NAIST-IS-DD0661026

博士論文

大規模な仮想共有空間を介した
多人数ユーザ間インタラクションの
普及環境における実現法

山本 眞也

2009年 2月 5日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

山本 眞也

審査委員：

伊藤 実 教授　（主指導教員）
加藤 博一 教授　（副指導教員）
安本 慶一 准教授　（副指導教員）

大規模な仮想共有空間を介した多人数ユーザ間インタラクションの普及環境における実現法*

山本 眞也

内容梗概

近年，多数の仮想的なオブジェクトを配置した仮想空間を様々な用途に活用するための研究が盛んにおこなわれている．これらの研究は，共有仮想環境（Virtual Environment，VE）のような多数のユーザで仮想空間を共有する研究と，拡張現実感（Augmented Reality，AR）や複合現実感（Mixed Reality，MR）のような現実のユーザやオブジェクトの動きをモーションキャプチャなどの技術によって取り込み，仮想空間に反映する研究の，2 つに分けられる．

一般に，VE では多人数のユーザを対象とするため，高性能なサーバとそれに見合う大きな通信帯域幅を必要とし，AR や MR では，現実のユーザの動きを取り込んで仮想空間に合成するために，モーションキャプチャスーツや全方位ディスプレイ，カメラアレイなど，特殊な装置を使うことが多い．これらの技術を一般ユーザの様々な用途に応用するためには，低コストで実現できることが重要である．また，VE 技術と AR/MR 技術を融合させ，多人数で共有する空間に現実世界のオブジェクトの情報を取り込めるシステムが望まれている．本論文では，VE 技術と AR 技術の両方に基づいた大規模共有仮想空間を一般的な普及環境において実現するためのフレームワーク，および，その実現方法に関する次の 2 つのトピックについて記述する．

* 奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD0661026, 2009 年 2 月 5 日.

第一に，コストのかかるサーバを排したP2P環境で一般的なネットワーク仮想環境（Networked Virtual Environment，NVE）を実現するために，多人数参加型オンラインロールプレイングゲーム（Massively Multiplayer Online Role Playing Games，MMORPG）を対象に，P2Pオーバレイネットワークアーキテクチャの提案をおこなう．一般的なMMORPGでは，参加ユーザ数の見積りの難しさから発生するサーバの設置，運用に要するコスト，および，ユーザ数の増加にともなうサーバや通信回線の増強にかかるコストが問題となる．これを解決するために，サーバを用いずにユーザ端末による分散処理をおこない，ゲームのある部分領域の混雑状況に応じて，複数の計算機によるゲームイベント配送木（以下，**負荷分散木**）を動的に構築することにより負荷分散を図るためのイベント配送機構を提案する．本イベント配送機構では，負荷分散木が深くなることによる遅延の増大を解決するために，配送木上のノードを順次入れ替えていくことでエンド・エンドのイベント配送遅延を徐々に短縮する．また，複数の部分領域にまたがるプレイヤの視野範囲が移動していく際にもイベント配送を効率よくおこなうため，ユーザの視野より大きな範囲で，発生するイベントの受信予約をおこなうsubscribe範囲と，受信予約を解除するためのさらに大きなunsubscribe範囲を設定し，余分な制御パケットを抑制しつつスムーズな視野の移動ができる手法を考案した．提案手法の性能を確認するため，LAN環境で動作するプロトタイプシステムによる実験と，ns-2によるシミュレーション実験，PlanetLabと呼ばれる地球規模のWAN上でのプロトタイプシステムによる実験をおこなった．その結果，提案手法が現実のネットワークゲームを実現する上で，各ノードの計算負荷・通信負荷がゲームシステムのスケーラビリティやゲームプレイに影響しないこと，一般的なネットワークにおいて，ある部分領域のプレイヤ密度が高くなった場合でも，通信遅延を十分に使用可能な範囲に抑えることができることを確認した．

第二に，実空間と仮想空間の間のインタラクションを実現するためのフレームワークの提案をおこなう．本研究の主題となる実・仮想空間混合対話型アプリケーションでは，オブジェクトの動きが予め登録されているMMORPGと比べ，現実のオブジェクトの動きを仮想空間に反映するためのデータ量が格段に大きい．よって，一般的なネットワークにおける使用可能な通信帯域の制限から，第一の研究

で提案する機構をそのまま用いることは難しい．そこで，ユーザ毎に，ユーザにとってより重要なオブジェクトがより高い頻度で更新されるよう **AR 情報**（各オブジェクトの位置，向きなどの更新情報）の配送頻度を自動調整する QoS 適応機構を考案した．提案する QoS 適応機構の性能を評価するため，提案手法を適用した際の各オブジェクトのデータの更新頻度の試算及びそれを基にした動画によるユーザ満足度を評価した．また，プロトタイプシステムを作成し，ユーザが視野を移動した際に視野内のオブジェクトの表示品質が適切に調整されるまでの時間を様々なオブジェクト数に対して計測した．その結果，通常のインターネット環境および無線 LAN 環境において提案機構が実用上十分な性能で動作することを確認した．また，仮想ユーザに対しては，実用上十分なデータの更新頻度と短い遅延を達成することを確認した．

キーワード

分散仮想環境，拡張現実感，インタラクション，QoS 適応制御，オーバレイネットワーク，負荷分散

A Study to Realize Interaction between Massively Many Users through Virtual Space with Ordinary Computing Environments *

Shinya YAMAMOTO

Abstract

Recently, there are many studies regarding to Augmented Reality / Mixed Reality (AR/MR) and Virtual Environment (VE). VE requires high performance servers and high speed networks for supporting massively many users. AR and MR often require special devices (e.g. a motion capturing suit, an immersive display, a camera array, and so on) for capturing movements of users in a real world and reflecting them into a virtual space. In order to utilize VE and AR/MR for various purposes, a technology for realizing these techniques with low cost is essential. A technology for realizing virtual space systems with massively many users where movements of real users are reflected, is also expected. This thesis describes two topics about frameworks to realize VE and AR techniques with ordinary (widespread) computing environments.

First, we propose a scalable communication mechanism based on the peer-to-peer (P2P) technology for Massively Multiplayer Online Role Playing Games (MMORPG). In order to reduce the overall management cost , it is desirable to compose the game system only of player nodes in a P2P fashion without specific servers or high-speed networks. We propose a new distributed event delivery method for MMORPG. In our method, we let each responsible node monitor the number of player characters, and dynamically construct/extend a tree of multiple nodes (called *load balancing tree*)

* Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0661026, February 5, 2009.

so that the communication and computation overhead of each node in the tree does not exceed the predefined threshold by delivering event messages through the tree. A certain number of backup nodes are prepared to cope with failure of nodes in the balancing tree. Each backup node is also used to shorten the end-to-end event delivery latency while events are delivered through the load balancing tree. For the purpose, the backup node is replaced with one of intermediate nodes in the tree if the end-to-end latency improves. We also provide an efficient mechanism, where player nodes can quickly and seamlessly switch responsible nodes for subscription while the player characters move over sub spaces.

Though experiments in LAN and in PlanetLab using our prototype system and simulations with ns-2, we have confirmed that the proposed method can achieve reasonable performance in event delivery for MMORPG in practical environments. In the experiment in LAN, when the number of players is 100, the required network bandwidth at the responsible node was around 13 Mbps if the load balancing tree is not used. If the tree is used, the bandwidth was reduced to 0.8 Mbps. In the simulation with ns-2, the maximum end-to-end delivery delay was reduced by about 45 percent from the initial delay and even the maximum delay converged to below 250msec. In the experiment in PlanetLab, the CPU load at each node were regulated below 0.3 percent. Additionally, the average communication delay was regulated to around 300 msec. These results show that the proposed method can achieve practical performance in terms of overhead and end-to-end latency required for MMORPG.

Secondly, we propose a framework called FAIRVIEW which realizes cooperative work and interaction between mobile users in real world and remote network users. In our target application systems, the data amount exchanged between users is much bigger than that in MMORPG, because we would like to capture and reflect more realistic and complex movement of objects for smooth interaction. So, it is difficult to use the communication mechanism which we proposed for MMORPG as is since the ordinary internet has the strict limit of the communication bandwidth. We propose a QoS adaptation mechanism which maximizes users' satisfaction by prioritizing transfer of AR

information (information for updating each object's position and orientation) among objects based on relative position and orientation of objects in the users' view.

We have evaluated our QoS adaptation mechanism by simulation. When 5,000 objects were shared, the update frequency of the important object by our method was 50 points better than the uniform method which uniformly reduces update frequencies of objects. In a wireless communication environment where users share the same bandwidth, our method achieved 5 points better update frequency of important objects than the uniform method. The calculation delay was about 30msec when 1,500 objects were shared. In addition, the change of the update frequency with the movement of the user's view was reflected in at least 200msec. As a result, we confirmed that the proposed mechanism realizes smooth interaction involving a large number of real and virtual users/objects in an ordinary internet environment.

Keywords:

Distributed Virtual Environments, Augmented Reality, Interaction, QoS Adaptation, Overlay Network, Load Balancing

目 次

図 目 次

表 目 次

1. 序論

本論文は，筆者が奈良先端科学技術大学院大学情報科学研究科に在学中におこなった，仮想空間を様々な社会活動に対応させるための研究をまとめたものである．

仮想空間とは，座標情報とコンピュータグラフィックスからなるオブジェクトを用いた擬似的な空間のことである．仮想空間では，ユーザは**アバタ**と呼ばれるユーザの分身で空間を擬似的に動き回り，空間や空間内に配置された物体の状態を認識し，また物体の状態を変更することで，他のユーザとのインタラクションをおこなう．仮想空間を応用した研究は，大きくわけて，共有仮想環境（Virtual Environment，VE）のように多数のユーザで仮想空間を共有する研究と，拡張現実感（Augmented Reality，AR）や複合現実感（Mixed Reality，MR）などのようにモーションキャプチャなどの技術によって現実のユーザの位置情報や運動を取り込み，仮想空間に反映する研究の，2 つに分けられる．

VE 技術は，多人数のユーザで同一の仮想空間を共有し，ユーザ間のインタラクションにより作業を進めることが特徴である．代表的な研究として，ネットワークゲームのようにインターネットなどのネットワークを用いて仮想空間を共有する Network Virtual Environment（NVE）が挙げられる [1, 2, 3, 4]．また，NVE で問題となるスケーラビリティを確保するためにサーバを用いずに P2P ネットワークによって実現する Distributed Virtual Environment（DVE）も挙げられる [5, 6, 7, 8, 9, 10].

