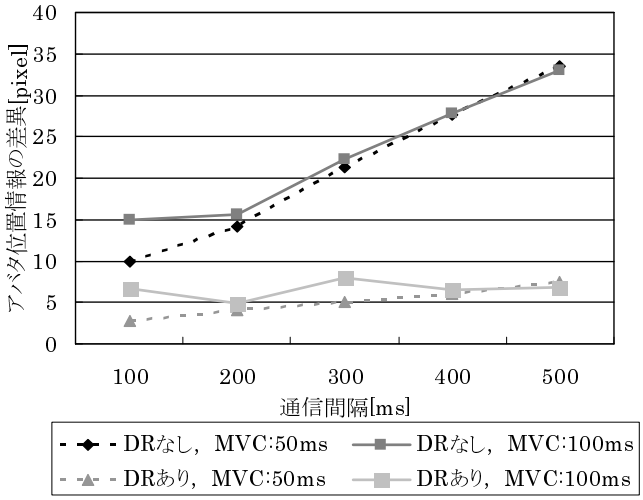

図 5.8: アバタ位置情報の差異（単位：pixel）

表 5.1: MVC ループの間隔（設定：50ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	53.1	52.5	52.4	52.2	52.1
標準偏差 [ms]	47.1	24.0	21.9	20.0	18.9

サーバとの通信間隔を指定しており，それらの命令はすべて同じイベントキューで管理される．そのため，他に負荷のある処理が発生すればそれらの実行までにより多くの時間を要する．その一方で，setInterval 関数では，指定された命令を指定時間後にイベントキューに登録するだけである．よって，前回の命令が実行されるまでに多くの時間を要した場合，次回の命令は即座に実行される傾向があるため，イベントの定期実行が不安定となる．この影響は，Web ブラウザの実装によっては軽減されるが，JavaScript がシングルスレッドで動作する言語であるため，完全には回避できない．すなわち，JavaScript の言語仕様が抱える本質的な問題である．そ

表 5.2: MVC ループの間隔（設定：100ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	102.0	101.8	101.5	101.2	101.0
標準偏差 [ms]	91.1	58.4	48.4	43.1	40.9

表 5.3: Web サーバとの通信間隔（MVC ループ：50ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	104.8	200.1	300.0	399.9	499.8
標準偏差 [ms]	38.0	88.8	124.0	146.7	179.9

表 5.4: Web サーバとの通信間隔（MVC ループ：100ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	101.9	200.0	300.0	400.1	500.0
標準偏差 [ms]	90.8	58.0	78.8	54.8	42.3

のため，DR の利用により，更新処理の突発的な発生に対してより正確な状態提示による，ユーザの違和感の払拭が非常に有効となる．

次に，表 5.3，5.4 より，Web サーバとの通信については，設定した間隔に従って実行されている．しかしながら，MVC ループを 50[ms] に設定した場合，その変動は大きい．これは，上記と同様にタイマ処理の仕様に起因する．ただし，通信間隔を 100[ms] とした場合のみ，MVC ループ間隔を 100[ms] に設定した方が変動が増加した．これは，MVC ループと通信を同じ間隔に設定したことにより，同じタイミングでイベントキューに登録され両者の処理がほぼ同期するためである．なお，これらは MVC ループと同様に DR の有無による差異はなかったため，DR の有無に関わらず集計した．これは本システムでの DR の導入による計算コストは軽微であり，ゲームへの影響はほとんど発生しなかったことを裏付ける．

図 5.9，5.10 より， DR なしの場合では，通信間隔の増加に伴い相手アバタの更新頻度が低下している．そこで，DR を利用することにより通信間隔に関わらず，相手アバタの更新が可能となる．ここで，図 5.10 の DR なしの場合において，通信間隔 100[ms] と 200[ms] を比較すると，両者の間で相手アバタの更新頻度にあまり差は見られなかった．これは，通信間隔の変動が大きく，短い時間間隔で自己アバタのデータを Web サーバにアップロードした場合，他の端末がダウンロードする前に前回のデータを更新してしまう可能性が高いためである．逆に，短い時間間隔で更新した場合，次回の更新まではより多くの時間を要する場合が多く，このため実質の更新頻度は 200[ms] と同程度になってしまう．以上の議論より，Ajax を用いたリアルタイム Web ゲームを構築する場合，通信間隔は MVC ループの間隔より大きく取るべきである．これは，リアルタイム Web ゲームにおける DR の有用性を示す一因といえる．

