平成 17 年度茨城大学大学院理工学研究科情報工学専攻修士学位論文

# P2P 型分散仮想環境における
# 共有オブジェクトの管理手法に関する研究

提出年月日：　平成 18 年 2 月 10 日
著　者　　：　茨城大学大学院理工学研究科情報工学専攻 04NM705H 河野　義広
指導教官　：　茨城大学工学部情報工学科教授　米倉　達広

# 目次

## 第1章　はじめに

近年，インターネットの急速なブロードバンド化に伴い，多人数参加型オンラインゲームをはじめとする実時間対話型アプリケーションが急増してきた．しかしながら特に，アクションゲームやスポーツゲームなどのリアルタイム性の高いアプリケーションでは，通信遅延の問題が顕著であることが指摘されている[1]-[3].

通常，分散仮想環境(以下 DVE と略記)では，サーバ・クライアント型トポロジが用いられることが一般的である[1]．この型の DVE では，仮想空間の整合性(consistency)の面で有利ではあるが，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の問題が知られており，応答性(throughput)が犠牲となる[4]．そこで最近ではネットワーク通信の高速化や, 通信データの高圧縮化など個別の対策がとられているが, これらは通信遅延自体の本質的な軽減に寄与するものではない. 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 が自分以外の 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 の状態を同期させるのが難しく仮想空間に不整合が生じる．これは consistency-throughput のトレードオフ問題[4]として知られており，整合性を優先するには応答性が犠牲となる．つまり，P2P 型 DVE では Local な情報は即座に提示できる一方，Remote の情報は提示内容(時刻)に差が生じるというように, 情報提示時刻の不整合の問題が大きく, 共有オブジェクトの扱いが困難であることが知られる[1]，[3].

また，P2P を用いた大規模ネットワークにおいて更新情報を共有する通信プロトコルの研究なども報告されている[6]-[10]．しかし，ここでは実際のネットワークではなく，計算機シミュレーションのみによる評価であると同時に各アバタ（DVE 上の仮想プレイヤ）情報の通信のみが想定されており，実際の共有オブジェクトの扱い方に言及してはいなかった．更に，peer 間の円滑なインタラクションを犠牲にすることなく，アバタ間の幾何学的な関係をもとに通信トラフィックを減少させる研究もなされている[11]．しかしながら，ここでは consistency- throughput のトレードオフ問題に関して, その解決策を講じているわけではない.

ここで，オンラインゲームに関していえば，P2P で MMOG (Massively Multiplayer Online Game)を実現するための分散アーキテクチャを提案した研究[12]や，株式会社マルチタームによるネットワークプログラミングの煩雑さを解消するための通信ライブラリ MPS (MassplayerSystem)[13]の開発などをはじめ，オンラインゲームの研究・開発が盛んに行われている．MPS では，あらゆるタイプのゲームジャンル，プラットフォーム，及びネットワークトポロジに依存しないゲームの開発が可能となっている.

更に近年，オンラインゲームは携帯電話や PDA などの携帯端末への広がりも見せている．現在，携帯アプリの開発は Java[14]が主流となっているが，Java VM (Virtual Machine)上での実行や HTTP (Hyper Text Transfer Protocol)に依存した通信のため，リ

アルタイム性の高いアプリケーションの開発が困難である．そこで，携帯端末上の実行速度などの観点から米国 QUALCOMM 社の BREW (Binary Runtime Environment for Wireless)[15]が注目を集めている．BREW とは，携帯端末向けプラットフォームであり，Java に比べ高速な処理性能や, TCP や UDP といったより低レベルな通信プロトコルが利用可能である．そのような中で，株式会社マルチタームでは携帯電話上での多人数参加型 RPG (MMORPG)などの開発に適した通信ライブラリ MMB (MassplayerSystem Mobile for BREW)[13]の販売，株式会社タイトーでは P2P 対戦が可能な EZ アプリ(BREW)対応のオンラインゲームサービス[16]が，それぞれ開始されている．ここで[16]の開発には，株式会社吉田鎌ヶ迫の携帯電話用 P2P フレームワーク「Spear」[17]が採用されており，タイムラグの少ない通信対戦が可能となったことで，エンドユーザが増加しても通信スピードの劣化はないとされている．このように，オンラインゲームの開発において P2P の技術は注目されつつあり，今後の研究が期待される分野である．

実際, P2P 型オンラインゲームの研究としては，P2P 型マルチプレイヤゲームにおいて，ゲームの不正行為を検出するためのプロトコル[18]，[19]，フィールド型球技を実現するためのプロトコル[33]を提案した研究なども報告されている．文献[33]では，ボールなどの共有オブジェクトの扱いに関して議論がなされているが，その管理権(共有オブジェクトを制御する権利)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．

そこで本論文では，応答性の良さをある程度維持したまま，P2P 型 DVE において，仮想空間の整合性及び一貫性のある共有オブジェクトの管理手法を目的とする．

## 第2章　関連研究

DVE 構築の際には，consistency-throughput のトレードオフ問題が発生する. 通常は，仮想空間の整合性を得るため，サーバ・クライアント型のトポロジを適用することが一般的である．そのため，サーバ・クライアント型では応答性の向上が課題となり，一方のP2P型では整合性の保証が課題となる．そこで本章では，DVEに関する関連研究についていくつか紹介する．なお，以下で紹介する関連研究のうち，2.1-2.3はサーバ・クライアント型，2.4-2.8はP2P型に関する研究である．

### 2.1. NPSNET

・目的

NPSNET[20]は，アメリカの軍事シミュレーションを目的として開発されたシステムで，HMD (Head Mounted Display: HMD)や各種センサーにより，ユーザ間の複雑なインタラクションを可能とする. NPSNET では，サーバ･クライアント型トポロジを採用している．

・問題点

サーバ・クライアント型では，各クライアントからのメッセージを仲介するサーバの過負荷により，各クライアントへの応答性が低下してしまう．そのため，階層化や負荷分散を行ってシステムを構成することが必要である．

・解決策

NPSNET では，サーバの階層化により負荷分散を行っている．また，サーバから各クライアントへの通信にはマルチキャストを用いる．ここでは，各アバタを仮想空間内での位置情報などにより複数のマルチキャストグループに分類し，それぞれのアバタが適当なグループにのみ更新メッセージを送信することで全体のメッセージ数を減らしている．NPSNET におけるマルチキャストグループは，セルとよばれる六角形の領域によって分割される（図 1）.

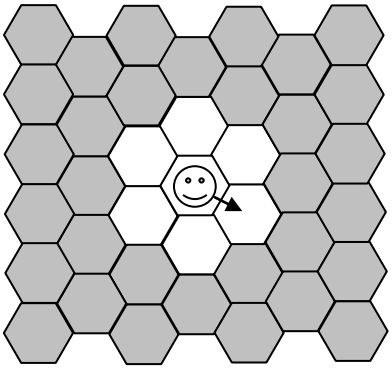


図 1. NPSNET での領域分割
Fig.1 VE division in NPSNET

## 2.2. チッタトロン

・目的

チッタトロン[21]，[22]は，多人数空間共有型アプリケーションである．内容は，空間内に隠された宝を速く集めるというゲームである．ユーザは画面に描画される自分のキャラクタを操作し，仮想空間内を自由に移動する．なお，サーバ・クライアント型のトポロジを採用している．

・問題点

2.1 の NPSNET では，静的に仮想空間を分割しているため，局所的な利用者の増加により一部のサーバに負荷が集中し，クライアントへのサービスの品質低下といった問題が生じる．

・解決策

そこで，この問題に対してチッタトロンではマルチサーバを用いることで対処する．サーバは，仮想空間の情報をクライアントへ配信するVS (Virtual Space)サーバとユーザの集中を回避するための負荷分散の実行を指示するマスターサーバの2 種類からなる．

サーバの負荷が増大した場合には，管理する仮想空間をサーバ間で動的に変更する．これにより，サーバ間での負荷を均等に分散することができる．図 2 に VS サーバの領域分割を示す．また，各クライアントへの送信はマルチキャストを用いる．この手法では，サーバでの処理を効率化することで応答性を向上させているが，ネットワーク距離に比例した通信遅延は必ず存在する．そのため，実時間対話性の高いアプリケーションでは，通信遅延に依存する応答性はボトルネックとなる．

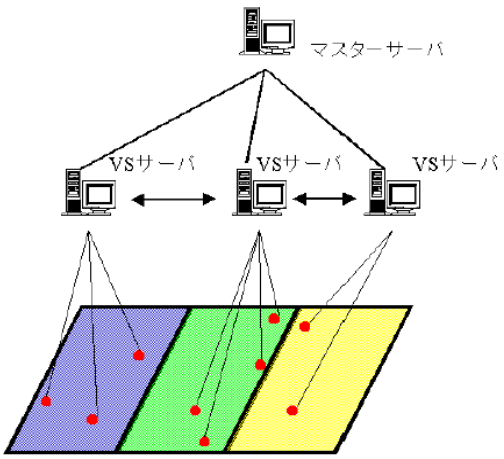



図 2. VS サーバによる領域分割
Fig.2 VE division with VS server

## 2.3. マルチサーバにおける共有オブジェクトの同期機構

・目的

マルチサーバ型仮想空間共有アプリケーションにおいて，複数のユーザがサーバに対して同時に共有オブジェクトの取得要求を行った場合，通信遅延の影響によりサーバ間で異なる判定を行い，仮想空間の不整合が生じる場合がある．また，不公平な判定となる場合もある．文献[23]では，これらの問題に対処することを目的とする．

・問題点

通常，上記の問題点を解決するためには，待ち時間を設けることが一般的である．しかしながら，取得要求に対する応答の遅れが生じ，リアルタイム性が損なわれてしまう．

・解決策

そこで，サーバ側でユーザの動きを予測し，その予測に基づくオブジェクトの取得確率を用いることで，待ち時間を短縮させる手法を提案している．これにより，応答性を高めと同時に，公平性な判断を可能としている．

## 2.4. ANGEL

・目的

文献[24]では，P2P型のネットワーク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と評価を目的とする．

・問題点

通常, P2P で DVE を構築する際は, 各 peer が全結合し通信を行うことが一般的である．しかしながら，特定の peer が通信のボトルネックとなる場合は，同期取得のため DVE 全体の応答性が低下することになる．

・解決策

そこで，メッシュ型 P2P を各 peer が部分的に結合するハイブリッド型の P2P に再構成することで，DVE 全体の応答性を向上させる（図 3）．文献[24]では，メッシュ型 P2P を遅延に基づいて再構成するための機構として，ANGEL (Architecture for Network Games utilizing Eligible Leaders)を提案している（図 4）．ANGEL では，通常ノードから受信した遅延報告に基づいて，マスターノードが経路情報を動的に再構成することにより経路の最適化を行っている．