AR 技術や MR 技術は，現実のユーザの動きをモーションキャプチャなどの技術によって位置情報を取り込み，仮想空間に反映することによって，アバタをユーザの動きに連動させることが特徴である．仮想空間に入り込む感覚を提供する MR 技術では，代表的な研究として，舞台やエキストラを精巧な 3D グラフィックで描画し，モーションキャプチャ装置を付けた俳優が，HMD や全方位ディスプレイなどに描画された仮想空間を見ながらモーションキャプチャ技術によってアバタを操作し演技することで映画のリハーサルなどをおこなう MR-PreViz[11] や，サポートツールとして，任意のオブジェクトをハンドヘルドカメラによって撮影することで，そのオブジェクトの精密な 3D オブジェクトをリアルタイムに作成す

るインタラクティブ3次元モデリングシステム [12] などが挙げられる．また，現実空間を拡張することを目的としたAR技術に関する研究として，ウェアラブルコンピュータによってユーザの位置を推定し，あらかじめ登録された情報をウェブカメラから得た映像上に表示することで現実空間を拡張するWeavy[13]や，任意のマーカをウェブカメラによって撮影すると，あらかじめ登録された3Dグラフィックがマーカ上に上書き表示されるARtoolkit[14]などが挙げられる．

その他の関連研究として，作業を支援することに注力し様々な技術を活用した協調作業支援（Computer Supported Cooperative Work，CSCW）[15]や，複数のユーザを映像から抜き出し，リアルタイムに仮想空間内で合成する臨場感通信（Tele-immersion）が挙げられる [16, 17]．CSCWは，作業者同士での情報共有が難しい高所や工場などの場面において，作業者同士で位置情報，作業手順，コミュニケーションなどの情報を共有するための作業支援に関する研究である．CSCW分野では，必ずしも仮想空間を用いる必要はないが，作業手順の説明など，口頭でのコミュニケーションが難しい場面において，遠隔地からの情報共有の手段のひとつとして仮想空間技術が用いられる場合が多い．Tele-immersionは，遠隔地のユーザの運動や位置関係を映像によって取得し，その映像を3次元化し，仮想空間に配置することにより，地理的に離れた現実空間を仮想的に統合する環境を構築する研究である．

VE技術とAR/MR技術を併せて用いることで，現実空間でおこなわれている様々な社会活動（例えば，コンサート，ショッピング，展示会，スポーツ，ゲーム，旅行，講習会，会議，共同作業など）に，遠隔ユーザがネットワークを介して仮想的に参加することを可能にするアプリケーションが実現できる．しかし，既存のVE技術は，仮想的な空間・オブジェクトの共有のみを対象としているため，現実空間の様子を反映できない．既存のAR/MR技術は，多人数の参加を対象としておらず，また，特殊な装置を用いることが多いため，適用範囲は狭い．よって，従来の技術をそのまま社会活動を支援するアプリケーションの実現に適応することは難しい．以上の問題を解決するため，本論文では，(1) 多人数参加型のVEを普及環境で実現するためのP2Pオーバレイネットワークアーキテクチャ，(2) 実・仮想空間混合対話型アプリケーション構築のためのフレームワーク，の2つのト

ピックに関しておこなった研究内容を記述する．

第一に，Massively Multiplayer Online Games（MMOG）を対象とし，多数のユーザ間で，高性能サーバや高速ネットワークを用いることなく，短い遅延での実時間通信をおこなうためのネットワークアーキテクチャを提案する．現在の MMOG は，ゲーム空間上の全プレイヤの位置や状態などから構成される大域的なゲーム状態を，クライアント・サーバによる集中制御方式で管理するものが多い．集中制御方式はセキュリティの確保やゲーム状態の管理が楽な反面，サーバに負荷が集中するためスケーラビリティ，コストの面で問題を持つ．この点を補うため，MMOG における大域的なゲーム状態をプレイヤ（ノード）間のイベント（プレイヤの行動や属性の変化など）の交換により，分散処理する方式が必要とされている．このような分散処理方式を実現するためには，(1) 各ノードの計算負荷および通信量がゲームの参加人数に関係なく一定水準以下に抑えることができること，(2) 特定のノードの故障や離脱により他のプレイヤが影響されないこと，が重要になる．また，RPG（Role Playing Game）や FPS（First Person Shooter）タイプのゲームでは，(3) 実時間性，すなわち，ゲーム状態の更新間隔が十分に短いこと，が求められる．既存研究は，(1)– (3) の要件を全て満たしていない．これら (1)–(3) 全ての課題を解決するため，先行研究 [18, 19] で提案されている MMORPG（Massively Multiplayer Online Role Playing Game）を対象とした分散型イベント配送方式を拡張する．本論文では，部分領域のプレイヤ数が増加する場合に負荷分散をおこない，同時に故障ノードへの対応および遅延の少ないイベント配送を実現する方法を提案する．具体的には，本イベント配送機構では，先行研究に残された課題である配送木が深くなることによる遅延の増大を解決するために，配送木上のノードを順次入れ替えていくことでエンド・エンドのイベント配送遅延を徐々に短縮する手法を提案する．また，従来研究において対処されていなかった，複数の部分領域にまたがるプレイヤの視野範囲が移動していく際にもイベント配送を効率よくおこなうため，ユーザの視野より大きな範囲で発生するイベントの受信予約をおこなう subscribe 範囲と，受信予約を解除するためのさらに大きな unsubscribe 範囲を設定し，余分な制御パケットを抑制しつつスムーズな視野の移動ができる手法を考案した．

提案手法における負荷分散木の構築による負荷分散の性能について評価するため，LAN 環境で動作するプロトタイプシステムによる実験をおこなった．また，負荷分散木上のノードの交換による遅延短縮の効果を評価するため，ns-2[20] によるシミュレーション実験をおこなった，さらに，実環境上での本イベント配送機構の性能を評価するため，PlanetLab[21] と呼ばれる WAN 環境上でのプロトタイプシステムによる実験をおこなった．LAN 環境で動作するプロトタイプシステムによる実験では，アバタがある範囲に 100 人集中した場合でも，サーバ代わりとなるノードは 5 分木による負荷分散木によって，通信量を 13Mbps から 0.8Mbps に抑えられた．また，ns-2 によるシミュレーション実験では，負荷分散木のノード交換手法によって，35 秒程度で 45%程度の遅延を削減できた．さらに，PlanetLab での WAN 環境実験では，各ノードの CPU 利用率は 0.3%未満に抑えられ，平均通信遅延も 300ms 程度に抑えられた．これにより，提案手法が現実のネットワークゲームを実現する上で，実用的な性能を達成できることを確認した．

第二に，上記のイベント配送方式を拡張し，現実空間から参加するユーザ（以下，**実ユーザ**と呼ぶ）と遠隔地から参加するユーザ（以下，**仮想ユーザ**と呼ぶ）によるインタラクションを提供するための実・仮想空間混合対話型アプリケーションに対応したフレームワーク FAIRVIEW を提案する．実・仮想空間混合対話型アプリケーションにおいて，実ユーザおよび仮想ユーザが円滑かつリアルに協調作業できるようにするには，(1) 実ユーザと仮想ユーザが同じ空間を共有できること，(2) 各ユーザが共有空間内を自由に移動でき，それぞれの空間内の位置や向いている方向に応じて，実・仮想ユーザの区別なく同じように空間が観測できること，(3) 各ユーザは空間内へのオブジェクトの導入や，各オブジェクトに対しアクション（移動や加工）を起こすことができ，かつ，そのリアクションが他のユーザに観測できること，が望ましい．また，普及性のためには，(4) 特殊な装置や高性能・高価なサーバやネットワークによらず上記の要件 (1)-(3) が実現できること，(5) 多数のオブジェクト（ユーザを含む）が同じ空間に同時に存在できること，が望まれる．そこで，実・仮想ユーザが同時に参加できる様々な対話型アプリケーションを，普及したコンピューティング環境で実現することを目的に，要件 (1)-(5) を満たすフレームワーク FAIRVIEW（Framework for Interaction between

Virtual and Real Worlds）[22] を提案する．本研究では，共有するオブジェクトやユーザ数が非常に多いことを前提条件とする．そのため，利用可能な通信帯域に制限のある普及したコンピューティング環境では，全ユーザが全ての情報をリアルタイムに共有することは難しい．また，リアルタイム性を重視し，ユーザが利用できる通信帯域におさまるようにオブジェクトの更新品質を均一に下げると，ユーザには見えないオブジェクトの更新などに影響を受け，ユーザにとって重要なオブジェクトの更新品質まで下がってしまい，ユーザの満足度は低くなってしまう．そこで，ユーザ端末間で利用可能な帯域の範囲内で，オブジェクトとそれを観測するユーザのペア毎にデータの送信間隔を制御する QoS 適応機構を提案する．

提案する QoS 適応制御機構を実機シミュレーションにより評価した結果，一般的なネットワーク環境を用いたモバイルユーザと遠隔ユーザの環境上で，5000 個のオブジェクトを共有した場合，均一に品質を落した場合と比べ，もっとも注目するオブジェクトの更新間隔が毎秒 40 回程度，無線通信によって帯域を他のユーザと共有したネットワーク環境においても，毎秒 5 回程度向上する．また，1500 個以上のオブジェクトを共有した場合でも，計算遅延は 30ms 程度であり，ユーザの移動にともなう品質の変更は，遅くとも 200ms 以内にはユーザに反映される．これにより，実用的な更新レートおよび短い遅延時間でオブジェクトが更新されることを確認した．

以下，2 章では，本論文であつかう仮想空間の基本概念および諸定義について述べる．3 章では，提案する多人数参加型の VE を普及環境で実現するための P2P オーバレイネットワークアーキテクチャについて述べる．4 章では，実・仮想空間混合対話型アプリケーション構築のためのフレームワークについて述べる．最後に，5 章で，本論文における結論を述べる．

2. 仮想空間の基本概念および諸定義

本章では，本論文であつかう概念の説明および諸定義をおこなう．

本論文では，インターネットに接続されたシステムに参加する端末からなる論理ネットワーク（以下，**オーバレイネットワーク**）をあつかう．オーバレイネットワーク上の端末を**ノード**と呼ぶ．ノードは，論理端末であり，現実の端末は複数のノードを持つことができる．すなわち，あるユーザがシステムに参加したとき，ユーザがアバタを操作し仮想空間を共有するため，ユーザノードがオーバレイネットワークに加えられる．ユーザ端末が同時にシステムの一部（サーバなど）を担うとき，ユーザノードとして以外に，システムの処理を担当する論理ノードがオーバレイネットワークに加えられる．このとき，そのユーザの端末は1つであるが，オーバレイネットワークでは，2つのノードとして認識される．また，本論文では，ユーザのログイン/ログアウトを管理し，ログイン中の全ノードのアドレスリストを保持する**ロビーサーバ**の存在を仮定する．

仮想空間Vは，2次元または3次元の任意形状の空間と，空間内に存在するアバタの集合$A = \{a_1, ..., a_n\}$，および，その他のオブジェクトの集合$O = \{o_1, ..., o_m\}$からなる．オブジェクトは，仮想空間上での位置情報などからなる属性Sを持ち，時間tにおけるオブジェクトoの属性を$S(t, o)$であらわす．また，オブジェクトの状態を変化させるなどの仮想空間を変更する動作を**イベント**と呼ぶ．イベントには，オブジェクトの移動，他のオブジェクトへの働きかけ，オブジェクトの色，形状の変化，ユーザの行動（例えば，移動，オブジェクトを押すなど），それによるリアクションなどが含まれる．

本研究では，通信帯域幅の制約に合わせるため，各ユーザが受け取るイベントを，そのユーザにとってのイベントの重要度に応じて取捨選択するという方針を採用する．そこで，関心領域（Area of Interest，AOI）を用いる．AOI には様々なタイプがある．例えば，文献 [23] では，DIVE（Distributed Interactive Virtual Environment）における仮想空間内の各オブジェクトを中心とする一定の空間的な範囲を**オーラ**と定義している．このとき，自分のオーラと他のオブジェクトのオーラが交差したとき，オブジェクト間でのインタラクションが可能となる．テキスト，グラフィック，音声，など，複数のインタラクション要素に対し，それぞれオー

ラを設定することで，相手の姿を見ることは出来るが，声は聞こえないといった現実世界に近い自然な状態を作り出すことができるとしている．また，MASSIVE（Model, Architecture, and System for Special Interaction in Virtual Environment）では，オーラを拡張した概念として**フォーカス**と**ニンバス**についても言及している．フォーカスはユーザの可視範囲であり，ニンバスは対象の被可視範囲である．これにより，オーラに比べ，インタラクションソースの選択をよりユーザの関心に沿う形で実現することができる．