ここで，Web サーバとの通信遅延は，いずれの場合においても大きな差異はなく，

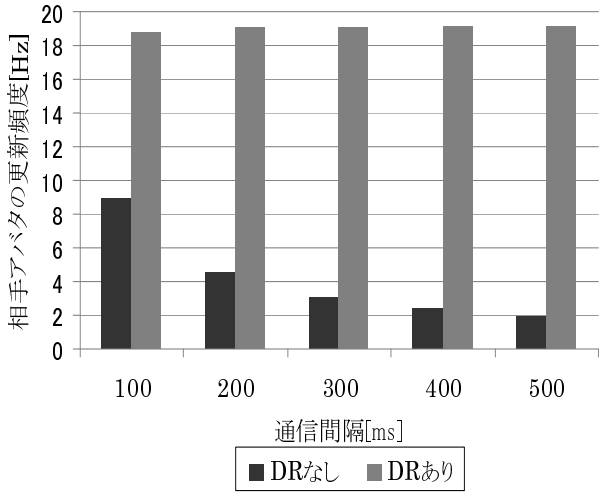

図 5.9: 相手アバタの更新頻度（MVC ループ：50ms，単位：Hz）

平均で 68.8[ms]，標準偏差は 21.3[ms] であった．本実験では，Web サーバと各クライアントの通信はインターネットを介している．しかしながら，ユーザ数は 2 名と少ないため，HTTP 通信のオーバーヘッドが影響するものではなかった．

ここで，ユーザ数が増加した場合について考察する．ユーザ数の増加に伴い，サーバ，クライアント，ネットワーク，それぞれの負荷が増大し，いずれかがボトルネックとなりゲームが破綻する可能性がある．そこでまず，サーバに関しては，クライアントからのリクエスト数，保持データ量が $O(n)$ で増加する（n はユーザ数）．次に，ネットワークではトラフィック量が $O(n)$ で増加する．ただし，サーバの機能はクライアント間のメッセージ仲介のみであり，送受信データフォーマットは軽量な JSON を利用しているため，サーバ，及びネットワークがボトルネックとなる可能性は低い．一方，クライアント側では，受信データ量と描画アバタ数の各々が $O(n)$ で増加する．これは，Web ブラウザ上で動作するプログラムにとっては大きな負担となる．さらに，アバタ同士の相互作用の処理（接触判定等）を行う場合，その処理負荷は $O(n^2)$ で増加する．以上の議論により，ユーザ数の増加に伴いクライアントの負荷がボトルネックとなる可能性が高いと予想される．

今後はユーザ数や，利用回線の帯域などを考慮し，サーバ，クライアントの負荷，HTTP 通信によるオーバーヘッドと，それらがリアルタイム Web ゲームに与える影響について調査する必要がある．

最後に，表 5.5，5.6 より，Web サーバとの通信間隔の増加に伴いクライアント間のデータ伝送遅延が増加している．また，通信間隔 100[ms] の場合のみ，MVC ループ間隔 50[ms] の方がデータ伝送遅延は小さくなっているのに対し，それ以外の場合

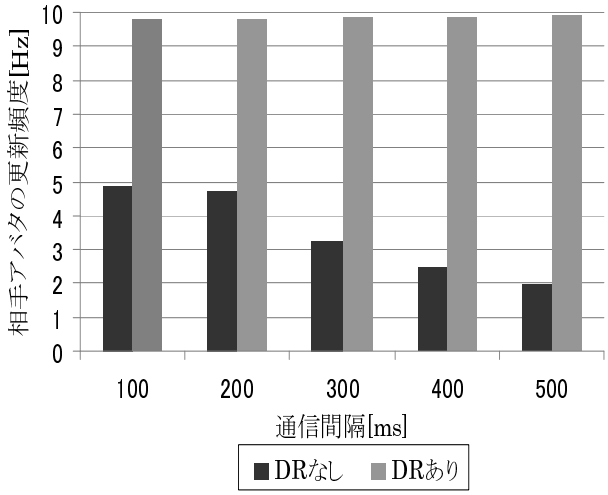

図 5.10: 相手アバタの更新頻度（MVC ループ：100ms，単位：Hz）

表 5.5: クライアント間のデータ伝送遅延（MVC ループ：50ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	120.7	181.8	239.7	297.0	348.7
標準偏差 [ms]	40.6	71.5	90.6	101.1	120.9

では大きな差は見られない．このように，通信間隔がクライアント間のデータ伝送遅延に影響を及ぼすため，遅延の程度とアプリケーションの因果律（ゲームの勝敗など）に破綻を来たさないための条件，並びに DR の適用範囲の考察を個別に行う必要がある．