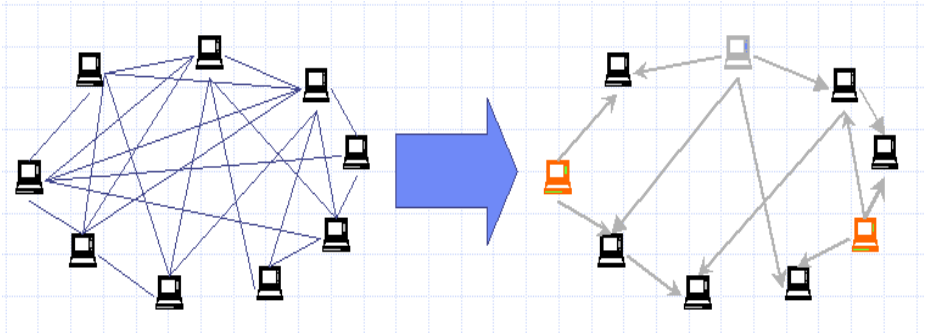


図 3. メッシュ型 P2P の再構成
Fig.3 Reconstruction of Mesh-P2P

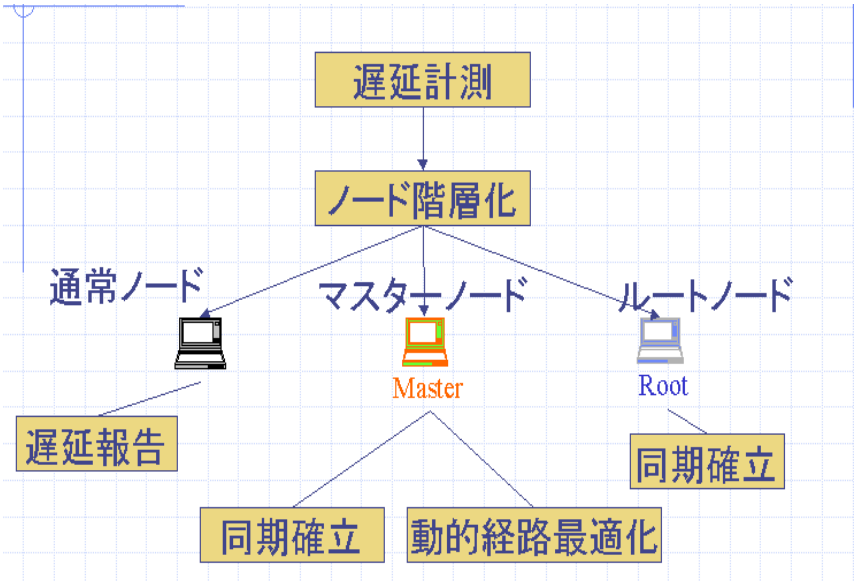



図 4. ANGEL の構成
Fig.4 The architecture of ANGEL

### 2.5. MiMaze

・目的

MiMaze[25]では, インターネット上でリアルタイムの高い複数参加型迷路ゲームを P2P 型で実現することを目的とする.

・問題点

P2P 型では, ユニキャストにより更新情報の共有を行う場合, 利用者数 $N$ に対して $O(N^2)$ のメッセージ通信が必要となり, 通信トラフィックが増加する. この問題に関しては, 以下で説明する. また, サーバが存在しないため, 整合性を保証する必要がある.

・通信について

各プレイヤは，まずゲームサーバから迷路マップをダウンロードする．その後，サーバが使われることはない．また，各プレイヤは，MBone (Internet Multicast Backbone)[26]を利用して他のプレイヤと接続し，IP マルチキャストにより通信を行う．MBone とは，マルチキャストルータ同士のトンネリングを行うことで，仮想的なネットワークを構築する技術である．これにより，インターネット上でのマルチキャストの利用が可能となる．

通信データは，自己アバタの位置及び方向である．これらのデータを IP マルチキャストにより各プレイヤへ送信する．

・整合性の保証

MiMaze では，端末間で同期を取るための機構が取り入れられている．ここでは，応答性に関して 160 ms まで許容すると制限し，同期を取っている．つまり，応答性を犠牲により整合性を得ている．

## 2.6. P2P 型ネットワークゲームにおける負荷分散機構

・目的

P2Pの環境で特定のサーバを設置することなく多人数参加型ネットワークを実現することを目的とする[27]-[29].

・問題点

従来の負荷分散機構は，仮想空間を均等に分割し，分割された複数の領域を固定数のサーバに対応させる方式や，P2P でいくつかのプレイヤの端末に割り当て処理する方法が提案されている．これらの従来法を P2P 環境に適用した場合，特定の領域へのプレイヤの集中により，その領域を担当するノードに過負荷が生じる．

・解決策

文献[27]では，均等分割した各領域のプレイヤ数とそこで発生するイベントの頻度を監視し，それをもとに領域内のプレイヤの集合を部分集合に動的に分割し，新たに割り当てたノードに分散処理させることにより，各ノードの負荷を一定水準以内に抑える方式を提案している．また，発生する各イベントの一貫性（発生時刻，及び発生順序に矛盾が生じないこと）を保証するため，タイムスロット（フレーム同期のための待ち時間）とよばれる同期機構を採用している．ここでは，200 ms程度のタイムスロットを想定している．

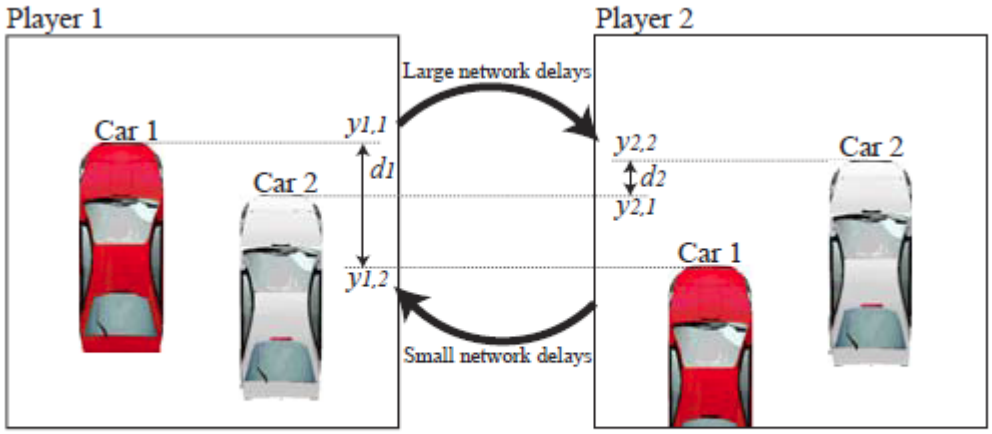
## 2.7. ネットワーク型レーシングゲーム

・目的

ネットワーク型レーシングゲームでは，高いリアルタイム性が要求されるため，コンピュータデータの入力から出力までの遅延時間に厳しい制限が伴う．したがって，通信遅延やパケットロスにより，ゲーム中の発生イベントの因果関係や状態の一貫性に乱れが生じるため，因果順序制御や予測制御といった機能が必要となる（図5）．

・解決策

文献[30]では，MU (media unit)とよばれる車の位置情報やタイムスタンプが含まれたデータの受信期限を設けることで，因果順序制御を行っている．更に，MU の受信期限をネットワークの負荷に応じて変更する．また，過去に受信した MU から現在の位置を予測することで，Dead Reckoning を用いた因果順序制御を行っている．Dead Reckoning とは，受信したデータの履歴情報をもとに，データの受信間隔よりも短い時間間隔で状態の予測・提示を行う手法である．Dead Reckoning 法に関する研究についていえば，Singhal の PHBDR (Position History Based Dead Reckoning) [5]や，塙らによる数値解析に基づく多項式型 Dead Reckoning 法の誤差解析[31]，[32]などが知られる．




(a) Case in which the positional relations of the two cars between players 1 and 2 are not consistent

図 5. ネットワーク型レーシングゲームにおける通信遅延の影響
Fig.5 Influence of network latency in networked racing games

## 2.8. Shared Soccer

・目的

Shared Soccer[33]では，インターネット上に共有された仮想空間(Shared Simple Virtual Environment : SSVE)において，P2P によるサッカーゲームの実現を目的とする．

・問題点

DVE におけるサッカーなどのフィールドゲームでは，ボールは共有オブジェクトと見なすことができる．したがって，共有オブジェクトとなるボールを各 peer で管理しなければならない．そのため，共有オブジェクトの管理権を決定する必要がある．

・解決策

以下のシステム構成により共有オブジェクトの管理権を決定する（図 6)．

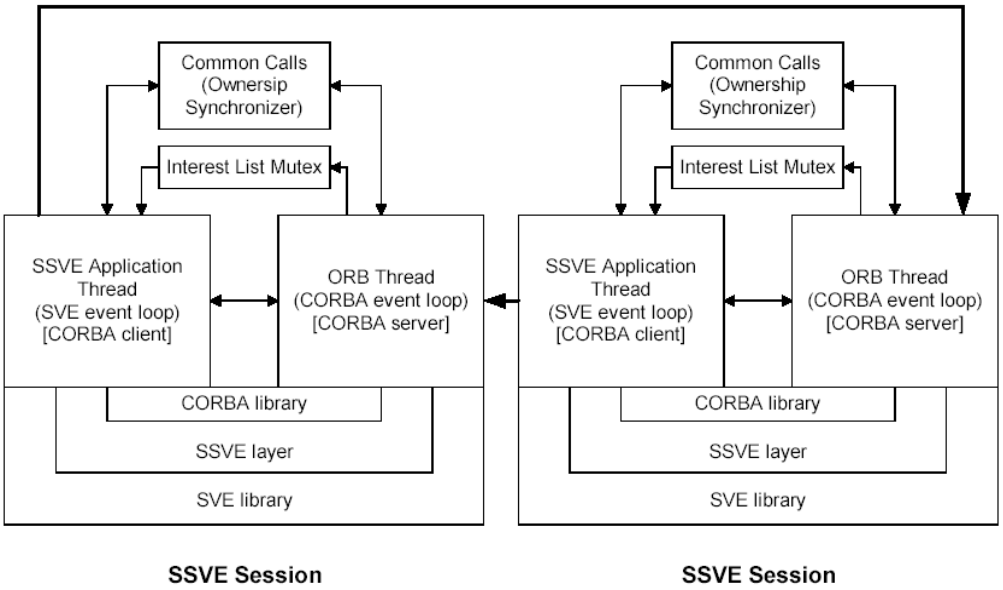



図 6. SSVE のアーキテクチャ
Fig.6 SSVE Architecture Diagram

この図では，CORBA (Common Object Request Broker Architecture)[34]を用いてシステム間の通信を行っている，CORBA とは，OMG (Object Request Broker Architecture) が作成した標準規格である．CORBA 対応のアプリケーションは，ORB (Object Request Broker)と呼ばれるオブジェクト間の通信メカニズムを提供するコンポーネントを通じて，サーバの機種，OS に関わらずインターネット間で連携操作ができる．