本論文では，ユーザに影響を与えないイベントの除外を行い，その後，取得するイベントの重要度を設定する．この 2 段階の過程でイベントの取捨選択をおこなうために，AOI を 2 つのレベルで設定する．仮想空間において，あるユーザ u のイベントを受信する範囲を subscribe 範囲 $Subscribe(u)$，アバタを介して見ることができる範囲を可視範囲 $View(u)$ と設定する．このとき，ユーザ u に影響を与えないイベントの除外を subscribe 範囲によっておこない，取得するイベントの重要度設定を可視範囲によっておこなう．また，ユーザ u において，仮想空間，余計なイベントの除外をおこなう subscribe 範囲，取得するイベントの重要度設定をおこなう可視範囲のそれぞれの大小関係は，$View(u) \leq Subscribe(u) \leq V$ であるとする．

3. ネットワークゲームを対象とした分散仮想環境

3.1 はじめに

本章では，先行研究 [18, 19] で提案されている手法を拡張し，部分領域のプレイヤ数が増加する場合に負荷分散を行う一方で，同時に故障ノードへの対応および遅延の少ないイベント配送を実現する方法を提案する．また，提案手法を実装し評価実験を行うことで，本手法の有効性を示す．

MMOG における大域的なゲーム状態をプレイヤ（ノード）間のイベントの交換により，分散処理するような方式を実現するためには，(1) 各ノードの計算負荷および通信量がゲームの参加人数に関係なく一定水準以下に抑えることができること，(2) 特定のノードの故障や離脱により他のプレイヤが影響されないこと，が重要になる．また，RPG（Role Playing Game）や FPS（First Person Shooter）タイプのゲームでは，(3) 実時間性，すなわち，ゲーム状態の更新間隔が十分に短いこと，が求められる．

提案方式では，文献 [5, 18, 19] と同様に，ゲームで発生するイベントの登録・通知をゲーム領域の分割によってできた部分領域ごとにある基準で選出されたノード（以下，**エリアノード**と呼ぶ）に割り当て，分散処理させる方式を採用する．また，先行研究 [18, 19] が対象としているように，ゲームの進行に伴い，各部分領域のプレイヤ数が増加しても，エリアノードの計算負荷および通信量が一定以下に常に抑えられるよう動的負荷分散を行うとともにエンド・エンドの配送遅延ができるだけ短くなるよう制御する．このための制御は，各エリアノードは，プレイヤ数とそこでのイベントの発生頻度を監視し，複数ノードからなる木 (**負荷分散木** と呼ぶ）を動的に構築し，イベント配送を負荷分散木を経由して行うことで，ノードあたりのイベント配送処理のための計算負荷および通信量が予め定めた水準を超えないようにする．本研究で新たに評価基準 (2) を満たすため，**バックアップノード**を用いて，エリアノードの故障にも対処するよう先行研究を拡張した．また，評価基準 (3) を満たすため，負荷分散木上のノードをバックアップノードと動的に入れ替えることにより，エンド・エンドのイベント配送遅延を徐々に短縮する方法を考案した．また，従来の研究で考慮されていなかった，各プレイ

方式	サーバの有無	負荷分散	ノード離脱対応	遅延対策
文献 [1]	有	○	–	–
文献 [4]	有	○	–	–
文献 [5]	無	△	×	○
文献 [7]	無	○	○	△
文献 [9]	無	△	×	○
文献 [10]	無	○	△	○
文献 [18, 19]	無	◎	△	×
提案手法	ロビーサーバのみ	◎	○	○

表 1 比較表

ヤの視界が複数の部分領域にまたる場合に，領域の境界を意識させないシームレスなイベント配送制御法を考案した．

既存手法および提案手法の要件 (1)–(3) に対する対応状況を表 1 に示す[1] [2]．

LAN 環境で動作するプロトタイプシステムによる実験と，ns-2 によるシミュレーション実験を行い，さらに，PlanetLab の WAN 環境で簡易実装の実験を行うことで，提案手法が現実のネットワークゲームを実現する上で，実用的な性能を達成できることを確認した．

3.2 提案方式の概要

本論文で対象とする MMORPG では，ゲーム空間において，局所的にプレイヤの密度が高くなることを想定する．ただし，プレイヤの密度が高くなるのは，同時にはたかだか数箇所であると仮定する．

[1] 表 1 において，各方式が項目に対して，対応が十分である場合は○（特に良い場合は◎），対応が不十分である場合は△，対応していない，もしくは言及されていない場合は×，必要ない場合は–と表記している．

[2] 文献 [1, 4] は，高性能であるが，従来通りのサーバを用いた NVE であるため，高コストである．

3.2.1 先行研究の概要と本研究での改善点

本研究グループでは，ネットワークゲーム，特に，MMORPG (Massively Multiplayer Online Role Playing Game) を対象とした，分散型イベント配送の新しい方式を提案してきた [18, 19]．本研究では，先行研究 [18, 19] におけるイベント配送機構を拡張するとともに，詳細な実装を行い，実験評価を行った．

具体的には，文献 [18, 19] の提案手法では，エリアノードをリング状に接続し，論理リングによる Chord[24] などの分散ハッシュを用いることで，領域をまたがるイベントの配送を実現している．しかし，対象とする MMORPG において，プレイヤキャラクタの移動距離は，せいぜい近隣の部分領域までであるため，本研究では，各エリアノードに仮想空間上において隣接する (上下左右斜めの 8 つの) 部分領域のアドレスを持たせることでこの機構を単純化した．特定のワープポイントなどは，別個にアドレス表を持たせることで対処する (3.3 節).

また，先行研究 [18, 19] では，ノードの離脱に対し，バックアップノードを設置することで対処している．エリアノードが離脱した際には，バックアップノードが適応されるまで親ノードに代行処理をさせ，中間ノードが離脱した際には，孫ノードの 1 つを一時的に中間ノードと見立てる手法を提案している．しかし，この手法は，アルゴリズムが複雑であり，一時的に代行させるノードに代行するだけの処理能力がない場合や帯域に余裕がない場合が考えられる．そこで，本研究では，バックアップノードに定期的に存在確認させることで，離脱に対し，即座にバックアップノードが適応できるように改善を行った (3.3.2 節).

加えて，先行研究 [18, 19] では，視野範囲がある部分領域と一致するという制限をおき，プレイヤキャラクタが別領域に移動した際に，初めてイベント配送の予約を切り替える手法を取っている．しかし，この手法では，2 つの領域の境界をまたいだ瞬間に両領域で同時発生したイベントを取り逃してしまう恐れがある．そこで，本研究では，視野範囲の移動に伴う，観測可能な部分領域群の動的切り替え制御方式を新たに提案する (3.4 節).

さらに先行研究 [18, 19] では，負荷分散木を構築する際に，エンドエンドのイベント配送遅延について考慮していなかった．負荷分散木は，再帰的に構築されるため，負荷分散木が深くなると，それに従いイベント配送遅延も大きくなり，

これが原因でリアルタイム性が失われる恐れがある．そこで，本研究では，負荷分散木を構成するノードを入れ替えることで，イベント配送遅延を短縮する手法を新たに提案する (3.5 節).

本研究は，これらの変更点に加え，実験および実装によって本研究の有効性を詳細に評価した.

3.2.2 諸定義

ゲーム空間は，背景と複数の**オブジェクト**からなるものと定義する．ゲーム空間上の，ある時刻 t における全てのオブジェクトの属性情報を，**ゲーム状態**と呼び，$GS(t)$ と表記する．$GS(t)$ はイベントの発生により刻々と変化して行く．ゲーム進行の一貫性を保証するため，一般に，時刻 t におけるゲーム状態 $GS(t)$ は，全てのプレイヤで共有されなければならない.

ゲーム空間上の，各キャラクタの現在位置からの視界に相当する部分領域を**視野範囲** と定義する．視野範囲は，プレイヤにおけるゲーム画面に相当し，基本的にキャラクタを中心とする矩形で表す．視野範囲は，仮想空間上の連続しない複数の部分領域を含んでも良い．時刻 t において，ゲーム空間上のある部分領域 v に存在する全てのオブジェクトの属性情報（部分領域 v のゲーム状態）を $GS(t, v)$ と表記する．時刻 t において部分領域 v を視野範囲内に納めている全てのプレイヤにおいて，状態 $GS(t, v)$ が矛盾なく共有されているとき，**ゲームは一貫性がある** と定義する.

3.2.3 提案方式

本論文では，ログイン/ログアウトおよびプレイヤリストを管理するサーバを用いた Hybrid P2P 環境において，3.1 節の要件 (1)–(3) を満たし，かつ，一貫性のあるゲームを実現するため，以下の，3 つの機構を提案する.

- **分散型イベント配送機構：**図 1 に示すように，ゲーム空間を複数の部分領域に分割し，各部分領域で発生したイベントの登録・通知を，エリアノードと呼ばれるプレイヤノードに処理させる.

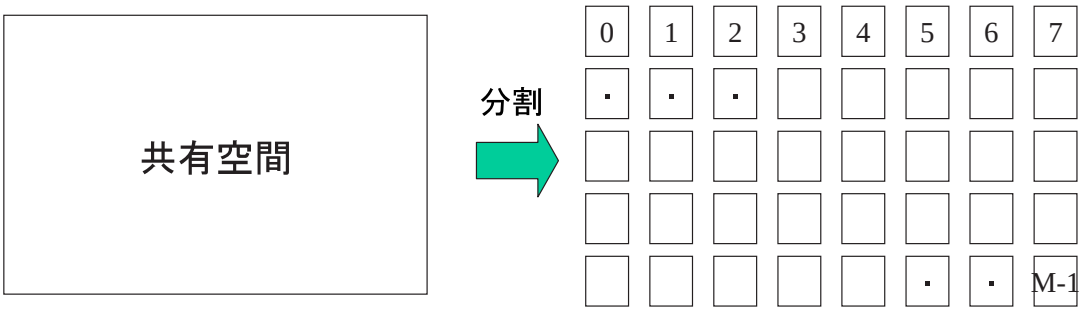

図 1 仮想空間の分割

- **視野範囲の移動に伴うイベント配送制御：**視野範囲は一般に複数の部分領域にまたがるため，プレイヤキャラクタの移動に伴い，イベントを受信したい部分領域は変わる．従って，本機構では，該当するエリアノードとプレイヤノードの間のイベント配送パスの設定切り替えをユーザに意識させることなく高速に行なう機能を実現する．
- **イベント配送処理の動的負荷分散**: ある部分領域内のプレイヤ数が増えると，エリアノードの計算負荷および通信量が増大する．このため，複数のプレイヤノードからなる木（負荷分散木）を必要に応じて動的に構築し，木に沿ってイベントを配送することで，エリアノードおよび木の中間ノードの負荷を軽減する．また，負荷分散木の構築の際には，エンド・エンド配送遅延の短縮，ノードの故障への対処法も考慮する．

上記 3 項目の詳細は，3.3–3.5 節で説明する．

3.3 イベント配送機構

図 1 に示すように，仮想空間を長方形で均等な M 個の部分領域に分割し，それぞれの領域でのイベントの登録・配送を，エリアノードに担当させる．部分領域を $v_0, v_1, ..., v_{M-1}$ と表記する．各部分領域のエリアノードは計算能力および余剰通信帯域に従って，ロビーサーバによって割り当てられる．

エリアノードは，自分のゲーム進行に問題のない程度に計算能力および余剰通信帯域に十分余裕がある場合には，同時に他の領域のエリアノード，負荷分散木上の**中間ノード** (3.5.1 節で述べる)，**バックアップノード** (3.3.2 節で述べる) としても割り当てられる．