5.5 まとめ

本章では，Non-Lockstep 型 S/C において，1.2 節の課題 1 の通信頻度に着目し，リアルタイム Web ゲームの実現可能性について検討した．具体的には，リアルタイム Web ゲームを実現するための設計について考察し DR の有効性について評価した．そこで，その設計方法として階層構造モデルを採用し，情報抽出レイヤ内でのネットワーク影響の軽減を試みた．また，HTTP 通信による制限を克服するため，Ajax を利用しサーバとの通信と MVC ループの処理を分離した．さらに，MVC ループ内

表 5.6: クライアント間のデータ伝送遅延（MVC ループ：100ms，DR の有無に関わらず集計）

通信間隔 [ms]	100	200	300	400	500
平均 [ms]	165.6	169.9	231.3	277.5	338.6
標準偏差 [ms]	57.3	49.8	66.7	51.1	46.2

で DR を行うことで，通信頻度が低い環境においても，実時間での相手アバタの更新が可能となった．評価実験により，Web ブラウザにおける更新頻度の変動が大きいことを確認した．また，DR が効果的に機能すれば，リアルタイム Web ゲームにおける整合性，応答性の両方の要件を満たせることが判明した．ただし，本評価はユーザ数 2 名での実験結果のみであり，ユーザ数の増加に伴う影響，及びその範囲については今後の評価実験が必要である．

最後に，本章で対象としたリアルタイム Web ゲームは，インターネット上の不特定多数のユーザが参加可能なゲーム空間であり，サイバーワールドを構成する要素であると捉えることができる．したがって，本章で提案したゲームとプロトコル基盤はサイバーワールドの発展に少なからず寄与するであろう．

第6章　結論

6.1　総括

本研究では，DVE の設計方法と整合性・応答性の関連性に着目し，リアルタイム型 DVE の実現可能性について検討した．具体的には，DVE の技術的課題に関して，通信性能が影響する課題のうち，通信頻度，分散インタラクションを主な研究対象とした．さらに，本研究の立場としては，通信遅延の軽減を試みるのではなく，その影響をユーザに感じさせないことを目標とし，各端末が互いに協調することで通信遅延，通信頻度の影響の軽減を試みた．

そこでまず，DVE のモデルをトポロジと役割の観点から 4 種類に分類し，そのうちリアルタイム性の高い Non-Lockstep 型 DVE の 2 種類のモデルに関して検討した．

最初に，Non-Lockstep 型 P2P に関して，分散インタラクションの課題に着目し，応答性の良さをある程度維持したまま，仮想空間の整合性を維持する共有オブジェクトの管理手法を提案した．その結果，Non-Lockstep 型 DVE において，共有オブジェクトの排他制御が可能となる方式を，国内外を通じて初めて提案した．

第二に，Non-Lockstep 型 S/C に関して，通信頻度の課題に着目し，ユーザが Web ブラウザを通して簡単に利用できるリアルタイム Web ゲームの実現可能性について検討した．その結果，Web 上でのリアルタイム型 DVE は，Ajax，DR などの効果的な利用により実現可能であることを確認した．

上記の結果は，リアルタイム型 DVE の実現可能性を示唆するものである．このことより，本研究の成果は，今後のリアルタイム型 DVE の設計方法に少なからず寄与するであろう．

6.2　今後の展望

本研究では，リアルタイム型 DVE の実現可能性について検討し，一定の成果を得ることができた．しかしながら，本研究で提案した分散インタラクションを実現する手法では，通信遅延が一定であることを前提としており，ジッタを考慮した手法の検討には至らなかった．今後は，ジッタに関する定量評価が必要である．また，

ゲームの公平性を保証するための手法の検討，インターネット上での実運用に向けた課題の検討，並びに AtoZ の提要範囲の検討も必要である．

続いて，リアルタイム Web ゲームに関しては，DR の厳密な評価，予測誤差の揺れを軽減するためのコンバージェンスの組み込み，ユーザ数，ネットワーク環境による HTTP 通信への影響の調査が必要である．

また，上記研究のさらなる展望として，ゲーム種類と遅延・更新間隔の関連性について調査を行いたい．具体的には，ゲームの種類に応じた遅延と更新間隔の許容限界を明らかにすることで，例えば，ゲーム特性に応じた最適な DVE モデルの選択，並びにゲーム状況に応じた適切な通信頻度の動的変更などを試みる．これらのケーススタディによって得られた知見をまとめ，オンラインゲームの設計における包括的なガイドラインの策定を目指す．