更に，CORBA による通信を行った上で，共有オブジェクトの管理権に関して，Ownership Synchronizer を用いてシステム間の同期を取っている．これにより，共有オブジェクトの排他制御が可能となる．しかしながら，管理者に関して同期を取るため，応答性が犠牲となる．

## 2.9. 関連研究の総括

前述のように DVE を構築する際のトポロジは，サーバ・クライアント型と P2P 型に大別できる．サーバ・クライアント型では，サーバの負荷分散やマルチサーバにおけるサーバ間の整合性の保証が課題となる．一方，P2P 型では，通信トラフィックの軽減や端末間の整合性の保証が課題となる．

また，共有オブジェクトを扱った研究もいくつか行われており，2.3 のサーバ・クライアント型の場合は応答性の向上が課題となり，2.8 の P2P 型では整合性の保証が課題となっている．しかしながら，いずれの研究においても仮想空間の整合性は最も重要な要素となるため，待ち時間の設定といった最低限の同期機構が必要不可欠である．

そこで本研究では，応答性を最優先するため，DVE の属性情報に関する端末間の同期取得は一切行わず，P2P 型 DVE において，仮想空間の整合性及び一貫性のある共有オブジェクトの管理手法の開発を目的とする．

## 第3章　P2P フィールド型 DVE モデル

本論文では, 応答性の良さをある程度維持したまま, P2P 型 DVE において, 仮想空間の整合性及び一貫性のある共有オブジェクトの管理手法の開発を目的とする. 本研究では, DVE として仮想の物理空間を想定し, アバタや共有オブジェクトの属性が位置や速度などの物理情報を示すような DVE(フィールド型 DVE とよぶ)を対象とする. フィールド型 DVE では, フィールド内のアバタや共有オブジェクトの属性(位置など)は時間的に連続して変化するものとする. また, 各アバタは共有オブジェクトに接触でき, その状態に変化を与えることができる.

P2P のフィールド型 DVE を構成するには共有オブジェクトの管理権, 並びに各 peer での物理属性(共有オブジェクトと全アバタの属性)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 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的に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属性が一致する（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[35]．本論文では，第 3 章で筆者らが既に提案している AtoZ，Dead Zone について概説する[35]-[41]．更に，第 5 章でそれらの概念をマルチプレイヤ対応[36]に拡張し，第 6 章で Dead Zone 内での管理権競合を回避するための閾値調整法[37]について議論する．そして，第 7 章で Critical Case を回避するためのプロトコル[41]を提案し，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，その有効性を実験的に検証する．

# 第4章　AtoZ (Allocated Topographical Zone)

## 4.1. AtoZ

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 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[42], [43]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 7）. i 番目のアバタの属性値を $p_i$, 全アバタ属性の集合を $P=\{p_1, p_2, \ldots, p_j, \ldots, p_n\}$, アバタの添え字集合を $N=\{1, 2, \ldots, n\}$とすれば, $p_i$ の AtoZ は具体的に

$$AtoZ(p_i)=\{\mathbf{x} \mid \mathbf{x} \in Z, Access(\mathbf{x}, p_i)=\min_{j \in N}(Access(\mathbf{x}, p_j))\} \quad (1)$$

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, $Access(\boldsymbol{x}, p_i)$ はアバタ $p_i$ が $\boldsymbol{x}$ にアクセスするまでの時間である. 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.

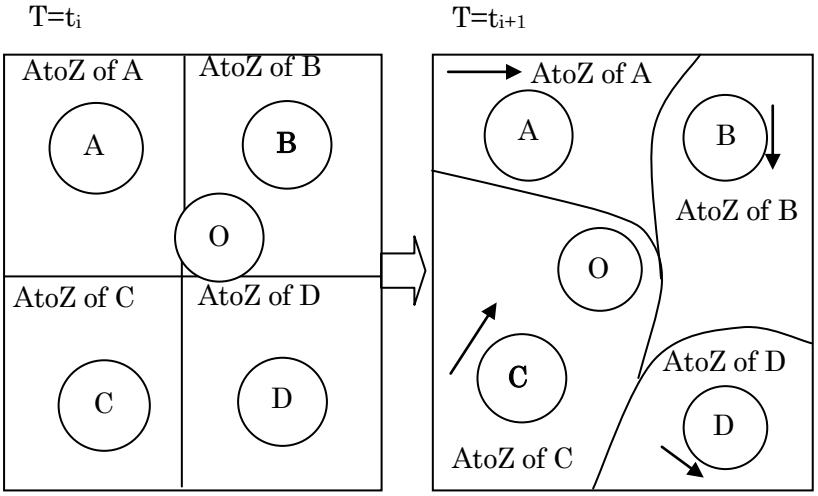



図 7. AtoZ の概要
Fig.7 Overview of AtoZ

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属性情報を他の全 peer に絶えず送信し続け，自分以外のアバタの物理属性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.

＜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.

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. これについては文献[35]で提案した相補予測により対処する.

### 4.2. 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属性の差異が 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（図 8)．

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生し得る．この現象を Phantom Case とよぶ（図 9）.

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, ゲームの応答性が犠牲となるため，できる限り排除する必要がある.

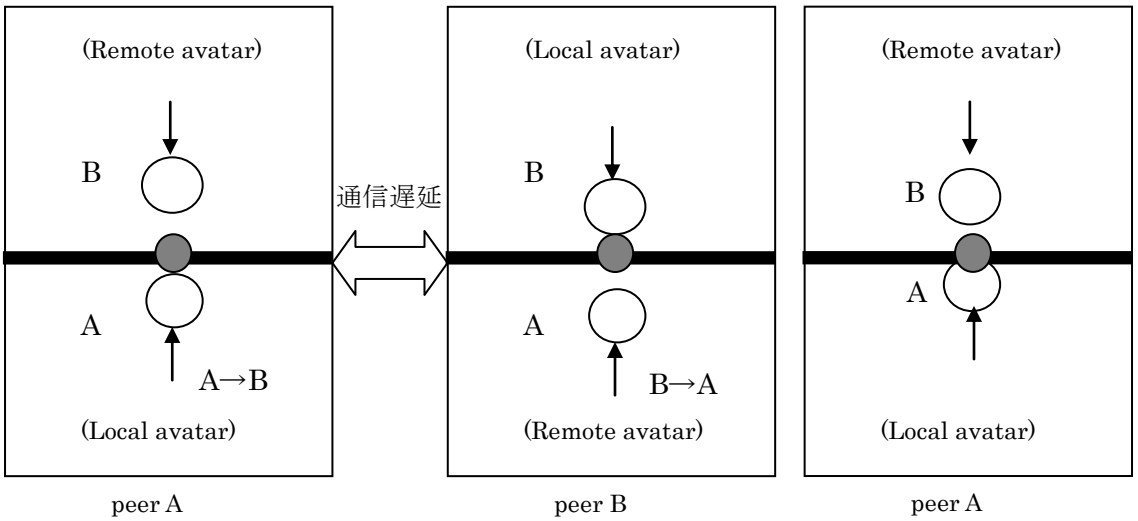



図 8. Critical Case の例

Fig.8 Example of Critical Case

図 9. Phantom Case の例

Fig.9 Example of Phantom Case

### 4.3. 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，遅延とフレーム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.

$$DZ(p_i, p_j) = \{\mathbf{x} \mid \mathbf{x} \in Z, \frac{\left|Access(\mathbf{x}, p_i) - Access(\mathbf{x}, p_j)\right|}{\Delta t} < ([d/\Delta t]+1) \quad \} \qquad (2)$$

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，Δt[ms]は更新間隔となっており，([d/Δt]+1)で遅延フレーム数を算出できる. その他は式(1)と同様である. 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 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.

## 第5章　マルチプレイヤプロトコル

前述のプロトコルは，3 個以上のアバタ(すなわち peer)が存在する DVE にも拡張することができる．ここでは，n 個の peer が存在する DVE について考察する．この場合，各 peer は n 個のアバタすべての AtoZ を計算しなければならないため，AtoZ の計算量は $O(n^2)$になると考えられる．これでは，n の増加とともに不利になる．また，アバタの数が増加することで，アバタ間で共有オブジェクトの取り合いがより多く発生し，管理権の移行も多発する．そのため，共有オブジェクトの管理権が競合するケースが増加し，Critical Case 発生の危険性頻度も増加する．

そこで，各 peer は自己アバタの AtoZ と関連する Dead Zone のみを計算することで計算量を O(n)に抑えることを考える．更に，表1に示す状態遷移行列を用いることで，共有オブジェクトの管理権競合を回避する．この表は，各 peer において領域種別や管理権の矛盾を迅速に発見するために利用される．このとき，各 peer では領域種別の peer 間での整合性と管理権の整合性を常に確認する．そのため，各 peer は自分が主張する共有オブジェクトの領域種別と管理権の情報を，他の全 peer に送信する必要がある．このようにして，各 peer がこの状態遷移行列を共通に保持できれば，互いに管理権の競合を回避することが可能となる．

表1において，左上端のセルでは複数のアバタが Dead Zone 内で管理権を主張し続けるため，複数のアバタで共有オブジェクトを同時に制御する可能性がある．したがって，Critical Case 発生の危険性が高く，最も回避すべきケースであるといえる．また，中央の「Time-consuming」セルではすべてのアバタが管理権を譲歩するため，いずれかのアバタが再び管理権を取得するまでゲームは膠着状態に陥る．右下端のセルは管理権の整合状態を示す．上記の概念を考慮した上で，マルチプレイヤ対応のプロトコルを以下に示す[36]．

<P2P 型マルチプレイヤ仮想球技での AtoZ による分散処理プロトコル>

Step (1)．全ての peer(i 番目の peer に焦点をあてる)は，他の peer から各アバタの属性データを定期的に受信する．各パケットには，最新のフレームでの領域種別(共有オブジェクトが AtoZ，Dead Zone のどちらの領域に存在しているか，あるいは Unknown なのか)及び管理権を主張しているかどうかのフラグ情報が含まれる．

Step (2)．各 peer は，受信した属性データをもとに自分自身の AtoZ のみを計算する(図 10)．

( for (j=0; j<Num_of_avatars; j++) 式(1)より，アバタ $p_j$ から共有オブジェクトまでのマハラノビスの距離[44]-[46]を計算し，アバタ $p_i$ の AtoZ を決定する．次に，アバタ $p_i$ と他のアバタとの Dead Zone を計算する．

( for (j=0;j<Num_of_avatars,j≠i; j++) 式(2)より，$DZ(p_i, p_j)$を計算する．

Step (3)．以下に示すように管理権の決定を行う;共有オブジェクトの位置をもとに現在の領域種別を計算し，関連するアバタが管理権を持っているかどうかを決定する．このフローチャートを図 11 に示す．

(遷移行列を用いた管理権決定アルゴリズム)

(a) 共有オブジェクトが $AtoZ(p_i)$に存在するかどうかチェックする．もし，存在するならば管理権フラグと領域種別をそれぞれ TRUE と ATOZ に設定し，Step 3-1)に進む．そうでないならば，(b)に進む．

(b) 共有オブジェクトが $p_i$ の関連する Dead Zone, つまり $DZ(p_i,p_j)$に存在するかどうかチェックする(ただし, $j \neq i$ である). もし, 存在するならば(c)に進む. そうでないならば, 管理権フラグと領域種別をそれぞれ FALSE と UNKNOWN に設定し, Step 3-2)に進む. Unknown とは, 少なくとも自分が管理者でないことだけは分かっており, 領域種別がどうなっているかまでは関知しないという意味である.