図 1 の各部分領域 $v_i(0 \leq i \leq M-1)$ のエリアノードは，v_i において発生したイベントを，v_i を観測可能な (視野範囲が v_i と共通部分を持つ) プレイヤ (ノード) に配送する．イベント配送機構として，publish/subscribe モデル [25] を用いる．すなわち，各部分領域 v_i のエリアノードは，領域 v_i 内でのイベント発生 (publish) 時に，v_i にイベント配送予約 (subscribe) しているプレイヤノードにのみイベントを配送する．各プレイヤノードは受信したイベントをもとに，観測可能な各領域 v_i におけるゲーム状態 $GS(t, v_i)$ を更新する．プレイヤ間でゲーム状態に矛盾が生じないように，イベントの配送は**タイムスロット** と呼ばれる固定時間単位で集計して行う．また，ノードの故障・離脱などによりゲーム進行が妨げられないようにするため，各エリアノードおよび中間ノードには，バックアップノードを設置する．以下，これらの詳細を述べる．

3.3.1 イベント配送とゲーム状態の更新

各プレイヤ p の視野範囲を $V(p)$ と表記する．また，部分領域 v_i のエリアノードを $A(v_i)$ と表記する．

ゲームにおける時刻は，タイムスロットのシーケンス番号で表す．すなわち，タイムスロットの長さを Δ，ゲームの開始時刻を T_0 とすると，時間 $[T_0 + i \times \Delta, T_0 + (i+1) \times \Delta)$ を，タイムスロット t_i で表す．

提案方式では，イベントの配送は以下の手順で行う．

1. ゲーム開始時，まず，各プレイヤ (ノード)p は，$V(p)$ と重なりを持つ各部分領域 v_j のエリアノード $A(v_j)$ に対して，**subscribe メッセージ**を送信する．

2. $A(v_j)$ は subscribe メッセージを受け取ると，subscribe を受領し，領域 v_j における最新のゲーム状態 $GS(t, v_j)$ を p に返送する．

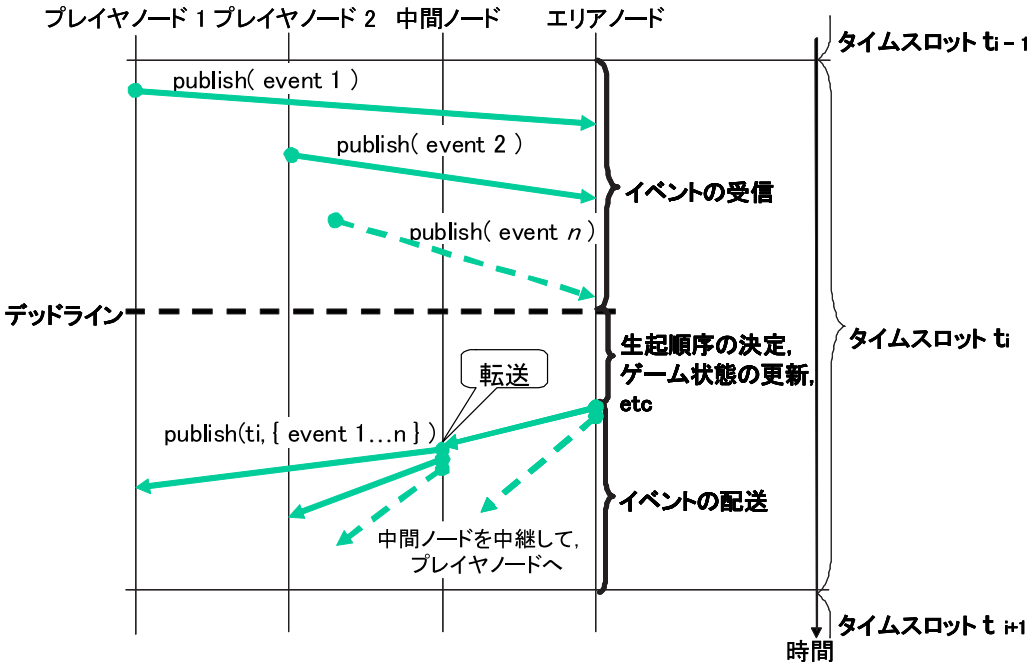

図 2 タイムスロット

3. プレイヤ（ノード）p はイベントを起こしたいとき，イベントの発生位置（p のキャラクタの現在位置など）を指定して，その位置を含む部分領域（v_m とする）のエリアノード $A(v_m)$ にイベントメッセージを送信する[3]．

4. 各エリアノード $A(v_j)$ は，現在のタイムスロット t_i におけるデッドライン（図 2）まで，複数のプレイヤノードからイベント発生要求を受信し，受け取ったイベント間に生起順序をつける[4]．生起順序に従って，ゲーム状態を $GS(t_{i-1}, v_j)$ から $GS(t_i, v_j)$ に更新し，生起順序を付けたイベントのリストを，subscribe しているプレイヤノードに送信する．

5. 各プレイヤノードは，受け取ったイベントのリストをもとに，ゲーム状態

[3] 複数の部分領域に同じイベントを起こしたい場合は，対応するエリアノード群にイベントメッセージをそれぞれ送る．この場合，イベントは別々のエリアノードで処理されるので，同じタイムスロットでイベントが処理されるとは限らない．

[4] 順序のつけ方は，乱数ベース，要求の到着時刻ベース，それらの混合などの方式が使用可能．

を $GS(t_i, v_j)$ に更新する．

提案方式では，ゲーム状態 $GS(t, v)$ の更新を，各プレイヤノードで行う．これは，1 回のタイムスロットで発生するイベントの集合 E によって，ゲーム状態 $GS(t_1)$ からゲーム状態 $GS(t_2)$ に変化するとき，ゲーム状態 $GS(t_2)$ のデータサイズよりイベントの集合 E のデータサイズの方が小さいときに有効である．

一般に，MMORPG では，部分領域でのゲーム状態のサイズが大きくなること，また，エリアノードが故障したときのゲーム状態の回復がしやすいこと，などの理由から，提案方式ではこのモデルを採用している．

なお，あるプレイヤノードが，メッセージの消失などにより，あるタイムスロットのイベントリストを受け取れなかった場合には，タイムスロット番号を付与した NACK メッセージをエリアノードに送信し，失われたイベントリスト，あるいは，ゲーム状態を送るよう要求できる．これにより，各プレイヤノードでのゲーム状態の一貫性が保たれる．

3.3.2 ノード故障への対処

ロビーサーバはプレイヤノードの中から，処理能力が高く利用可能通信帯域に余裕があるものを，予め定めた一定個数選び，そのアドレスを**バックアップノードプール** と呼ぶバッファに保持する．

エリアノードの不慮の故障，離脱に対処するため，各エリアノードに 1 つのバックアップノードを，バックアップノードプールから順次取り出し，割り当てる．部分領域 v における，各エリアノード $A(v)$ とそのバックアップノード $B(v)$ は，定期的に交信を行い，$B(v)$ は $A(v)$ が機能しているかどうかを常に監視するとともに，領域 v に対する subscribe しているプレイヤノードのリストを定期的に $A(v)$ から受け取る．バックアップノードがエリアノードと交信不能になった場合には，全ての subscribe しているプレイヤノードにエリアノードを $B(v)$ に変更するよう要請し，このうち，1 つのノード（ID が最小など）に要請して最新のゲーム状態を受け取る．以上により，エリアノードの交替が完了する．

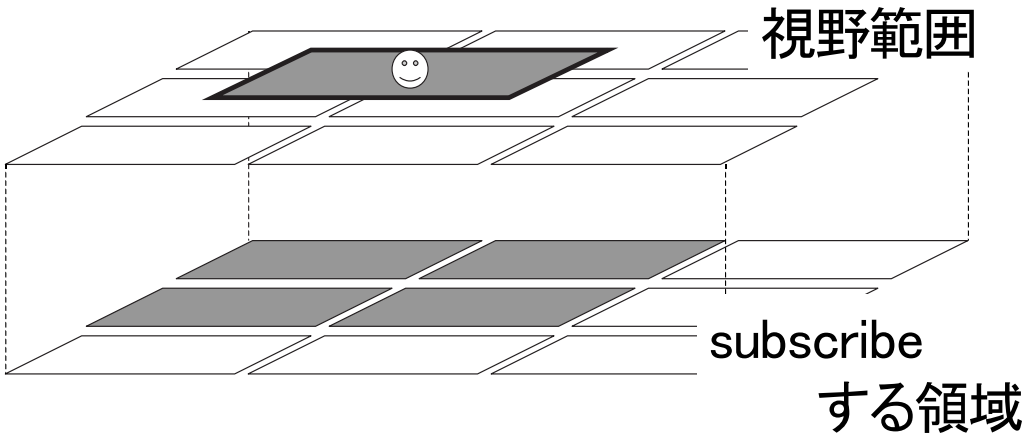

図 3 部分領域と視野範囲

3.4 視野範囲の移動に伴うイベント配送制御

ゲーム状態を複数の部分領域に分割して管理する場合，プレイヤキャラクタの移動に伴う視野範囲の変化に応じて，各プレイヤは subscribe する部分領域を動的に変更しなければならない．文献 [5] では，視野範囲がある部分領域と一致するという制限をおくことで，この制御を単純化している．しかし，この手法では，プレイヤが別領域に移動した際に，2 つの領域の境界をまたいだ瞬間に両領域で同時発生したイベントを取り逃してしまう恐れがある．文献 [7, 10] では，視野範囲が複数の部分領域を含むことを許しているが，プレイヤキャラクタの移動に伴う，観測可能な部分領域群の動的な切り替え方法については詳細に言及していない．また，観測している部分領域の数によっては，受信するイベント数が膨大になり，プレイヤの帯域を圧迫してしまう．提案手法では，視野範囲の移動に伴う，観測可能な部分領域群の動的切り替え制御は以下のように行う．

プレイヤ p の視野範囲 $V(p)$ と重なりを持つ部分領域の集合を，$\{v_{p_1}, ..., v_{p_m}\}$ とする．p は，各部分領域 $v_{p_i}(1 \leq i \leq m)$ のエリアノードに対し，subscribe を行う (図 3) ．なお，各プレイヤはゲーム参加時に，初期の視野範囲と重なりを持つ部分領域のエリアノードのアドレスを，ロビーサーバより取得するものとする．プ

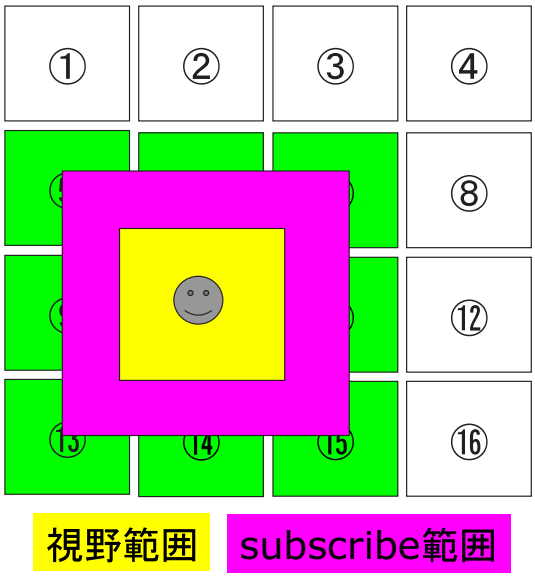

図 4 subscribe 範囲

レイヤキャラクタが移動してその視野範囲からある部分領域 v_i が外れた場合，エリアノード $A(v_i)$ に **unsubscribe メッセージ** を送信する．$A(v_i)$ はこのメッセージを受信すると，該当プレイヤノードを subscribe リストから削除する．同様に，新たな部分領域 v_j が視野範囲に入った場合，$A(v_j)$ に対し subscribe を行う．プレイヤキャラクタの移動に伴って，各部分領域 v_j のエリアノード $A(v_j)$ のアドレスを取得できるようにするため，各エリアノードには，隣接する（本論文の提案手法では，長方形の平面分割のため上下左右斜めの 8 つの）部分領域のエリアノードのアドレスを持たせる．各プレイヤノードは，subscribe の返信として，その領域の最新のゲーム状態と隣接領域のアドレスを受け取る．なお，3.3.2 節のエリアノードの交代が生じるときは，新たなエリアノードが隣接領域のエリアノードにアドレス変更を知らせる．