最後に，本論文の対象とは異なる発展例として，筆者らはサッカーやホッケーなどの集団仮想球技における戦略的パスアルゴリズムを提案している [64], [112], [113]. これらの研究では，チームメイトが協調しながらチーム全体としてボールを優位に扱える状態に導くことを目的とし，スペースの概念を定量化することでパスの成否について検討している．今後は，インターネット上でのチームの協調動作を実現するためのプロトコルの検討が課題として挙げられる．

以上の点に関して，今後さらなる調査・研究を進めて行きたい．

謝辞

最初に，博士課程全般に渡って，御指導，御鞭撻を頂きました，本学工学部情報工学科米倉達広教授に深く感謝致します．本研究に関する議論，本論文の執筆，研究者としての心構えなど，様々な面での御指導を頂きました．今後益々御活躍されることをお祈り申し上げます．

続いて，本研究並びに本論文に執筆に関して多くの御助言を頂きました，本学工学部情報工学科加納幹雄教授，鎌田賢教授，澁澤進准教授，羽渕裕真准教授に感謝致します．また，成蹊大学の岡本秀輔准教授，塙大助教に感謝致します．米倉研究室の皆様には，ディスカッションでの様々な議論，及び被験者実験で本研究に御協力頂きました．特に，大部由香さんには，筆者が本研究室に配属されてからの 6 年間，多大なる時間をあらゆる事柄に関してともにディスカッションさせて頂きました．これらディスカッションでのご指摘は，研究の方針から提案した各方式の詳細に至るまで，本研究の成果に大きな影響を与えて下さいました．ここに感謝致します．

また，筆者の博士課程への進学に際し，仕事との両立を快く御了承下さいました，株式会社インテック・ネットコア代表取締役社長荒野高志様に感謝致します．さらに，本研究に関しまして荒野社長を含め同社社員の方々には，ネットワーク関連の御指摘など本研究室とは異なる視点から様々な御意見を頂きました．これにより，本研究の目的を常に意識することができ，より充実した研究成果を得ることができました．ここに感謝致します．

最後に，筆者を支えて下さいました家族に感謝致します．

参考文献

[1] Adobe Systems Incorporated. “Adobe Flash”, 2008, http://www.adobe.com/.

[2] Ajaxian.com. “Ajaxian”, http://ajaxian.com/.

[3] M. Bassiouni:“Data Compression in scientific and statistical databases”, *Software Engineering*, Vol.11, No.10, pp.1047–1058, 1985.

[4] M. Bassiouni, H. Williams, M. Loper, “Intelligent filtering algorithms for real-time networked simulators”, In *Proceedings of IEEE Conference on System, Man and Cybernetics*, pp. 309–314, 1991.

[5] E.J. Berglund and D.R. Cheriton:“Amaze: a multiplayer computer game”, *IEEE Software*, Vol.2, No.3, pp.30–39, 1985.

[6] Eric J. Berglund and David R. Cheriton:“Amaze: A multiplayer computer game”, *IEEE Software*, Vol.2, No.3, pp.30–39, 1985.

[7] Y.W. Bernier, “Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization”, In *In Proceedings of Game Developers Conference 2001*, p.2001.

[8] P. Bettner and M. Terrano, “1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond”, In *In Proc. of GDC 2001*, p.2001.

[9] Blizzard Entertainment. “Diablo II”, 2000, http://www.blizzard.com/us/diablo2/.

[10] Blizzard Entertainment. “StarCraft II”, 2008, http://www.starcraft2.com/.

[11] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms. “Explicit Multicast (Xcast) Concepts and Options”. RFC 5058 (Experimental), November 2007.

[12] Jean-Sébastien Boulanger, Jörg Kienzle, Clark Verbrugge, “Comparing interest management algorithms for massively multiplayer games”, In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, p. 6, Singapore, 2006.

[13] T.K. Capin, J. Esmerado, D. Thalmann:“A dead-reckoning technique for streaming virtual human animation”, *IEEE Trans. Circ. Syst. Video Technol.*, Vol.9, No.3, pp.411–414, 1999.

[14] C. Carlsson and O. Hagsand:“DIVE-A platform for multi-user virtual environments”, *Computer Graphics*, Vol.17, No.6, pp.663–669, 1993.

[15] M. Carson and D. Santay, “NIST Net - A Linux based Network Emulation Tool”, In *ACM SIGCOMM CCR*, 33, pp. 111–126, 2003.

[16] Cometd.org. “Cometd Project”, http://cometdproject.dojotoolkit.org/.

[17] D. Crockford. “The application/json Media Type for JavaScript Object Notation (JSON)”. RFC 4627 (Informational), July 2006.