(c) 共有オブジェクトが関連する Dead Zone, つまり $DZ(p_i, p_j)$に存在する(ただし, $j \neq i$ である)場合, 共有オブジェクトの管理権決定は以下の式に従う. このとき, 共有オブジェクトの管理権が local avatar に存在すると判断したならば, 管理権フラグと領域種別をそれぞれ TRUE と DEADZONE に設定し Step 3-1)に進む. ;そうでないならば, FALSE と DEADZONE に設定し Step 3-2)に進む.

$$\text{local avatar} \ldots \text{if } |\text{local}(t)\text{-}O(t)| + th(t) < |\text{remote}_j(t)\text{-}O(t)| \ \forall j$$

$$\text{remote avatar} \ldots \text{otherwise} \qquad (3)$$

ここで, local(t)と $\text{remote}_j(t)$はそれぞれ, 時刻 t における local avatar の属性と j 番目の remote avatar の属性と定義する. また, th(t)は時刻 t における閾値とし, 所有権譲歩の程度を評価する. th(t)の値を大きく設定することで, Critical Case が発生する機会を抑制することができる. この th(t)の値は, 統計的あるいは経験的に定められる. 更に具体的には, 2 つ以上のアバタが管理権を主張したならば, th(t)を増加させる. 逆に, どのアバタも管理権を主張していないならば, th(t)を減少させるようにする.

Step 3-1) 共有オブジェクトの管理権は local avatar に属するものとし, オブジェクトと local avatar とのインタラクションを計算する. そして, オブジェクトの状態を更新し, DVE 上の全ての peer にそのデータを送信する.

Step 3-2) 共有オブジェクトの管理権は remote avatar に属するものとし, オブジェクトの更新情報を他の peer から受信する.

各 peer はフレーム更新時に, 自己アバタの属性を他の全 peer に送信する. また, 共有オブジェクトと静的なオブジェクト(壁やポール, ゴールなど)との衝突だけは自分自身で計算し, 共有オブジェクトの位置を更新する.

表 1. マルチプレイヤにおける管理権決定のための状態遷移行列
Tab.1 State transition matrix for the ownership determination in multi-player

| Current state / New state | Compete in Dead Zone | Compete in Dead Zone vs AtoZ | Concede in Dead Zone | Concede in Dead Zone vs AtoZ | Consistent in Dead Zone | Consistent in Dead Zone vs AtoZ | Consistent in AtoZ |
|---|---|---|---|---|---|---|---|
| Compete in Dead Zone | Most unstable | Most unstable | Oscillating | Oscillating | Warning | Warning | Warning |
| Compete in Dead Zone Vs AtoZ | Most unstable | Most unstable | Oscillating | Oscillating | Warning | Warning | Warning |
| Concede in Dead Zone | Oscillating | Oscillating | Time-consuming | Time-consuming | Pausing | Pausing | Pausing |
| Concede in Dead Zone Vs AtoZ | Oscillating | Oscillating | Time-consuming | Time-consuming | Pausing | Pausing | Pausing |
| Consistent in Dead Zone | Recovering | Recovering | Recovering | Recovering | Stable | Stable | Stable |
| Consistent in Dead Zone vs. AtoZ | Recovering | Recovering | Recovering | Recovering | Stable | Stable | Stable |
| Consistent in AtoZ | Recovering | Recovering | Recovering | Recovering | Stable | Stable | Stable(desirable) |

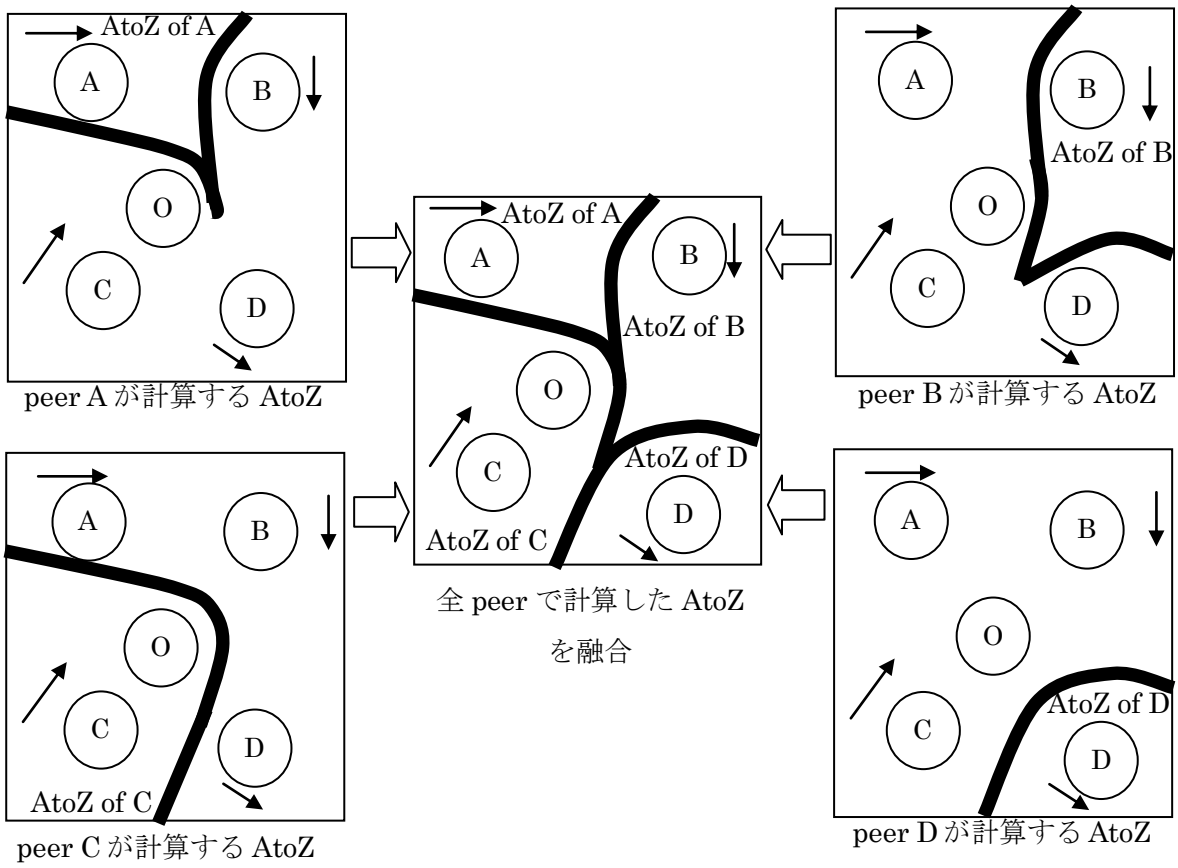



図 10. マルチプレイヤにおける AtoZ 計算
Fig.10 Computation of AtoZ for multi-player

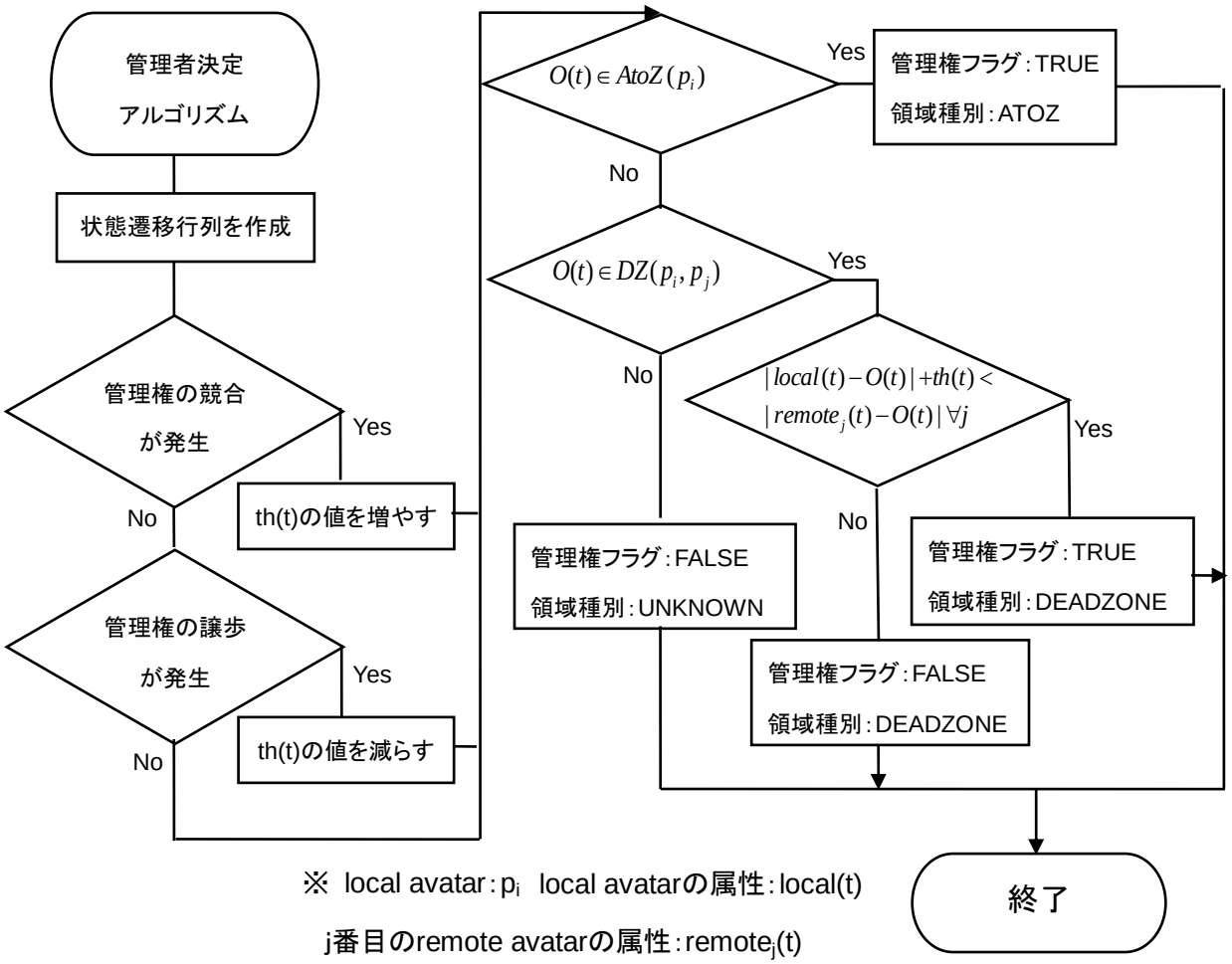



図 11. マルチプレイヤにおける管理者決定フローチャート

Fig.11 Ownership determination algorithm for multi-player

# 第6章　閾値調整法

## 6.1.　閾値調整法

前述のマルチプレイヤプロトコルでは，共有オブジェクトの管理権競合回避のため閾値 th(t)の調整法が課題となる[37]．そこで，閾値調整法として Clamp と Reset の 2 種類を採用し，それぞれの特徴を調査する．調整法を以下に示す．