各プレイヤは，subscribe を行っていない部分領域からはイベントを受けとることができない．そのため，ある部分領域が視野範囲に入ってから subscribe を行なう方法では，移動と同時に視野範囲の境界付近で発生したイベントを受信できない場合が生じる．そこで，提案手法では，視野範囲を上下左右に一定の距離 d_1

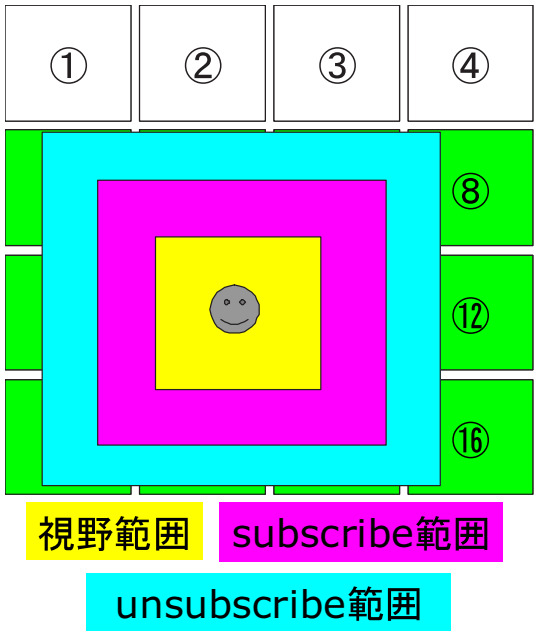

図 5 ubsubscribe 範囲

だけ広げた **subscribe 範囲**を設け，その範囲と重なりを持つ各部分領域のエリアノードにあらかじめ subscribe を行う方法を採用する (図 4)．また，subscribe 範囲の端辺と部分領域の境界が重なっているときに，プレイヤキャラクタが極めて近接した 2 地点の往復を繰り返すと，subscribe 範囲の境界に位置する部分領域群への subscribe と unsubscribe が繰り返されてしまう．この問題を回避するため，subscribe 範囲を上下左右に一定の距離 d_2 だけ広げた **unsubscribe 範囲** を設け，部分領域がこの範囲から外れた場合にのみ，その領域への subscribe を取り消すこととする (図 5)．これらの方法により，ある部分領域 v_i が視野範囲の境界と距離 d_1 以内に接近すると前もって subscribe が行われ，v_i が実際にゲーム画面に表示される前に，最新のゲーム状態 $GS(t, v_i)$ がプレイヤノードに配送される．また，プレイヤキャラクタが現在位置から距離 d_2 以内の範囲を頻繁に移動する場合にも，subscribe と unsubscribe が無駄に繰り返されるケースが減る．一般に，d_1, d_2 の値を大きくするほど効果を高めることができるが，反面，より多くのイベントが配送され通信量が増大するかも知れない．そのためトレードオフを考慮して適

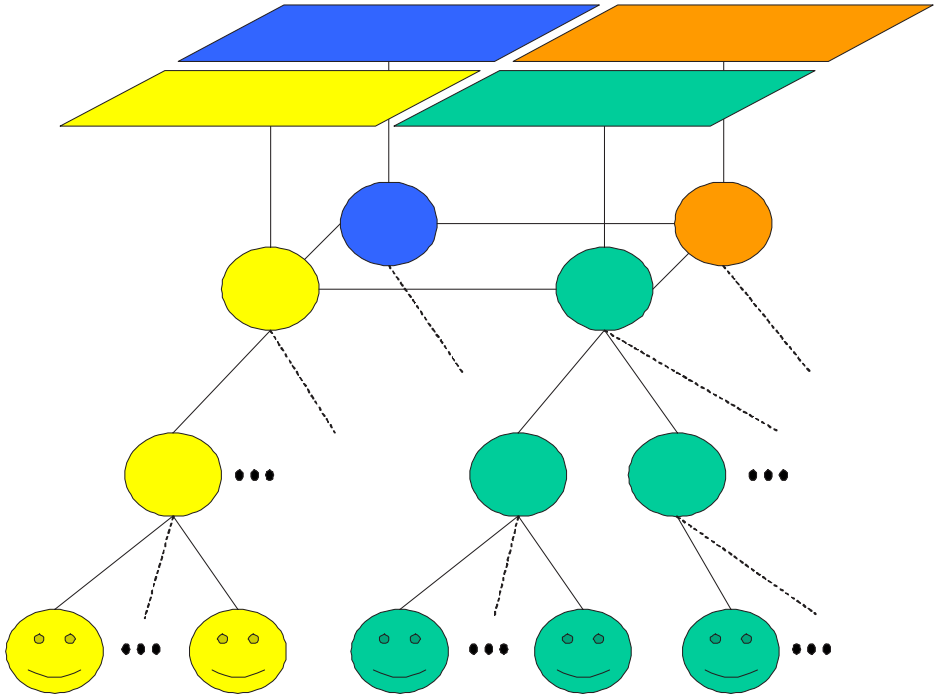

図 6 部分領域と負荷分散木

切な値を決定する必要がある.

3.5 部分領域での動的負荷分散

ある部分領域のプレイヤ数の増加に伴い，エリアノードにおける subscribe の負荷（計算負荷および通信量）が高まる．本節では，この負荷を次に述べる負荷分散木を用いて複数計算機に分散する.

3.5.1 負荷分散木の構築

ある部分領域のエリアノード a に対して，subscribe しているプレイヤノードの集合を $Player(a)$ と表記する．予め定めた閾値 C に対し，$|Player(a)| > C$ になったとき，エリアノード a は最大次数が k の負荷分散木を動的に構築する (図 6). こ

こで，C，k は定数であり，ゲームごとに適切な値が定められる．また，$C \geq k$ とする．この際，エリアノードとプレイヤノードの中継に用いるノード (**中間ノード**と呼ぶ) は，必要に応じてロビーサーバのバックアップノードプールから順次取り出して割り当てる．なお，本論文では，バックアップノードは常にプレイヤノードから補充され，バックアップノードプールは空にならないと仮定している．エリアノードは，イベントを負荷分散木に沿って配送することで，ノードあたりの負荷を軽減する．以下，負荷分散木の構築アルゴリズムの概略を示す．

（ステップ 1）エリアノード a は負荷分散木上の中間ノードの数 m および subscribe しているプレイヤの数 $|Player(a)|$ を監視し，各中間ノードのイベント配送数が C を超えないよう，$Player(a)$ を中間ノードに振り分ける．この際，subscribe しているプレイヤが増え，$|Player(a)| > m \times C$ になると，負荷分散木への新たな中間ノード（a の子ノード）の割り当てが行われる ($m := m + 1$)．逆に，subscribe しているプレイヤが減り，$|Player(a)| \leq (m-1) \times C$ になると，中間ノードを 1 つ削除する ($m := m - 1$)．

（ステップ 2）a の子ノードを $a_1, a_2, ..., a_k$ とする．中間ノードを追加すると，$m > k$ となる場合には，a の最も左の子ノード a_1 のイベント配送数を $C - k$ に減らし，新たに割り当てた中間ノードを a_1 の子ノード $a_{1,1}$ として接続する．この際，a_1 が管理する subscribe しているプレイヤの集合 $Player(a_1)$ のうち，k 個のノードを，子ノード $a_{1,1}$ に移管する．a_1 は子ノードの数 m_1 および subscribe しているプレイヤの集合 $Player(a_1)$ を管理し，$|Player(a_1)| \leq m_1 \times C$ の間は，$Player(a_1)$ の増加，減少に合わせて，（ステップ 1）と同様の方法で，子ノードの追加・削除を行う．$|Player(a_1)| > k \times C$ になったら，$a_2, ..., a_k$ に対して順次（ステップ 2）を適用する．$|Player(a_k)| > k \times C$ になったら，$a_{1,1}, a_{1,2}, ...$ に対して，（ステップ 2）を再帰的に適用する．

上記アルゴリズムにおいて，イベント配送をシームレスに行えるようにするためには，中間ノードの確保および一部の subscribe しているプレイヤの中間ノードへの移管などの操作は，subscribe しているプレイヤの数が閾値を超えてから行うのでなく，事前に行うことが望ましい．提案手法では，負荷分散木の各ノード a_i でのこれら事前操作を，例えば，条件 $|Player(a_i)| > (m_i - 1) \times C + \frac{1}{2}C$ 成立

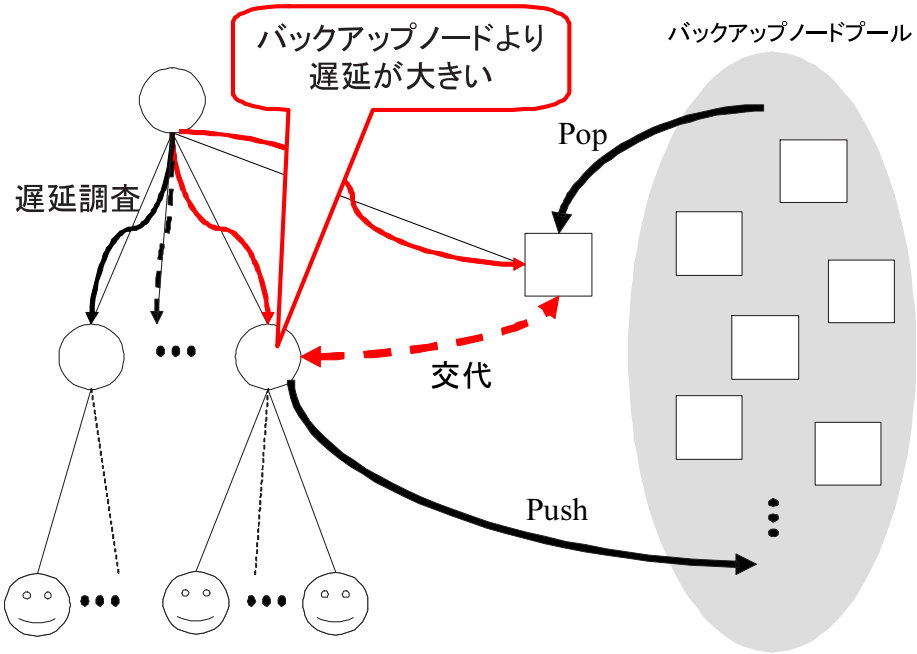

図 7 負荷分散木の更新

時に行い，実際のイベント配送は，条件 $|Player(a_i)| > m_i \times C$ が成立してから行うこととした．

3.5.2 負荷分散木経由のイベント配送遅延の短縮

負荷分散木が深くなるにつれ，イベント配送は複数の中間ノードを経由することになり，その結果，配送遅延が大きくなりゲームにおけるリアルタイム性が失なわれる恐れがある．そこで，負荷分散木を構成するノードを入れ替えることで配送遅延を短縮する．

ある部分領域 v に対する負荷分散木は，v のエリアノード $a(v)$ を根，中間ノードを節，$a(v)$ に subscribe しているプレイヤノードを葉として構成される（図 6）．このうち，葉ノードとなるプレイヤの視野範囲から v が外れるとそのノードは離脱する．一方，エリアノード $a(v)$ はゲーム上から離脱しない限りは不変である．したがって，配送遅延の改善は中間ノードを 3.3.2 節で述べたバックアップノー

ド $b(v)$ と入れ替えることで行うこととする (図 7). エリアノードや中間ノードの離脱へ対処するために用意しているバックアップノードを，イベントの配送遅延の短縮にも同時に使用できるため，遅延短縮のための余分な機構を必要としないという利点を持つ．以下，アルゴリズムについて述べる．