[18] E. Cronin, B. Filstrup, S. Jamin, “Cheat-proofing dead reckoned multiplayer games”, In *Proc. of 2nd Int. Conf. on ADCOG*, p.2003.

[19] E. Cronin, B. Filstrup, A.R. Kurc, S. Jamin, “An Efficient Synchronization Mechanism for Mirrored Game Architectures”, In *In Proc. of NetGames 2002*, pp. 67–73, 2002.

[20] CyberAgent, Inc. “アメーバブログ”, 1998-2008, http://ameblo.jp/.

[21] Jauvane C. de Oliveira and Nicolas D. Georganas:“VELVET: An Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs”, *Presence: Teleoperators and Virtual Environments*, Vol.12, No.6, pp.555–580, 2003.

[22] D. Delaney, T. Ward, S. McLoone:“On Consistency and Network Latency in Distributed Interactive Applications: A Survey-Part I”, *Presence*, Vol.15, No.2, pp.218–234, 2006.

[23] D. Delaney, T. Ward, S. McLoone:“On Consistency and Network Latency in Distributed Interactive Applications: A Survey-Part II”, *Presence*, Vol.15, No.4, pp.465–482, 2006.

[24] C. Diot and L. Gautier, “A distributed architecture for multiplayer interactive applications on the internet”, In *IEEE Proc. Multimedia Systems Conference,*, pp. 6–15, 1999.

[25] Electronic Arts Inc. “Ultima Online”, 2000, http://www.uoherald.com/.

[26] Electronic Arts Inc. “Command & Conquer”, 2008, http://www.commandandconquer.com/.

[27] Epic Games, Inc. “Unreal Tournament 3”, 2007, http://www.unrealtournament3.com/.

[28] T.A. Funkhouser, “Ring: A client-server system for multi-user virtual environments”, In *Symposium on Interactive 3D Graphics*, pp. 85–92, 1995.

[29] T.A. Funkhouser:“RING: A client-server system for multiuser virtual environments”, *Symposium on Interactive 3D Graphics*, pp.85–92, 1995.

[30] Google. “Gmail”, 2008, http://mail.google.com/.

[31] Google. “Google Maps”, 2008, http://maps.google.com/.

[32] GungHo Online Entertainment, Inc. “ラグナロクオンライン”, 2002-2008, http://www.ragnarokonline.jp/.

[33] Olof Hagsand:“Interactive multiuser VEs in the DIVE system”, *IEEE MultiMedia*, Vol.3, No.1, pp.30–39, 1996.

[34] D. Hanawa and T. Yonekura:“Improvement on the accuracy of the polynomial form extrapolation model in distributed virtual environment”, *Springer-Verlag, Visual Comput*, Vol.23, No.5, pp.369–379, 2007.

[35] Kazuki Hiraki, Shintaro Kawahara, Tatsuhiro Yonekura, ““Web-Com”: Web-browser-for-communication; aiming for web-based class”, In *AACE Proceedings of ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pp. 281–286, Luango, Switzerland, 2004.

[36] Kazuki Hiraki, Tatsuhiro Yonekura, Shintaro Kawahara, Susumu Shibusawa, “Development of the “Web-Com” interactive browser for web-based class”, In *IEEE Proceedings of 2004 International Conference on Cyberworlds*, pp. 343–350, Tokyo, Japan, 2004.

[37] Kazuki Hiraki, Tatsuhiro Yonekura, Susumu Shibusawa, “An evaluation of Web-Com: Web-based education system”, In *IEEE Proceedings of 2004 International Conference on Cyberworlds*, pp. 229–236, Singapore, 2005.

[38] Kazuki Hiraki, Tatsuhiro Yonekura, Susumu Shibusawa:““Web-Com”: Interactive browser for web-based education”, *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, Vol.E88–D, No.5, pp.912–918, 2005.

[39] D.J.V. Hook, M. Newton, D. Fusco:“An approach to DIS scalability”, *Proceedings of the 11th Workshop in Standards for the Interoperability of Distributed Simulation*, pp.347–355, 1994.

[40] id Software, Inc. “Quake IV”, 2005, http://www.quake4game.com/.

[41] IEEE:“IEEE standard for distributed interactive simulation - Application protocols”, *Proceedings of IEEE Standard 1278-1993*, 1993.

[42] Jagex Ltd. “RuneScape”, 2007, http://www.runescape.com/.

[43] James Ward. “Census - RIA Data Loading Benchmarks”, 2007, http://www.jamesward.org/census/.