・初期値:2.5[cm]
・増減値:th(t)の増減値には，次の 2 パターンを用意した．
1) Clamp (th(t)が整合時に安定)
th(t) = th(t-1) + 2.5 … 管理権の競合が発生（上限:25cm)
th(t) = th(t-1) – 2.5 … 管理権の譲歩が発生　(下限:2.5cm)
th(t) = th(t-1) … 管理権が整合　(4)
2) Reset (th(t)が整合時にリセット)
th(t) = th(t-1) + 2.5 … 管理権の競合が発生（上限:25cm)
th(t) = th(t-1) – 2.5 … 管理権の譲歩発生　(下限:2.5cm)
th(t) = reset to 2.5 … 管理権が整合　(5)

ここで，th(t)の初期値は 2.5 cm に設定した．これは，1 フレームでマレットが進行できる最大距離であり，その距離を互いが常に譲歩するよう設定した．図 12，13 にそれぞれの調整法における th(t)の増減の一例を示す．図 12，13 における管理権の状態は以下のとおりである．

- 1～10 フレーム: 競合
- 11～19 フレーム: 譲歩
- 20～28 フレーム: 競合
- 29～35 フレーム: 整合
- 36～44 フレーム: 譲歩

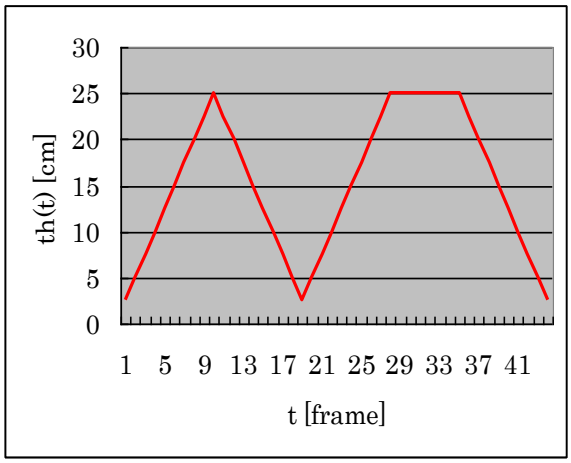



図 12.　Clamp における th(t)の増減の例
Fig.12 The behavior of th(t) in Clamp

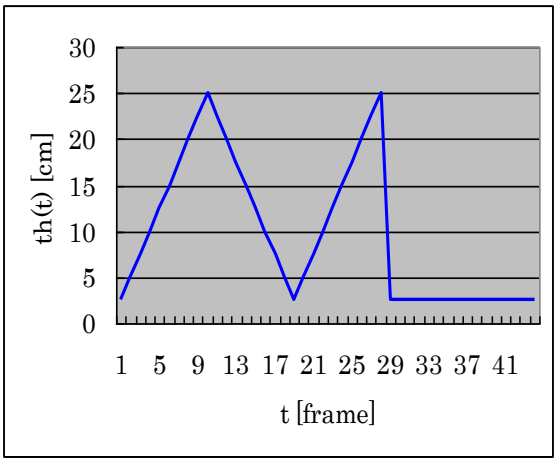



図 13.　Reset における th(t)の増減の例
Fig.13 The behavior of th(t) in Reset

## 6.2. 閾値調整法の比較実験

マルチプレイヤプロトコルにおける閾値調整法の特徴を調査するため，システムの評価実験を行う．

### 6.2.1. 実験タスク

マルチプレイヤ対応のアプリケーション例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 14)．実装は，マレット操作のためのマウスイベントの取得や，3D グラフィクス表示のため Microsoft 社の DirectX 8.0 を利用して行った[47], [48]．この図において左側は実際のゲーム画面，右側は AtoZ のようす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 DVE 全般における AtoZ, Dead Zone, Critical Case, Phantom Case の評価を行うためである．なお，peer 間の時刻同期については，あらかじめ NTP (Network Time Protocol)[49]などを用いて確保する．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．

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:$3.0\times10^{-5}$
- パック半径:0.041 [m]
- マレット半径:0.085 [m]
- パック速度の上限:3.00 [m/s]
- マレット速度の上限:1.00 [m/s]

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/frame]

図 14. エアホッケーのゲーム画面
Fig.14 The game screen of the virtual air-hockey

### 6.2.2. マレット自動化アルゴリズム

比較実験では，閾値調整法の特徴を調査することが目的となる．そこで，プレイヤによるマレットの操作といった評価を行う上で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ことにした．マレットの自動化アルゴリズムを以下に示す(図 15).

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．

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マレットの速度を決定する．

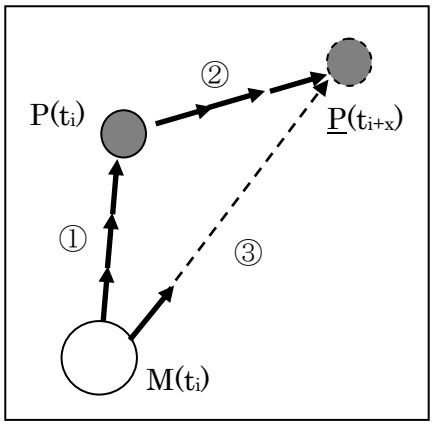



図 15. マレット自動化アルゴリズム
Fig.15 The mallet automation algorithm

図 15 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ものとする.

### 6.2.3. 実験条件

この実験の評価項目は

i)全 peer で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

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. ここで, 模擬遅延は文献[50]で提供されている NistNet を用いて発生した. また, 敵同士の通信遅延として, 以下の 4 種類を用意した. 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]
- 平均 114.0 [ms], 標準偏差 16.2 [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(2 種類)と遅延の種類(4 種類)の組み合わせ 8 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. 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. これらの実験結果を図 16-18 に示す.

### 6.2.4. 考察

図 16 を見ると, どちらの閾値調整法も遅延の増加に伴って管理権競合の確率が増加している. マルチプレイヤプロトコルでは, 各 peer は自己アバタの管理権状態や領域種別(AtoZ, Dead Zone, Unknown)の情報を他の peer に送信し, 各 peer はそれらの情報を元に閾値 th(t)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 は管理権の競合を即座に検出することができない．したがって，th(t)の値が引き上がらず，管理権競合の確率が増加する．一方，譲歩の確率は，Reset では遅延が増加してもあまり変化がないが，Clamp では遅延の増加に伴って譲歩の確率が増加している．このように，遅延が少ない場合は，Clamp も Reset も結果に大きな差異は見られないが，遅延の増加に伴って Clamp では譲歩の確率，Reset では競合の確率が増加傾向にある．この原因としては，Clamp では管理権の整合時に th(t)の値が一定となるため，譲歩時に th(t)が初期値まですぐに回復せず譲歩が増加し，逆に Reset では整合時に th(t)を初期値にリセットするため，任意の peer が管理権を容易に取得でき競合が増加しているものと考えられる.

図 17, 18 は, Critical Case と Phantom Case の発生頻度を示している. 図 17 を見ると，どちらの閾値調整法も遅延の増加に伴い Critical Case が増加している．一方，Phantom Case は，Reset では遅延が増加してもほとんど一定であったが，Clamp では遅延の増加に伴い Phantom Case も増加していた．そこで，Critical Case と競合の相関係数を調べてみると，0.942 という強い相関が見られ，Phantom Case と譲歩でも 0.965 という強い相関が見られた(表 2)．したがって，競合の確率が増加すると Critical Case が増加し，譲歩の確率が増加すると Phantom Case が増加する.

全体的な結果として, Clamp では Critical Case の割合が少なく Phantom Case の割合が多くなっており，Reset ではその逆の結果となっている．この原因については，競合や譲歩の確率と同様である．以上より，Clamp は Critical Case を排除する調整法であり，Reset は Phantom Case を排除する調整法であるといえる.