(1) $a(v)$ は，各タイムスロットで集約したイベントメッセージにタイムスタンプを付与して，$a(v)$ の子ノード $a_i(1 \leq i \leq k)$ に送信する．また，バックアップノード $b(v)$ に現在の subscribe リスト $Player(a)$ をタイムスタンプをつけて送信する．

(2) 各子ノード a_i は，イベントメッセージを受け取ると，メッセージの到着時刻とタイムスタンプから，$a(v)$ からの通信遅延 D_i を計算し，D_i を $b(v)$ に送信する．遅延変動を吸収するため，D_i は数タイムスロットに渡り受け取ったイベントメッセージから平均値として計算する．ここで，ノード間の時刻は NTP[26] などのプロトコルで緩やかに（最大数ミリ秒の誤差で）同期していると仮定している．

(3) $b(v)$ は，$Player(a)$ を含むメッセージを受け取ると，受信時刻およびタイムスタンプから，同様に $a(v)$ からの通信遅延 D_b を計算し記録する．また，各ノード $a_i(1 \leq i \leq k)$ から通信遅延 D_i を含むメッセージを受信すると通信遅延を比較する．$D_i > D_b$ であれば，$b(v)$ は，a_i と交代する．そのことを伝えるため $a(v)$ に交代依頼メッセージを送る．

(4) 交代依頼メッセージを受け取った $a(v)$ は，a_i に割り当てられていた subscribe リスト $Player(a_i)$ を $b(v)$ に送信し，以降のイベント配送先を a_i から $b(v)$ に変更する．$b(v)$ はメッセージを受け取ると，それ以降，a_i として振舞う．

(5) 交代完了メッセージを受け取った旧 a_i は，負荷分散木から離脱し，ロビーサーバのバックアップノードプールに追加される．また，新たな $b(v)$ がバックアップノードプールから割り当てられ，(1) からの手順を繰り返す．

なお，負荷分散木が 3 段以上の場合は，エリアノードの子ノード $a_1, ..., a_k$（負荷分散木の 2 段目）に対し一通り入れ替えが終了した後で，各 $a_i(1 \leq i \leq k)$ の子ノード（3 段目）に対し，(1)–(5) の手順を同様に再帰的に適用する．

3.6 実験と評価

提案方式を評価するため，ある部分領域のプレイヤ数を変化させて，その領域のエリアノードおよび負荷分散木の中間ノードにおける計算負荷，通信量の変化を計測した（3.6.1 節）．また，WAN 環境でのイベント配送遅延を ns-2 によるシミュレーションにより計測した（3.6.2 節）．さらに，提案手法に基づき実装したプロトタイプシステムを，PlanetLab[21] による WAN 環境で実行した際のイベント配送遅延の計測を行なった (3.6.3 節).

なお，MMORPG におけるパケットは，ノードアドレス (もしくはノード ID), 動作，アイテムの有無 (もしくは種類)，座標などからなるものとし，1 パケットの大きさを 50– 64byte と仮定した[5] ．

3.6.1 負荷分散木による CPU 利用率および通信量の変化

配送機構のプロトタイプシステムを作成し，100BASE-T の LAN 上の 8 台の PC で実行した．内訳は，プレイヤノード用の PC が 2 台 (ノード a，ノード b)，エリアノード用の PC が 1 台 (ノード A)，中間ノード用の PC が 5 台である（ノード c, d, e, f, g）．PC のスペックは，エリアノードが Pentium 4 3GHz，メモリ 1GB, その他のノードは CPU が Pentium 3/4 0.6〜3Ghz，メモリは 256〜1024MB である．OS はすべて Debian GNU Linux kernel 2.4 を使用した．コンパイラには gcc 2.95.4 を使用した．プレイヤノードの動作は，便宜上，受信用と送信用のプロセスの 2 つに分けて実装し，それぞれ，ノード a，b で実行した．ノード a は想定したプレイヤの数に見合った数のパケットをエリアノード A に送信し，ノード b はエリアノード A から（あるいは負荷分散木を経由して）送られたパケットを受信する．エリアノード A はノード a から送られてきたパケットを各タイムスロットで集約し，負荷分散木が存在するときはいくつかの中間ノードに，存在しないときはノード b に直接転送する (図 8).

[5] 文献 [27] において，リネージュのような MMORPG では，平均 10– 19 byte のイベントを送信するとあるが，これは頻繁にプレイヤキャラクタの座標を送っている為である．本研究では，設計上，プレイヤは 1 秒間に 2，3 度のイベントしか起こせないとしているため，50– 64byte が妥当であると判断した.

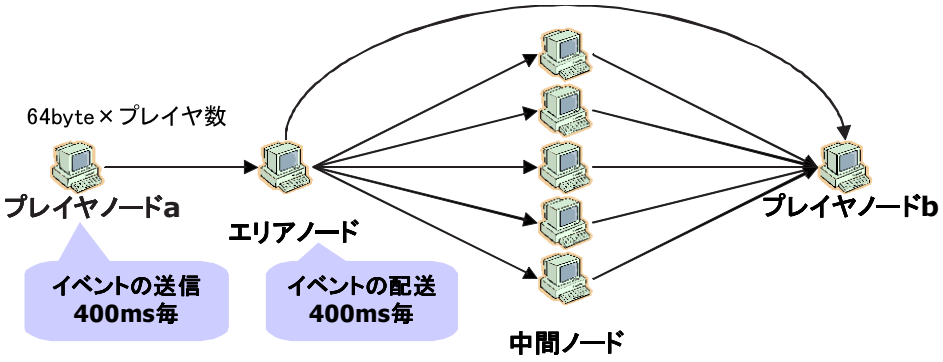

図 8 実験概要図

プレイヤノードからエリアノードに送られるイベントメッセージの大きさは 64 byte とし，エリアノードからプレイヤノードに配送されるメッセージの大きさは 64 byte にプレイヤの数をかけたものとした[6] ．負荷分散木は閾値を C=20，木の最大次数 $k = 5$ として構築した．また，タイムスロットの間隔は Δ=400ms と想定し，前半の 200ms の間，プレイヤが起こすイベントを受け取り，後半の 200ms で受け取ったイベントの集約と，全プレイヤへの送信を行うよう設定した．以上の設定のもとで，プレイヤ数を 100 まで徐々に増加させながら，エリアノードにおける CPU 利用率および最大通信量の変化と，イベント集約などにかかる処理遅延を計測した．計測結果をそれぞれ図 9，10，11 に示す．図 9，10 より，プレイヤノードの数が少ない間はその数に応じて通信量および CPU 利用率は増大している．負荷分散木を用いた場合 ($C = 20$) は，プレイヤ数がこの数を超えた時点から，CPU 利用率，通信量ともに抑えられていることがわかる．これは，負荷分散木が構築され，エリアノードはプレイヤ数÷ 20 の中間ノードに集約イベントを配送するようになったためである．このときの中間ノードの CPU 利用率は常に 0.1%以下に抑えられていた．この結果により，エリアノードのイベントの集計・配送に必要な CPU 利用率の増加は負荷分散木を用いない場合に比べ，半分以下に抑えることができ，エリアノードは負荷分散木を用いることで，より多く

[6] あるタイムスロットに，イベントを起こさないプレイヤもいるので，実際には，集約したメッセージはこれより小さくなる．

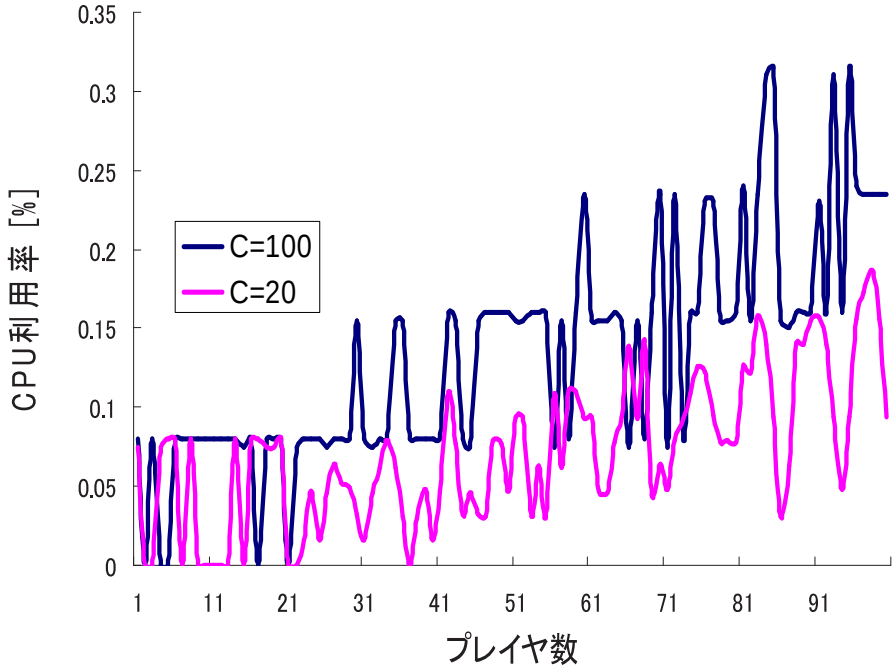

図 9 CPU 利用率の推移

のプレイヤを受け持つことができると考えられる．本研究では，プレイヤ数を全体では 1 万– 10 万人程度，一部分領域ではピーク時に 50– 100 人程度と想定している．プレイヤ数が 100 人のときのエリアノードの通信量を見ると，負荷分散木を用いない場合 (C=100) では約 13Mbps ，負荷分散木を用いた場合 (C=20) では約 0.8Mbps である．この結果から，提案手法では，プレイヤ数が増えた場合にも実用的な通信帯域に抑えることができることがわかる．ただし，C の値を小さくするとノードあたりの通信量は減少するが，負荷分散木は縦に深くなり，通信遅延は増加する．さらに，図 11 は，プレイヤ数を 100 まで増加させたときの処理遅延の計測結果である，処理遅延は，エリアノードによる 64[byte] $\times$ 100 のイベントの受信，集約，転送と，中間ノードによる集約イベントの受信および転送にかかる時間の合計である．

プレイヤ数が 100 人のときのエリアノードの処理遅延は，負荷分散木を用いな

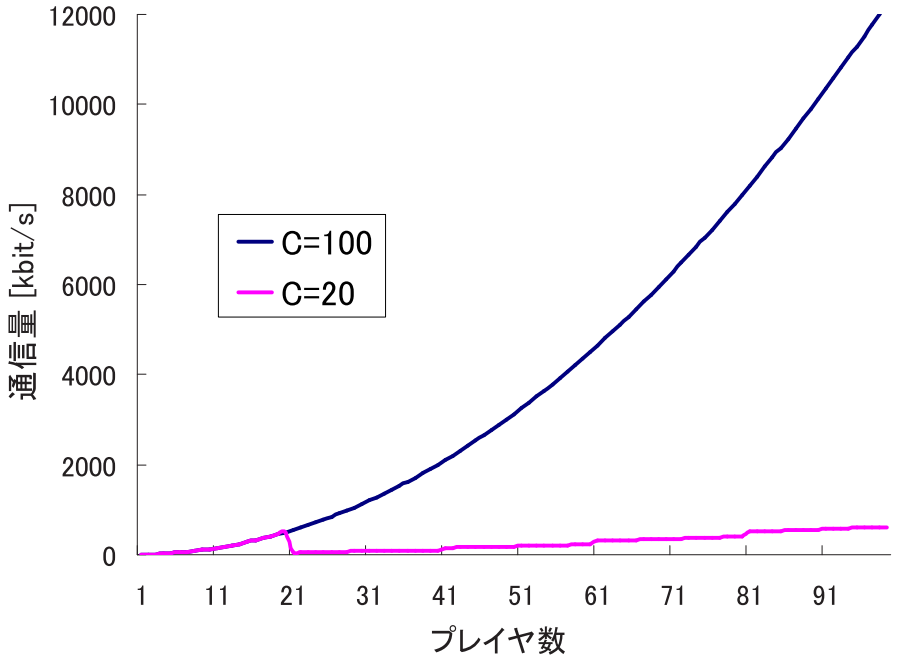

図 10 通信量の推移

い場合 (C=100) では 10ms 以上，負荷分散木を用いた場合 (C=20) では 3ms 程度である．

実験 1 の結果，負荷分散木を用いることで，CPU 利用率，通信量，処理遅延を抑制できることがわかった．しかし，中間ノードを経由することで通信遅延が増大する．よって，管理者は，これらのトレードオフを考慮して k，C の値を設定しなければならない．