[44] D.R. Jefferson:“Virtual time”, *ACM Transactions on Programming Languages and Systems*, Vol.7, No.3, pp.404–425, 1985.

[45] John Resig and jQuery Team. “jQuery”, http://jquery.com/.

[46] Yoshihiro Kawano, Masahiro Miyata, Dai Hanawa, Tatsuhiro Yonekura, “A New Concept for Real-Time Web Games - Developing Highly Real-Time Web Games”, In *4th International Conference on Web Information Systems and Technologies*, pp. 171–177, Madeira, Portugal, 2008.

[47] Yoshihiro Kawano and Tatsuhiro Yonekura, “On a Serverless Networked Virtual Ball Game for Multi-Player”, In *IEEE Proc. 2005 Int. Conf. on Cyberworlds*, pp. 256–260, Singapore, 2005.

[48] Yoshihiro Kawano and Tatsuhiro Yonekura, “Count Down Protocol: Asynchronous Consistent Protocol in P2P Virtual Ball Game”, In *NetGames ’06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, p. 32, Singapore, 2006.

[49] Laszlo Systems, Inc. “OpenLaszlo”, 2005-2008, http://www.openlaszlo.org/.

[50] Y.-B. Lin and E.D. Lazowska:“A study of time warp rollback mechanisms”, *ACM Transactions on Modelling and Computer Simulation*, Vol.1, No.1, pp.51–72, 1991.

[51] Linden Research, Inc. “Second Life”, 2003-2008, http://jp.secondlife.com/.

[52] J.M. Linebarger and G.D. Kessler:“Concurrency Control Mechanisms for Closely Coupled Collaboration in Multithreaded Peer-to-Peer Virtual Environments”, *Presence*, Vol.13, No.3, pp.296–314, 2004.

[53] livedoor Co.,Ltd. “livedoor Blog”, 1996-2008, http://blog.livedoor.com/.

[54] M.R. Macedonia, M.J. Zyda, D.R. Pratt, P.T. Barham, S. Zeswitz:“NPSNET: A Network Software Architecture for Large Scale Virtual Environments”, *Presence*, Vol.3, No.4, pp.265–287, 1994.

[55] Maid Marian Entertainment. “Tank Ball2”, 2004, http://maidmarian.com/Tank.htm.

[56] Microsoft Corporation. “Age of Empires III”, 2007, http://www.ageofempires3.com/.

[57] Microsoft Corporation. “Microsoft Silverlight”, 2008, http://www.microsoft.com/japan/silverlight/.

[58] mixi, Inc. “ソーシャル・ネットワーキングサービス [mixi（ミクシィ）]”, 1999-2008, http://mixi.jp/.

[59] Motion-Twin. “ディノパライソ”, 2005, http://www.smashbros.com/jp/.

[60] NCsoft Corporation. “リネージュ II”, 2004, http://lineage2.plaync.jp/.

[61] Yuka Obu, Tomoyuki Kato, Tatsuhiro Yonekura, “M.A.S.: A protocol for a musical session in a sound field where synchronization between musical notes is not guaranteed”, In *ICMA Proceedings of the 2003 Internatioal Computer Music Conference*, pp. 309–313, Singapore, 2003.

[62] S. Okamoto, M. Kamada, T. Yonekura:“Prototyping Tool for Web-based Multiuser Online Role-Playing Game”, *IEICE TRANS. INF. & SYST.*, Vol.E91-D, No.6, pp.1700–1703, 2008.

[63] OpenAjax Alliance, http://www.openajax.org/index.php.

[64] Kazufumi Osato, Tatsuhiro Yonekura, Yoshihiro Kawano, Dai Hanawa, “On an analysis of pass play in a virtual ball game”, In *Proceedings of the IEEE 2007 International Conference on Cyberworlds (CW2007)*, pp. 35–40, Hannover, 2007.

[65] I.S. Pandzic, E. Lee, N.M. Thalmann, T.K. Capin, D. Thalmann:“A flexible architecture for virtual humans in networked collaborative virtual environments”, *Computer Graphics Forum*, Vol.16, No.3, pp.177–188, 1997.

[66] Prototype Core Team. “Prototype JavaScript framework”, 2006-2007, http://cometdproject.dojotoolkit.org/.

[67] X. Qin:“Delayed consistency model for distributed interactive systems with real-time continuous media”, *Journal of Software*, Vol.13, No.6, pp.1029–1039, 2002.

[68] S.K. Singhal, “Effective remote modeling in large-scale distributed simulation and visualization environments”, PhD thesis, Department of Computer Science, Stanford University, Palo Alto, 1996.