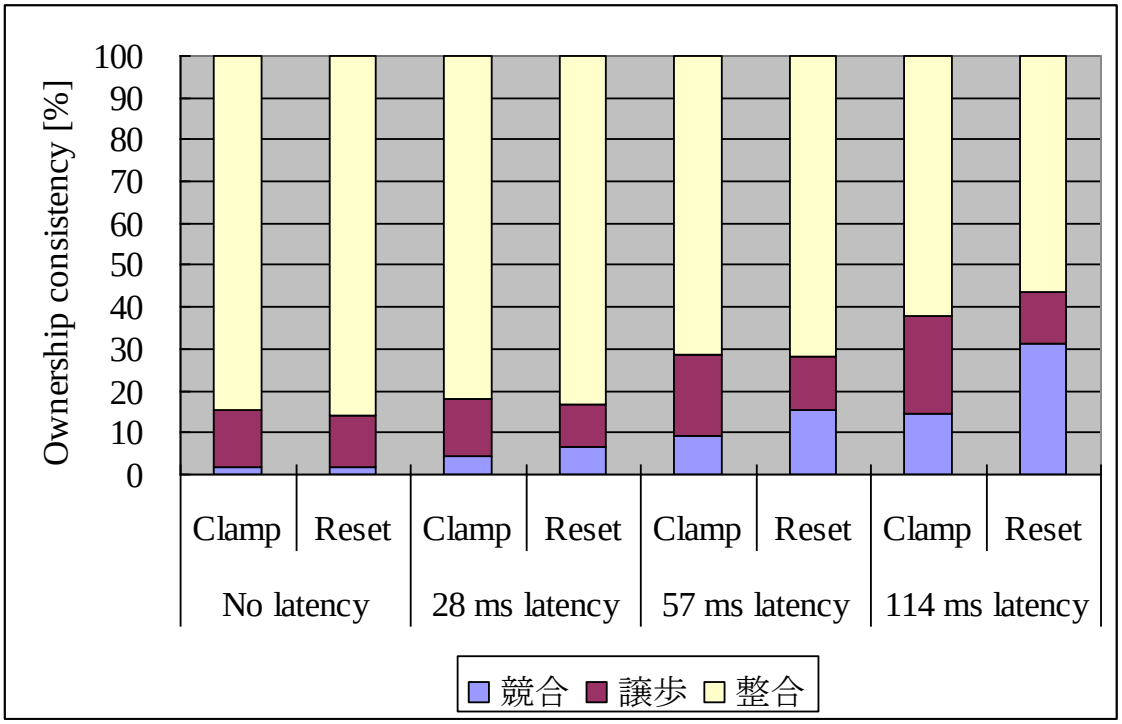



図 16. 各閾値調整法における管理権状態
Fig.16 The ownership consistency for each tuning method

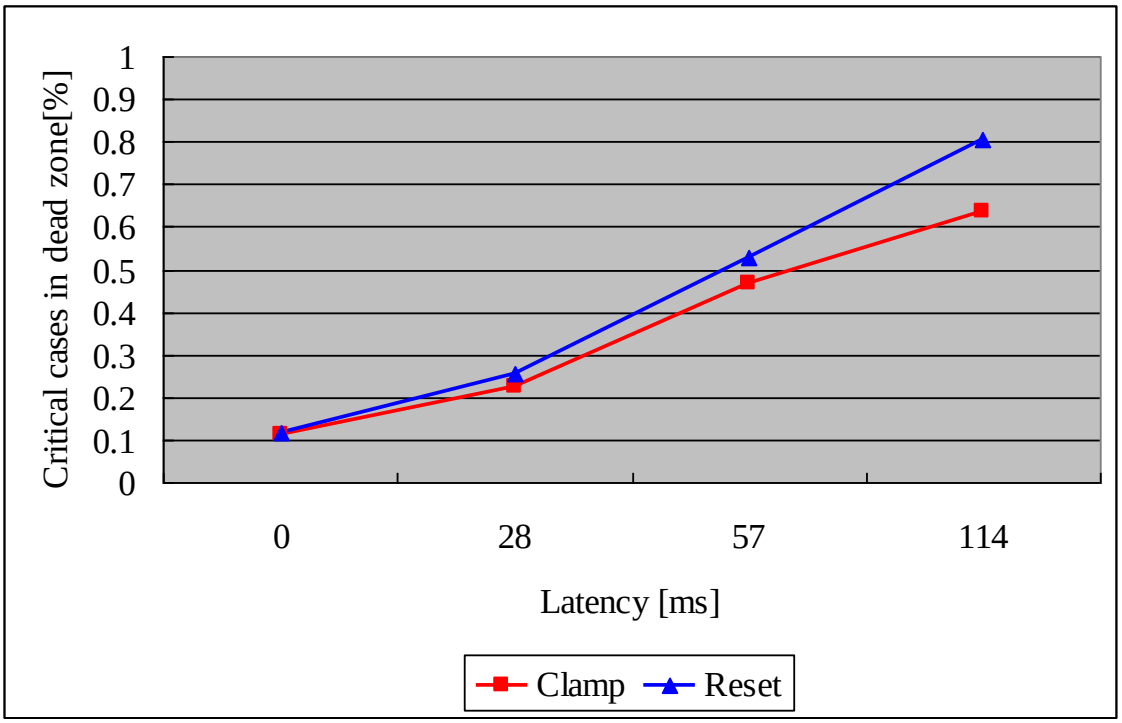



図 17. 各閾値調整法における Dead Zone 内での Critical Case 発生確率
Fig.17 The percentage of critical cases in dead zone for each tuning method

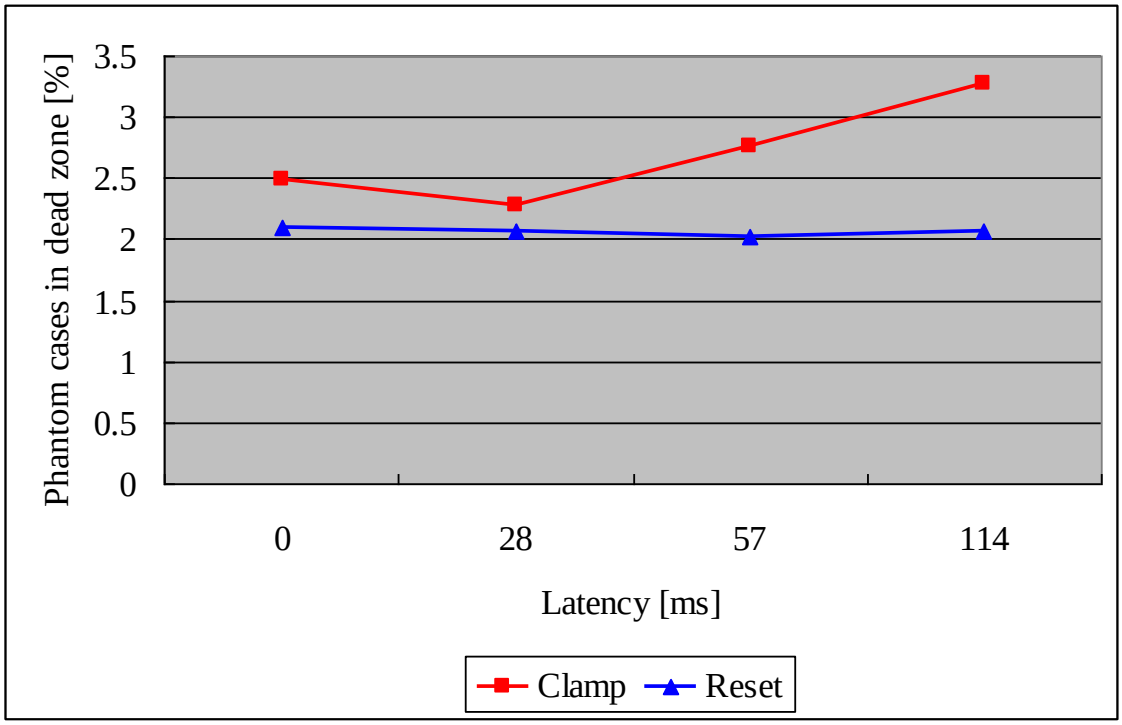



図 18. 各閾値調整法における Dead Zone 内での Phantom Case 発生確率
Fig.18 The percentage of phantom cases in dead zone for each tuning method

表 2. 全試行における相関係数
Tab.2 The correlation coefficient in all trials

| | |
|---|---|
| 整合と Critical Case の相関係数 | 0.942 |
| 譲歩と Phantom Case の相関係数 | 0.965 |

# 第7章　Count Down Protocol

第 6 章では，管理権競合を回避するための閾値調整法について議論した．しかしながら，いずれも閾値調整法も Critical Case を回避するものではなかった．そこで本章では，Critical Case の発生を回避するプロトコルを提案する[41]．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が課題となる．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．そこで本章では，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．

## 7.1. 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.

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，i 番目の peer が保持する各アバタの補正 CD 値を以下のように表す.

(i 番目の peer が保持する各アバタの補正 CD 値)

$$CD_i(1,t)=CD(1,t-l_{1i})-l_{1i},\ldots,CD_i(i,t)=CD(i,t),$$
$$CD_i(j,t)=CD(j,t-l_{ij})-l_{ij,}\ldots,CD_i(n,t)=CD(n,t-l_{in})-l_{in} \quad (6)$$

ただし，上式の左辺は各アバタの補正 CD 値を示し，その添え字は補正 CD 値を保持する peer を表す．また，右辺は当該 peer が保持する実際の補正 CD 値を示している．ここで, 各式の第 1 引数は当該 peer が補正 CD 値を保持するアバタ, 第 2 引数はその時刻[frame]となっている．なお，$l_{ij}$[frame]は peer i，j 間の片道の通信遅延を表す.

ここで，簡単に 2 つの peer の場合における CDP の概要を図 19 に示す．この図では，各 peer が保持する補正 CD 値が表されている．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，peer 間で対称の遅延を想定しているが，非対称の遅延にも容易に拡張可能である.

図 19 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 から遅延フレーム数 p を差し引いた時点での値となる．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.

(peer A が保持する補正 CD 値)

$$CD_A(A,t)=CD(A,t),\quad CD_A(B,t)=CD(B,t-p)-p \quad (7)$$

(peer B が保持する補正 CD 値)

$$CD_B(A,t)=CD(A,t-p)-p,\quad CD_B(B,t)=CD(B,t) \quad (8)$$

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 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 19 の場合，以下の式を満たすことになる.

$$CD_A(B,t)=CD(B,t-p)-p\le 0 \quad \text{かつ} \quad CD_B(A,t)=CD(A,t-p)-p\le 0 \quad (9)$$

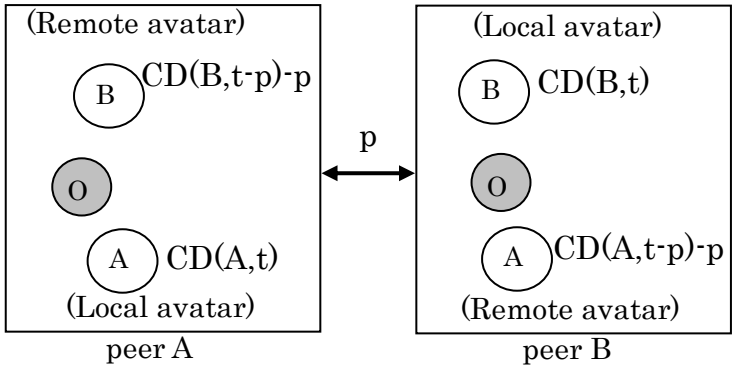



図 19. Count Down Protocol の概要
Fig.19 Overview of Count Down Protocol

これにより，管理権の譲歩が続きゲームが膠着状態（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，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．

### 7.2. 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．

CDP において，各 peer は他のアバタが遅延フレーム間に到達可能な最大距離を評価するため，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．同様に，他の peer における自己アバタの補正 CD 値も最小値に見積もられることになる．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．

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）.

## 7.3. Count Down Protocol の評価実験

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との比較実験を行った．また，実際のネットワーク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．なお，実験タスク，及び比較実験に使用するマレット自動化アルゴリズムは，6.2 で利用したものと同じである.

### 7.3.1. 実験条件

この実験の評価項目は

i)全 peer で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

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. ここで, 模擬遅延は文献[50]で提供されている NistNet を用いて発生した. 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, 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.

- Type 0:遅延なし
- Type 1:平均　28.0 [ms], 標準偏差　4.0 [ms]
- Type 2: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. 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. なお, 閾値調整法における閾値 th(t)の調整は以下のとおりである.