3.6.2 負荷分散木のノード入れ替えによる遅延短縮の効果

3.5.2 節の配送遅延短縮手法の効果を調べるため，WAN におけるエンド・エンド配送遅延を ns-2[20] によるシミュレーションで計測した．シミュレーションでは，WAN の数を 1，WAN に接続されている MAN の数を 10，各 MAN に接続されている LAN の数を 10，とし，WAN 内のノードの数が 1，各 MAN 内のノード

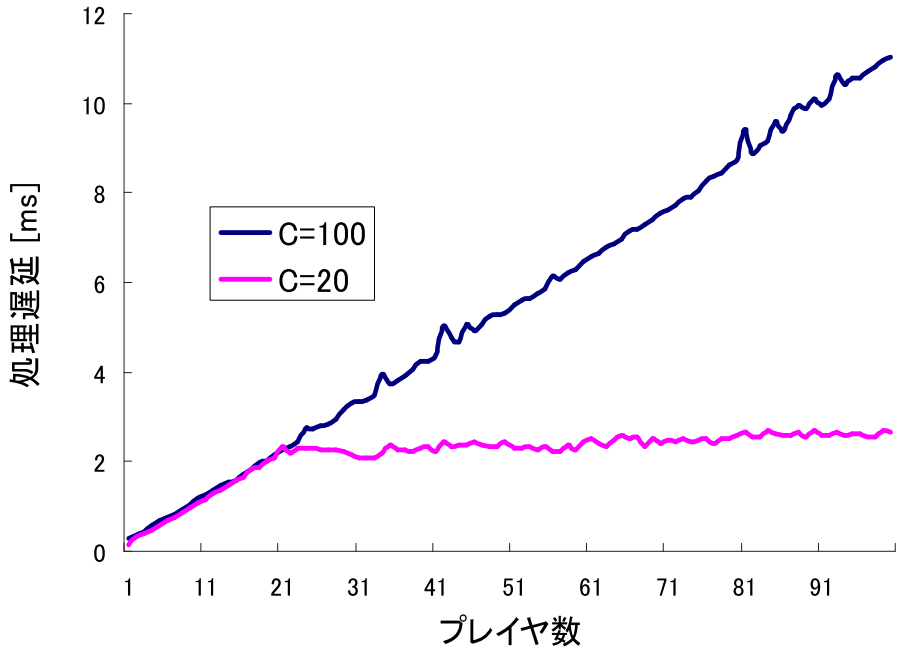

図 11 処理遅延の推移

の数が 1，各 LAN 内のノードの数が 10，であるような階層型ネットワークのトポロジを Tiers[28] を用いて生成し，各リンクの遅延時間は，LAN-LAN 間で 2ms，LAN–MAN–LAN 間が 10ms，MAN–WAN–MAN 間が 100ms となるよう設定した．この設定は，インターネットを模したものである．あるひとつの部分領域の負荷分散木を再現するため，このトポロジのエンドノードから，1 つのエリアノードおよび 125 のプレイヤノードをランダムに選択した．さらに，負荷分散木の中間ノード，バックアップノードは未使用のノードから無作為に選択した．各プレイヤのイベント発生頻度を $e = 5$ 回/秒，閾値 $C = 5$，負荷分散木の最大次数 $k = 5$ と想定した．負荷分散木は，エリアノードを根ノードとし，5 分木より 5 つの中間ノードにわかれ，さらに 5 分木より 25 個の中間ノード，さらに 5 分木より葉ノードとなる 125 個のプレイヤノードの構成になる．配送経路は，エリアノードから，2 つの中間ノードを経由し，プレイヤに送られるので，負荷分散木の高さは 4 と

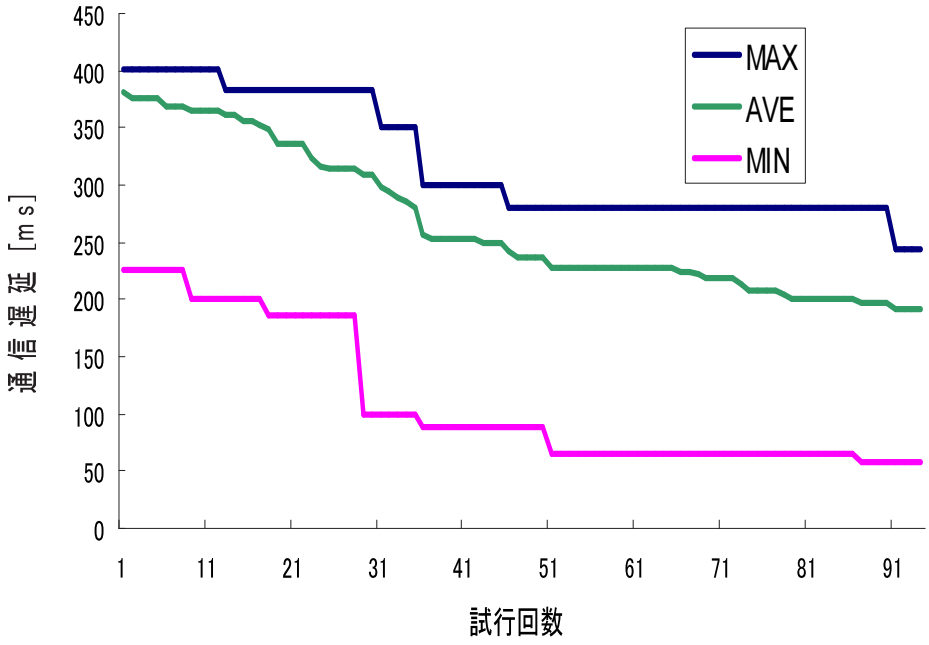

図 12 負荷分散木の更新による遅延の推移

なる．実験では，便宜上，全ての中間ノードが未更新の状態からシミュレーションを開始した．

以上の条件下で，中間ノードの更新回数に対する，エリアノードからプレイヤノードまでのエンド・エンド配送遅延の変化を計測した．実験結果を図 12 に示す．実験結果は 5 試行の平均である．図では，およそ 90 回程度の更新で配送遅延の最大 (図 12 中 “MAX”) および平均 (図 12 中 “AVE”) が最小値に収束しており，このときのエンド・エンド配送遅延は，更新しないときから 4 割程度削減されている．遅延短縮アルゴリズムは，1 タイムスロットにつき 1 回行なった．100 タイムスロットは実時間で 40 秒になる．実験結果より，35 秒程度で収束することになるが，実際の環境では，プレイヤが徐々に増えていく場合が多く，負荷分散木の構築と同時にこの手法を適応させるので，さらに短い時間で収束し，常に遅延の最大値が小さく保たれることが期待できる．

図 13 ノード分布図

3.6.3 WAN 環境上でのシミュレーション結果

WAN 環境における動作を調べるため，提案手法を PlanetLab[21] による WAN 環境上に実装し，イベント配送の遅延および各ノードの CPU 利用率を計測した．図 13 は，PlanetLab で用いることのできるノードの分布図である．実験には，これらのノードのうち，主に東アメリカのノードを利用した．時間帯や状況によってノードが足りない場合には，一時的に西アメリカ，ヨーロッパのノードを代用した．この実験では，ロビーサーバを 1 個，エリアノードを 4 個，中間ノードを 20 個，プレイヤノードを 100 個用意した．この実験では，それぞれの CPU 利用率を測るため，便宜上，それぞれのノードを全て違う PC が担当することとする．パケットサイズ，タイムスロットは，実験 1 と同じものを用いる．ゲームの設定として，まず，400x400 の座標を持つユークリッド空間をゲームで共有する仮想空間として定義し，4 つの部分領域に分割する．そして，それぞれの部分領域にエリアノードを配置しておく．それらのエリアノードに 5 つの中間ノードを割り当てておき，各エリアノードは負荷に応じて使用することとする．

ロビーサーバは，ログイン/ログアウトの管理を行う．移動距離は，1 タイムスロットにつき 50 までのランダムな値である．ログインときには，ランダムに座

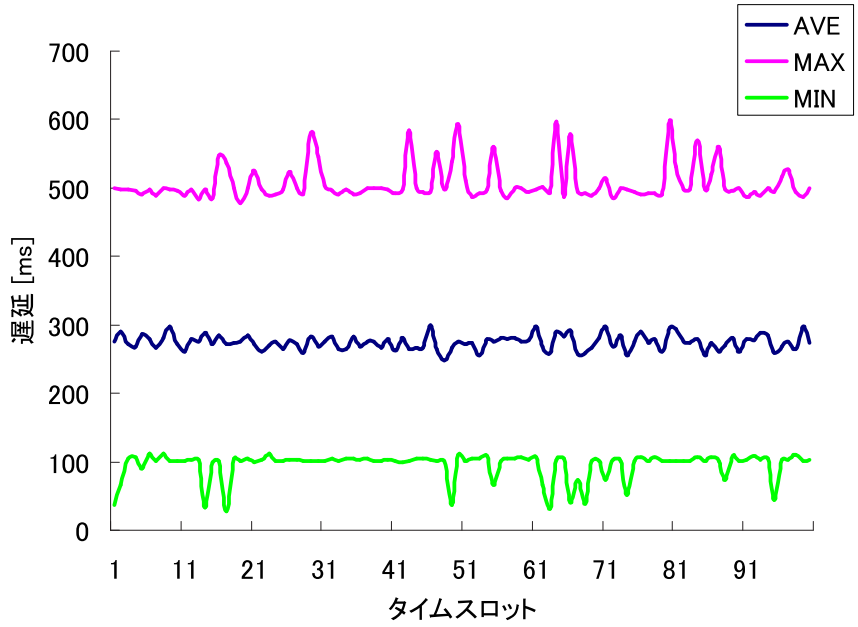

図 14 イベント配送における遅延の推移

標を決め，その場所に応じたエリアノードのアドレスをプレイヤノードに通知し，ゲームに参加させる．ゲームに参加したプレイヤノードは，ログイン/ログアウト，移動，アクション (アイテム使用や，攻撃，など) の動作を行なう．これらの動作は，9000 タイムスロット (1 時間) につき，2 回，4499 回，4499 回の割合でランダムに行なわれる．以上の設定のもとで，各ノードにおけるイベント配送遅延および CPU 利用率の変化を計測した．計測結果をそれぞれ図 14，15 に示す．

図 14 の示すイベント配送にかかる遅延の要因は，物理ネットワークによる遅延，タイムスロットの処理待ち時間，処理遅延である．タイムスロットの処理待ち時間の影響が特に大きい．なぜなら，タイムスロット開始と同時に到着したパケットは最大 200ms 待たされることになるからである．しかし，これはタイムスロット幅を小さくすることで解決できる．また，最大値 (図 14 中 “MAX”) は文献 [29] にあるプレイヤが違和感を感じるようになるとされる 400ms を越えている．しかし，実験 2 の結果より，中間ノードの交換による遅延の短縮手法を実装する