[69] S.K. Singhal and M.J. Zyda, *Networked Virtual Environments: Design and Implementation*. Addison-Wesley, Massachusetts, 1999.

[70] Jouni Smed, Henrik Niinisalo, Harri Hakonen, “Realizing bullet time effect in multiplayer games with local perception filters”, In *NetGames ’04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pp. 121–128, Portland, Oregon, USA, 2004.

[71] Sony Online Entertainment LLC. “Everquest II”, 2005, http://everquest2.station.sony.com/.

[72] Valve Corporation. “Counter-Strike”, 2007, http://www.steamgames.com/v/index.php?area=gameAppId=240.

[73] Valve Software. “Half-Life 2”, 2005, http://orange.half-life2.com/.

[74] Wikimedia Foundation Inc. “Wikipedia”, 2001-2008, http://ja.wikipedia.org/.

[75] Wikipedia. “MORPG”, 2008, http://ja.wikipedia.org/wiki/MORPG.

[76] Yahoo! Inc. “Flickr”, 2002-2008, http://www.flickr.com/.

[77] Syunnosuke Yamakawa and Tatsuhiro Yonekura, “On a dynamic caching method for field segmented dve by multi-server”, In *NetGames ’06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, p. 43, Singapore, 2006. ACM.

[78] Tatsuhiro Yonekura and Yoshihiro Kawano:“A Protocol for Peer-to-peer Multi-Player Networked Virtual Ball Game”, *IEICE Trans. Inf. & Syst.*, Vol.E88-D, No.5, pp.926–937, 2005.

[79] Tatsuhiro Yonekura, Yoshihiro Kawano, Dai Hanawa, “Peer-to-peer Networked Field-type Virtual Environment by Using AtoZ”, In *IEEE Proceedings of 2004 International Conference on Cyberworlds*, pp. 241–248, Tokyo, Japan, 2004.

[80] Masahiro Yoshida, Yuka Obu, Tatsuhiro Yonekura, “A protocol for remote musical session with fluctuated tempo”, In *IEEE Proceedings of 2004 International Conference on Cyberworlds*, pp. 87–93, Tokyo, Japan, 2004.

[81] Masahiro Yoshida, Yuka Obu, Tatsuhiro Yonekura:“A protocol for real-time remote musical session”, *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, Vol.E88–D, No.5, pp.919–925, 2005.

[82] YouTube, LLC. “YouTube”, 2005-2008, http://jp.youtube.com/.

[83] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala:“RSVP: A New Resource ReSerVation Protocol”, *IEEE Communications Magazine, 14(5)*, pp.116–127, 2002.

[84] (株) コーエー. “真・三國無双 Online”, 2007, http://www.musou-bb.jp/.

[85] マイスペース (株). “MySpace”, 1996-2008, http://jp.myspace.com/.

[86] 引地謙治, 森野祐直, 福田一郎, 松本壮樹, 瀬崎薫, 安田靖彦:“触覚を含む仮想空間共有システムの提案と評価”, **電子情報通信学会論文誌** *B*, Vol.J86-B, No.2, pp.268–278, 2003.

[87] 任天堂 (株). “大乱闘スマッシュブラザーズ X”, 2007, http://www.smashbros.com/jp/.

[88] (株) カカクコム. “価格.com”, 1997-2008, http://kakaku.com/.

[89] (株) ココア. “meet-me”, 2007-2008, http://www.meet-me.jp/.

[90] (株) スクウェア・エニックス. “ドラゴンクエスト公式サイト”, 2001-2007, http://www.square-enix.co.jp/dragonquest/.

[91] (株) スクウェア・エニックス. “ファイナルファンタジー XI”, 2002-2008, http://www.playonline.com/ff11/index.shtml.

[92] (株) スクウェア・エニックス. “ファイナルファンタジー公式サイト”, 2008, http://www.square-enix.com/jp/title/finalfantasy/.

[93] (株) スクウェア・エニックス. “ドラゴンクエスト IX”, 2009, http://www.level5.co.jp/products/dq9/.

[94] (株) ニワンゴ. “ニコニコ動画”, 2005-2008, http://www.nicovideo.jp/.

[95] 河野義広, 宮田昌廣, 塙大, 米倉達広:“Dead Reckoning を用いたリアルタイム Web ゲームの設計と評価”, **電子情報通信学会論文誌** *D*, Vol.J91-D, No.12, pp.–, 2008. （印刷中）.