・初期値:4 [cm]
・増減値:th(t)の増減値は, 次式に従う.

$$th(t) = th(t\text{-}1) + (4 \times Competition\_repeats)$$
… 管理権の競合が発生（上限:40 cm)

$$th(t) = th(t\text{-}1) - (4 \times Concession\_repeats)$$
… 管理権の譲歩が発生（下限:　4 cm)

$$th(t) = th(t\text{-}1)$$
… 管理権が整合　　(10)

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 21-23 に示す. 図 20 に閾値調整法における th(t)の増減の一例を示す. 図 20 における管理権の状態は以下のとおりである.

- 1～ 5 フレーム: 競合
- 6～ 9 フレーム: 譲歩
- 10～13 フレーム: 競合
- 14～19 フレーム: 整合
- 20～25 フレーム: 譲歩

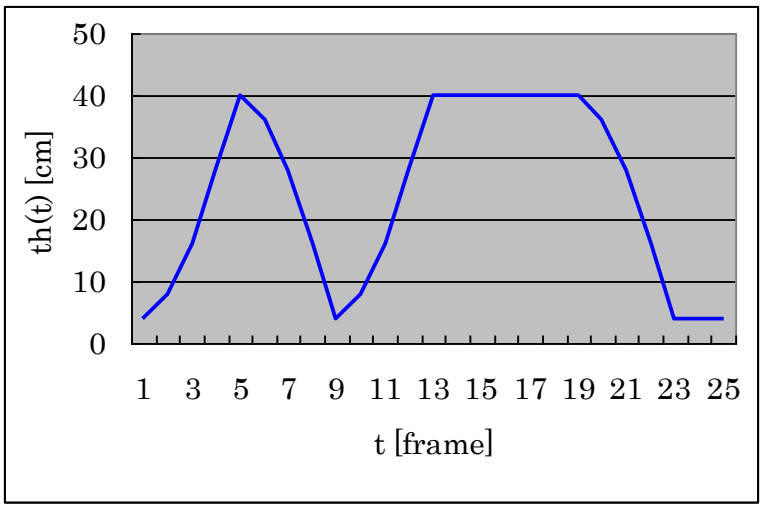



図 20.　閾値調整法における th(t)の増減の例
Fig.20 The behavior of th(t) in tuning method

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．

### 7.3.2. 考察

図 21 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)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)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．なお，CDP において，競合の状態が僅かに発生しているが，これは Dead Zone への進入と退出のタイミングのずれにより発生したものである．このとき，各アバタと共有オブジェクトとの距離は離れているため，Critical Case には発展し得ない．その一方で，ゲーム全体の 90%以上が譲歩の状態であった．この理由としては，マレットの自動化によりアバタ同士の接近の機会が増加し，Critical Case 発生の危険性が増加したためであると考えられる．CDP では，Critical Case 発生の危険性を回避するため，譲歩の状態が増加する．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 22, 23 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)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.

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．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．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．更に，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 は実際のネットワークゲームにも十分適用可能であると考えられる．

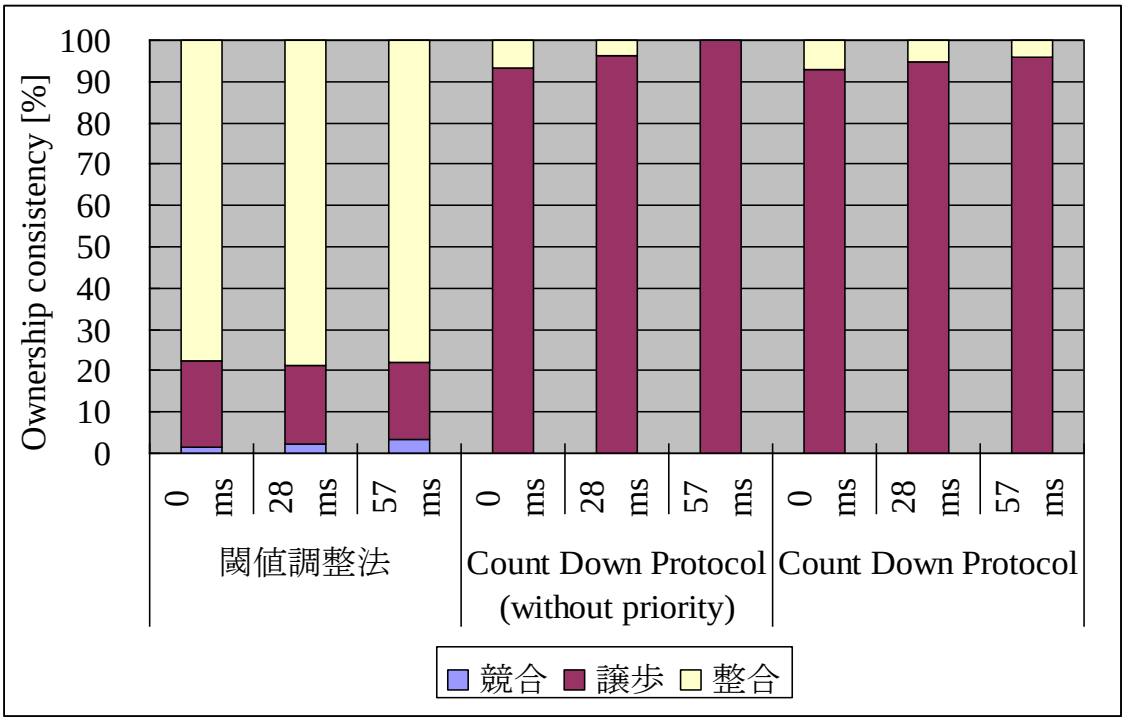



図 21. 各プロトコルにおける管理権状態
Fig.21 The ownership consistency for each protocol

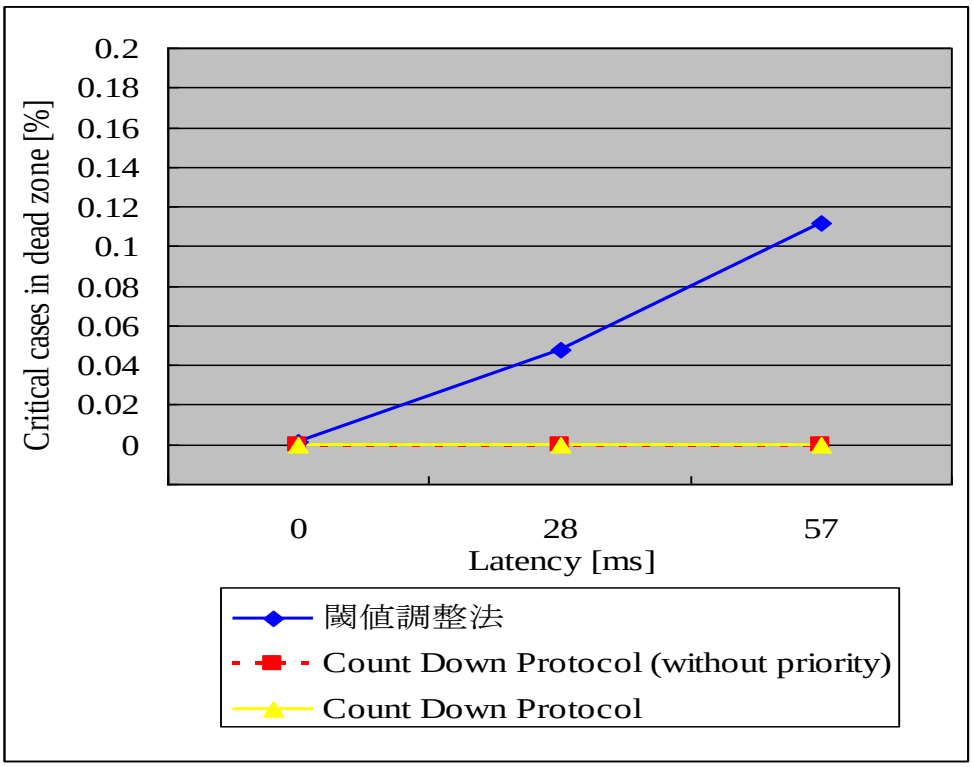



図 22. 各プロトコルにおける Dead Zone 内での Critical Case 発生確率
Fig.22 The percentage of critical cases in dead zone for each protocol

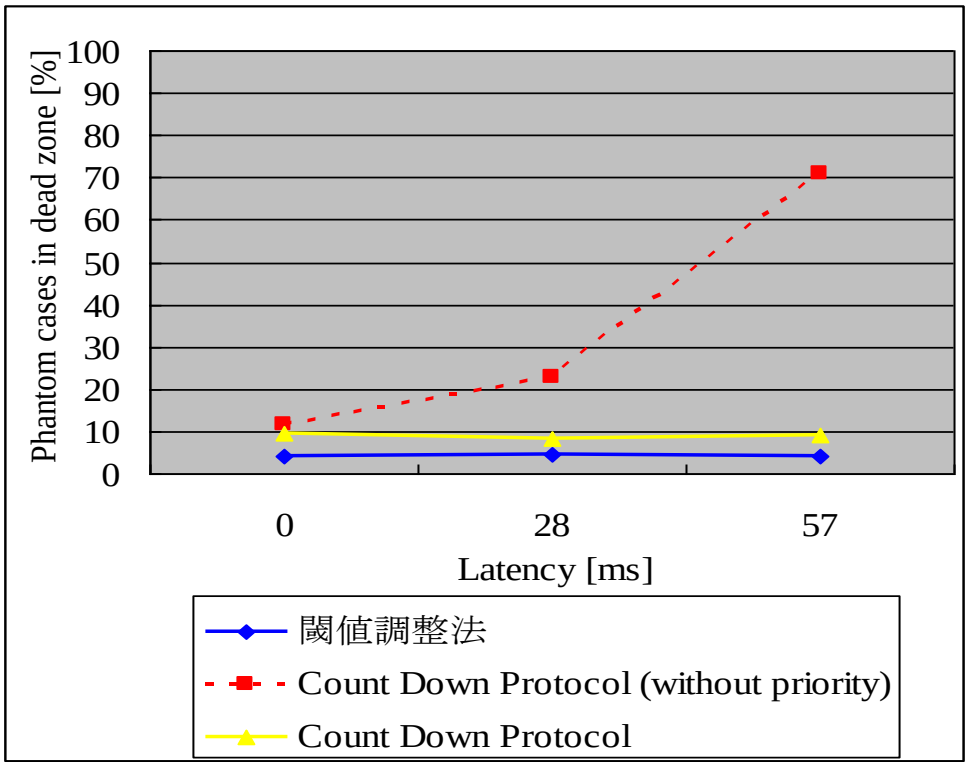



図 23. 各プロトコルにおける Dead Zone 内での Phantom Case 発生確率
Fig.23 The percentage of phantom cases in dead zone for each protocol

# 第8章　まとめ

本論文では，P2P 型分散仮想環境において，仮想空間の整合性及び一貫性のある共有オブジェクトの管理手法を目的とした．そこでまず，P2P のフィールド型 DVE における共有オブジェクトの管理手法に関して，マルチプレイヤ対応に拡張した AtoZ を提案評価した．更に，マルチプレイヤにおいて顕著な問題となる Critical Case の発生に関して，閾値調整法によりその発生頻度減少方法を考察した．しかしながら，Critical Case の発生を完全に回避するものではなかった．

そこで，P2P 型マルチプレイヤ仮想球技において，Critical Case の発生を回避するために CDP 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．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 CDP の有効性を検証した．

その結果，Critical Case の発生を完全に回避でき，本論文の課題である一貫性のある共有オブジェクトの管理が可能となった．更に，優先順位方式により Phantom Case の発生を最小限に抑制できた．また，実際に被験者による実験を行ったところ，Critical Case は一度も発生しなかった．これらの結果は，実際のネットワークゲームへの導入の可能性を示唆するものである．

今後の課題としては，共有オブジェクトの排他制御にとどまらず，Phantom Case の更なる削減のための複数アバタによる共有オブジェクトの同時制御，仮想球技を含めた AtoZ の適用範囲のモデル化，及びインターネット上での実運用に向けた課題の検討などが挙げられる．

**謝辞**

本研究に関して多くのご助言をいただき，議論をして下さいました米倉達広教授に深く感謝申し上げます.