[96] 河野義広, 塙大, 米倉達広, “相補予測機能を用いたネットワーク対戦型エアホッケーの試作”, **日本バーチャルリアリティ学会大会論文集**, 第 8 巻, pp. 403–406, 2003.

[97] 河野義広, 塙大, 米倉達広, “AtoZ を用いた P2P 型ネットワーク対戦球技におけるクリティカル・ケースの評価”, 技術研究報告 MVE2003–125, 電子情報通信学会, 2004.

[98] 河野義広, 塙大, 米倉達広:“相補予測と AtoZ(Allocated Topographical Zone) による P2P 型ネットワーク仮想球技へのアプローチ”, **日本バーチャルリアリティ学会論文誌**, Vol.9, No.2, pp.141–150, 2004.

[99] 河野義広, 米倉達広:“Count Down Protocol を用いた P2P 型仮想球技における Critical Case 発生回避の実現”, **電子情報通信学会論文誌** *D*, Vol.J89-D, No.10, pp.2219–2228, 2006.

[100] 河野義広, 塙大, 米倉達広, “相補予測機能を用いた P2P 型ネットワーク対戦エアホッケーの試作”, 技術研究報告 MVE2003–68, 電子情報通信学会, 2003.

[101] 金田裕剛, 峰松美佳, 斉藤匡人, 間博人, 徳田英幸, “P2P ネットワークゲームのための階層型遅延最適化ミドルウェアの提案と評価”, **第 *66* 回情報処理学会全国大会**, pp. 521–522, 2004.

[102] 桑子純一, 瀬崎薫:“分散環境におけるメディア同期”, **電子情報通信学会技術研究報告．*SSE*，交換システム**, Vol.98, No.298, pp.67–72, 1998.

[103] 引地謙治, 森野祐直, 福田一郎, 松本壮樹, 瀬崎薫, 安田靖彦:“触覚を含む仮想空間共有システムの提案と評価”, **電子情報通信学会論文誌** *(B)*, Vol.J86–B, No.2, pp.268–278, 2004.

[104] 古川俊治, 若林剛, 小澤壯治, 渡邊昌彦, 大上正裕, 北川雄光, 石井誠一郎, 有澤淑人, 大森泰, 納賀克彦, 北島政樹:“3.master-slave manipulator を用いた手術と遠隔手術指導 (＜特集＞未来のための今： Robotic surgery と遠隔手術指導)”, **日本外科学会雑誌**, Vol.101, No.3, pp.293-298, 2000.

[105] 橋本孝之, Thomas B. Sheridan, Mark V. Noyes:“時間遅れを有するテレオペレーションにおける予告情報の効果”, **日本人間工学会誌**, Vol.22, No.2, pp.91–92, 1986.

[106] 財団法人インターネット協会, **インターネット白書** *2007*. 株式会社インプレス R&D, 2007 edition.

[107] 塙大, “分散仮想環境における時間的・空間的要因に関する研究”, 博士学位論文, 茨城大学, 2005.

[108] 塙大, 笹本綾子, 米倉達広, “遅延のある分散仮想環境における予測機能の検討とゲーム応用”, 技術研究報告 MVE2001–141, 電子情報通信学会, 2002.

[109] 竹村治雄:“ネットワークを使ったコミュニケーション”, **日本 *VR* 学会誌**, Vol.4, No.1, pp.59–63, 1999.

[110] 波多野健, 山本泰秀, 高松亮, 佐藤誠:“予測動作提示による仮想遠隔共同作業環境の時間遅れ補償”, **日本バーチャルリアリティ学会大会論文集**, Vol.1, pp.155–158, 1996.

[111] 米倉達広, 塙大:“通信遅延を伴う分散仮想環境データのテイラー展開による予測手法”, **電子情報通信学会論文誌（*D-II*）**, Vol.J87-D-II, No.12, pp.2209–2220, 2004.

[112] 大里和史, 河野義広, 米倉達広, “仮想球技におけるパス到達ゾーンの自動生成手法”, **電子情報通信学会** *2* **種研究会サイバーワールド** *(CW)* **第** *5* **回研究会報告**, p.2006.

[113] 大里和史, 河野義広, 米倉達広, 塙大, “仮想球技におけるパス可能領域の自動生成手法”, **インタラクション** *2007* **論文集付録 ポスターセッション**, p.2007.

[114] 和田則仁, 古川俊治, 大谷吉秀, 熊井浩一郎, 徳山丞, 磯部陽, 窪地淳, 北島政樹:“インターネットを介した遠隔手術指導”, **日本外科学会雑誌**, Vol.104, No. 臨時増刊 (20030430), pp.324-325, 2003.