また，お忙しい中，本研究に関して様々な議論をして下さいました東京農工大の塙大先生に心より感謝致します．そして，本研究室において論文，学会など 3 年間苦楽を共にした平木和輝君，提案されたプロトコルなどに関して，厳しいご指摘を頂きました横田可奈子さん，山河旬之介君，有限会社ラーニングアイの新堀道信社長，コンピュータに関して数々の高度なテクニック及び知識を惜しげもなく披露してくれた田中貴史君，私の話をいつも楽しそうに聞いて下さった山本瑞秋君，加藤公久君，エアホッケーの実験に関して，素晴らしくアグレッシブな結果を残し，研究室の雰囲気を盛り上げてくれた卒研生の大里和史君，小林俊君，鈴木俊裕君，高橋司君，本研究に携わったすべての方々に多大なる感謝の意を表します.

最後となりましたが，これまでに本研究より誕生した概念，プロトコル達に感謝します.「生まれて来てくれてありがとう」．その中でも，半年以上もの長きに渡り議論されてきた Tuning Method 改メ閾値調整法は，Critical Case 発生の原因究明に多大なる恩恵をもたらしてくれました．このことは，Count Down Protocol の礎へと繋がる形となりました．ここに，心から感謝の意を表すると共に，今後より一層の自己研鑽を積んでいきたいと考えております.

*Thank you very much everyone!*

## 参考文献


[1] 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, pp59-63, Jun. 1999.

[2] 橋本孝之, T.B. Sheridan, M.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", *人間工学*, vol.22, no.2, pp.91-92, Mar. 1986.

[3] 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 VR 学会大会論文集*, vol.1, pp.155-158, Oct. 1996.

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

[5] S.K. Singhal, "Effective remote modeling in large-scale distributed simulation and visualization environments", *Ph.D. Dissertation*, Department of Computer Science, Stanford University, Palo Alto, Aug. 1996.

[6] J. Keller, G. Simon, "Toward a Peer-to-Peer Shared Virtual Reality", *RESH'02 (IEEE Workshop on Resource Sharing in Massively Distributed Systems) in conjunction with the 22nd Int. Conf. on Distributed Computing Systems (ICDCS'02)*, Vienna, Jul. 2002.

[7] B. Knutsson, H. Lu, W. Xu, B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games", *IEEE Infocomm 2004*, Hong Kong, Mar. 2004.

[8] Y. Kawahara, H. Morikawa, T. Aoyama: "A Peer-to-Peer Message Exchange Scheme for Large-Scale Networked Virtual Environments", Telecommunication Systems, vol. 25, no. 3-4, pp.353-370, March-April 2004.

[9] 川原圭博, 松本延孝, 森川博之, 青山友紀, "ネットワーク仮想環境のための分散型通信アーキテクチャ", *電子情報通信学会技術研究報告*, IN2001-229, Mar. 2002.

[10]"JXTA Project's", http://www.jxta.org/.

[11]A. Goldin, C. Gotsman, "Geometric Message-Filtering Protocols for Distributed Multiagent Environments", *PRESENCE*, Vol. 13, Issue 3, The MIT Press, June 2004.

[12]C. GauthierDickey, D. Zappala, V. Lo, "A Distributed Architecture for Massively Multiplayer Online Games", Apr. 2004.

[13]"株式会社マルチターム", http://www.multiterm.co.jp/.

[14]"Java Technology", http://www.sun.com/java/.

[15]"QUALCOMM BREW Home", http://brew.qualcomm.com/brew/en/.

[16]"パズルボブル ONLINE!", http://www.taito.co.jp/.

[17]"株式会社吉田鎌ヶ迫", http://www.yoshidakamagasako.com/.

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

[19]E. Cronin, B. Filstrup, A.R. Kurc, S. Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architectures", *In Proc. of Net Games 2002*, pp.67-73, Apr. 2002.

[20]M. Zyda, F. Zyda, "NPSNET-IV: Inserting the Human into the Networked Synthetic Environment," *SIGGRAPH Video Review*, Issue 124, Entry 16, SIGGRAPH 97, Los Angeles, California, 3 - 9 August 1997.

[21]井芹威，堀真人，藤川和利，下條真司，宮原秀夫，“多人数参加型ネットワークアプリケーションの広域インターネット環境における利用実験”，*電子情報通信学会技術研究報告*，IN2000-121，MVE2000-91，Nov. 2000.

[22]M. Hori, T. Iseri, K. Fujikawa, S. Shimojo, H. Miyahara, "CittaTron: a Multi-server Networked Game with Load Adjustment Mechanisms on the Internet," *Proc. of the 2001 SCS Euromedia Conference*, Spain, pp.253-260, Apr. 2001.

[23]藤川和利, 伊藤敦史, 下條真司, 宮原秀夫, "多人数参加可能なネットワーク型仮想空間共有アプリケーションにおける同期機構に関する考察", *電子情報通信学会技術研究報告*，IA2002-32, pp.73-79, Oct. 2002.

[24]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，Mar. 2004.

[25]C.Diot, L.Gautier, "A distributed architecture for multiplayer interactive applications on the internet", *IEEE Proc. Multimedia Systems Conference*, pp.6-15, Jul. 1999.

[26]“The internet multicast backbone”, http://www.serpentine.com/~bos/tech/mbone/.

[27]公原勝彦，安本慶一，伊藤実，“P2P 環境でのネットワークゲーム向け負荷分散機構の提案”, *第11 回マルチメディア通信と分散処理 (DPS) ワークショップ論文集*, pp.79-84, Dec. 2003.

[28]山本眞也，村田佳洋，安本慶一，伊藤実，“P2P 環境でのネットワークゲーム向け付加分散機構とその評価”，*情報科学技術フォーラム（FIT 2004）講演論文集*，Sep. 2004.

[29]S. Yamamoto, Y. Murata, K. Yasumoto. M. Ito, “A Distributed Event Delivery Method with Load Balancing for MMORPG”, *ACM NetGames 2005 Workshop*, Oct. 2005.

[30]T. Yasui, Y. Ishibashi, T. Ikedo, "Influences of Network Latency and Packet Loss on Consistency in Networked Racing Games", *in Proc. ACM NetGames'05*, Oct. 2005.

[31]D. Hanawa, T. Yonekura, “On the Error Modeling of Dead Reckoned Data in a Distributed Virtual Environment”, *IEEE Proc. 2005 Int. Conf. on Cyberworlds*, Singapore, pp.279-286, Nov. 2005.

[32]塙大，“分散仮想環境における空間的・時間的要因に関する研究”，*平成十七年度茨城大学大学院理工学研究科博士論文*，Sep. 2005.

[33]J.M. Linebarger, 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, Jun. 2004.

[34]“CORBA: OMG’s Website”, http://www.omg.org/corba/.

[35]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VR 学会論文誌*, Vol.9, No.2, pp.141-150, Jun. 2004.

[36]T. Yonekura, Y. 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, May. 2005.

[37]Y. Kawano, T. Yonekura, "On a Serverless Networked Virtual Ball Game for Multi-Player", *IEEE Proc. 2005 Int. Conf. on Cyberworlds*, Singapore, pp.271-278, Nov. 2005.

[38]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”，*電子情報通信学会研究技術報告*，MVE2003-125，pp.25-30,

Mar. 2004.

[39]T. Yonekura, Y. Kawano, D.Hanawa, "Peer-to-peer Networked Field-type Virtual Environment by Using AtoZ", *IEEE Proc. 2004 Int. Conf. on Cyberworlds*, Tokyo, pp.241-248, Nov. 2004.

[40]河野義広，米倉達広，“Peer-to-peer 型マルチプレイヤ仮想球技への拡張”，*電子情報通信学会研究技術報告*，MVE2004-41，pp.7-12，Dec. 2004.

[41]河野義広，米倉達広，“P2P 型マルチプレイヤ仮想球技におけるクリティカル・ケース回避プロトコルの提案”, *電子情報通信学会研究技術報告*, CW2006-5, pp.15-20, Jan. 2006.

[42]瀧剛志，長谷川純一，“勢力範囲に基づいたチームスポーツ解析”，*情報処理*，42 巻 6 号，pp.582-586，Jun. 2001.

[43]T. Taki, J. Hasegawa, “Visualization of Dominant Region in Team Games and Its Application to Teamwork Analysis”, *IEEE proc. Computer Graphics International (CGI'00)*, Switzerland, pp.227-238, Jun.2000.

[44]I.N. Bronshtein, K.A. Semendyayev, *Handbook of Mathematics 4th ed*. Cap.3.5 Vector Algebra and Analytical Geometry, New York: Springer-Verlag, 2003.

[45]T.W. Anderson, *An introduction to multivariate statistical analysis 2nd Edition*, John Wiley & Sons, New York, 1984.

[46]有馬哲，石村貞夫，“*多変量解析のはなし*”，東京図書，1987.

[47]“Microsoft DirectX: Homepage”, http://www.microsoft.com/windows/directx/.

[48]早川啄哉，田口剛史，中野美紗子，辻 直樹，谷川 敬一郎，中村 健二，竹内 克明，“*DirectX8&VC++ 3D の基礎とゲームの作り方*”，工学社，2002.

[49]D. Mills, "Internet Time Synchronization: The Network Time Protocol", *IEEE Trans. on Communication*, vol.39, no.10, pp.1482-1493, Oct. 1991.

[50]M. Carson, D. Santay, "NIST Net - A Linux based Network Emulation Tool", *ACM SIGCOMM CCR*, vol.33, no.3, pp.111-126, Jul. 2003.