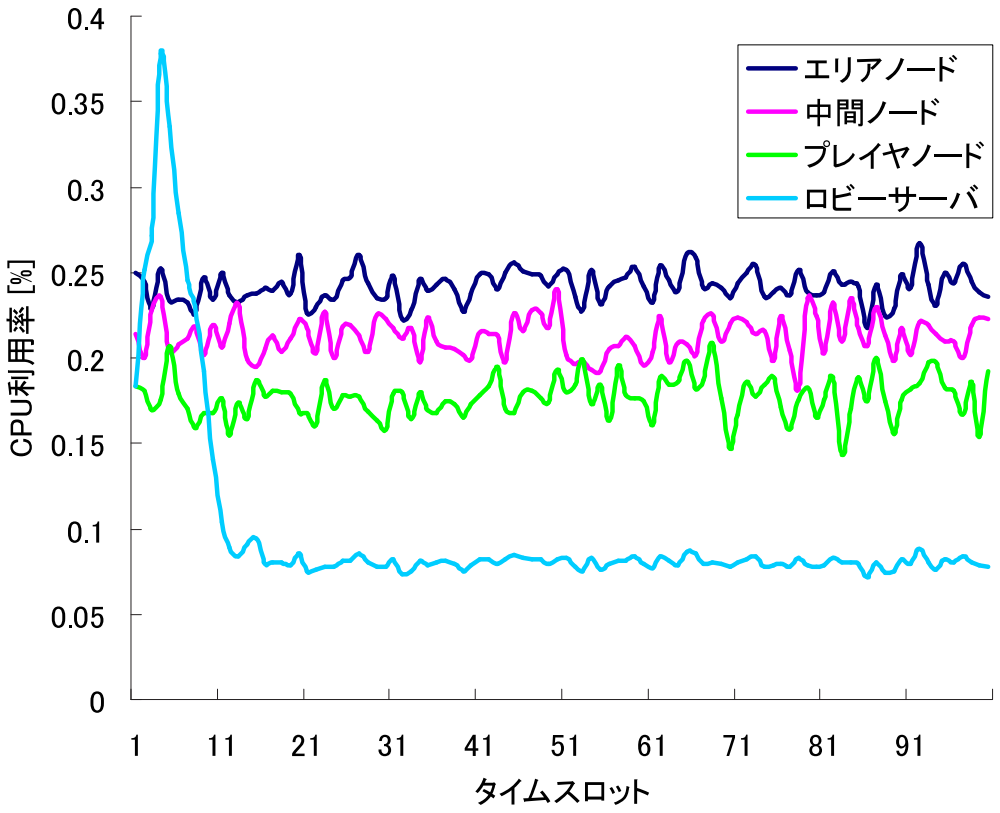

図 15 各ノードの CPU 利用率の推移

ことで，400ms 未満に抑えることが可能であると考えられる．配送遅延の平均は，300ms 未満で安定している．

図 15 は，各ノードの CPU 利用率の推移を示している．ロビーサーバの CPU 利用率は，最初の 10 タイムスロットにおいて，CPU 利用率が 0.4% になっているが，これは 100 ノードが一斉にログインしたためである．このようなことは稀であり，また，CPU 利用率もさほど高まっていない．以上のことから，ロビーサーバは一般的な PC の性能でも十分に実現可能であり，提案手法はスケーラビリティを持つと考えられる．また，その他のノードの CPU 利用率も 0.3%未満である．提案手法では，ユーザ端末に，エリアノードや中間ノードを割り当てるが，その CPU

負荷は十分に小さいと考えられる．以上の結果から，WAN 環境においても提案方式は十分に有効であると考えられる．

3.7 結論

本章では，多人数参加型のネットワークゲームを対象とした，分散型イベント配送方式を提案した．提案方式では，仮想空間を分割した各部分領域におけるプレイヤ数が増えると，エリアノードを根とする負荷分散木を動的に構築することにより，各ノードの計算負荷および通信量を与えられた閾値以内に制御する．また，同時に負荷分散木上のノードを動的に入れ替えることによって，イベント配送遅延を短縮する．

プロトタイプシステムと ns-2 でのシミュレーションによる実験により，提案方式が，各ノードの計算負荷および通信量を負荷分散木を用いない場合より小さく抑えられること，一般的なネットワークにおけるメッセージ配送遅延をある部分領域のプレイヤ密度が高くなっても，十分に使用可能な範囲に抑えることができること，を確認した．

4. 実・仮想混合対話型アプリケーション用DVEフレームワーク

4.1 はじめに

本章では，実空間と仮想空間の間のインタラクションを実現するためのフレームワークの提案を行う．実・仮想空間混合対話型アプリケーションにおいて，実ユーザおよび仮想ユーザが円滑かつリアルに協調作業できるようにするには，(1) 実ユーザと仮想ユーザが同じ空間を共有できること，(2) 各ユーザが共有空間内を自由に移動でき，それぞれの空間内の位置や向いている方向に応じて，実・仮想ユーザの区別なく同じように空間が観測できること，(3) 各ユーザは空間内へのオブジェクトの導入や，各オブジェクトに対しアクション（移動や加工）を起こすことができ，かつ，そのリアクションが他のユーザに観測できること，が望ましい．また，普及性のためには，(4) 特殊な装置や高性能・高価なサーバやネットワークによらず上記の要件 (1)-(3) が実現できること，(5) 多数のオブジェクト（ユーザを含む）が同じ空間に同時に存在できること，が望まれる．

本章で提案する FAIRVIEW では，要件 (1)-(3) を満たすため，アプリケーションの舞台となる実空間と，実空間に対応する仮想空間を用意し，両方の空間を重ね合わせる（以下，**ハイブリッド空間**と呼ぶ）．その空間では，実ユーザは仮想オブジェクトとその動作を，仮想ユーザは実空間の実オブジェクトとその動作を，それぞれ観測できる仕組みを実現する．要件 (4) を満たすため，FAIRVIEW の実行環境として，仮想ユーザはインターネット接続可能な PC のみを使用し，実ユーザは，無線通信機能を持つウェアラブルコンピュータ（HMD 含む）もしくは携帯端末を所持し，無線 AP（アクセスポイント）経由でインターネット上の仮想ユーザの PC と通信できる環境を想定する．実オブジェクトの空間における位置，向きなどの情報（以下，**AR 情報**と呼ぶ）は，既存の AR 技術を用いて短い周期で計測し，ユーザ端末間でリアルタイムに交換することで，各ユーザ毎の空間の視界をそのユーザの端末のディスプレイに 3D グラフィックスを用いて再現する．要件 (5) に対し，多数のオブジェクトが空間に存在する時も実・仮想ユーザ間の協調作業のために十分な精度でオブジェクトの動きを更新するため，AR 情報を実

時間でユーザに配送する機構（以下，**AR 情報配送機構**と呼ぶ）を提案する．AR 情報配送機構には，ユーザノード間で利用可能な帯域の範囲内で，オブジェクトとそれを観測するユーザのペア毎に AR 情報の送信間隔を制御する QoS 適応機構を組み込む．この QoS 適応機構は，ユーザの視野およびオブジェクトとの距離に応じて，どのオブジェクトがユーザにとってより重要であるかを自動判定することでオブジェクト間の重み付けを行い，より重要なオブジェクトの動作をより滑らかに表示できるよう，AR 情報の送信間隔を決定する．

提案する QoS 適応機構について評価するため，文献 [22] での各オブジェクトのデータの更新の頻度（以後，**更新レート**と呼ぶ）の試算及びそれを基にした動画のユーザ満足度に関するアンケートの評価結果に加え，プロトタイプシステムを作成し，ユーザが視界を移動した際に視界のオブジェクトの表示品質が適切に調整されるまでの時間を色々なオブジェクト数に対して計測した．その結果，通常のインターネット環境および無線 LAN 環境において提案機構が実行可能であることを確認した．また，仮想ユーザに対しては，実用上十分な更新レートと短い遅延を達成することを確認した．

4.2 関連研究

文献 [1, 6, 9] は，ネットワークゲームにおける AOI （Area of Interest）[7] の管理手法に関する研究である．これらの研究は，仮想空間上で各ユーザが観測できる領域を様々な方法で限定し，限定された領域内のユーザ間でのみ空間の状態更新のための情報を交換することで，通信量を抑え，ユーザ数に対するスケーラビリティを保つ．文献 [9] は，プレイヤキャラクタの位置関係からゲーム空間をボロノイ図に基づいて動的に分割し，同じ領域に所属するプレイヤ間で直接イベントの交換を行わせることでゲーム状態を管理する手法を提案している．文献 [1] は，空間をマイクロセルと呼ばれる小領域に分割し，各セルのプレイヤ数をもとに，複数あるサーバのそれぞれが管理するゲーム空間の範囲を動的に変更することで，ゲームのイベント処理の負荷をサーバ間で振り分ける方式を提案している．

[7] ゲーム空間全体の中で，ゲームプレイヤが関心を持っている部分空間のこと．一般には，そのプレイヤが観測することのできる視野範囲が相当する．

文献 [6] では，共有仮想空間をハニカム（六角形）構造で区切り，各ハニカムセルにおいて，各プレイヤが自身の AOI 内に存在する他のプレイヤやオブジェクトと効率よく情報交換する仕組みを，分散ハッシュテーブル（DHT: Distributed Hash Table）の 1 つである Pastry[30] を用いて実現するための手法を提案している．

これらの既存手法では，交換すべき情報をインターネットなどの普及ネットワーク環境でリアルタイムに交換できるようにするため，ユーザの観測可能エリアを制限している．しかし，本論文で対象とする実ユーザ・仮想ユーザ混合対話型アプリケーションでは，実ユーザや実オブジェクトの動きを仮想ユーザが滑らかに観測できる必要があり，そのために交換すべき情報量は，仮想空間のみを対象としたゲームなどに比べ，はるかに大きくなり，これら既存手法をそのまま適用するのは難しい．

文献 [3, 8] は DVE（Distributed Virtual Environment）における負荷分散および QoS 適応制御に関する研究である．文献 [8] では，共有空間を分割管理する既存ゲームアーキテクチャ[1, 6, 9] で発生する「境界問題」（隣り合った空間が別のサーバで管理されることによる不整合など）に対処するため，共有空間全体を一括管理する専用サーバを複数設け，共有空間での処理を臨機応変に分散する方法を提案している．この手法は高性能サーバおよび高速ネットワークを前提とし，また，実空間を対象としていない．従って，普及環境で QoS 適応制御を行うことで実ユーザ・仮想ユーザのインタラクションを実現する FAIRVIEW の目的とは異なる．文献 [3] では，多数の遠隔ユーザ間で 3D 仮想空間を共有する大規模 DVE を実現するためのネットワークアーキテクチャVESIR-6 を提案している．VESIR-6 では，ネットワーク資源の効率的な使用のために，共有オブジェクトの状態更新の配送に multicast を使用し，anycast を用いた処理の負荷分散を行うとともに，IntServ/DiffServ ベースの QoS 保証機構を用いてフロー毎の伝送レートの制御を行っている．しかし，VESIR-6 は，実ユーザ・仮想ユーザ間の空間共有で必要となる無線ネットワークでの通信を考慮しておらず，また，オブジェクトの更新はマルチキャストグループへの join/leave により管理されているのみで，ネットワークリソースの制限に合わせて更新頻度を変更するといった制御を行っていない．

地理的に離れた現実空間を仮想的に統合する環境を構築するための研究に，Tele-

immersion（臨場感通信）がある．TEEVE [17] は，遠隔地のユーザに対して，各ユーザの視点で互いの 3D 実写映像を実時間で再現し，協調作業環境を実現している．しかし，Tele-immersion 環境を構築するには 3D カメラや広帯域のネットワーク環境など特殊かつ高価な装置や設備を必要とし，普及性の面で問題がある．

4.3 FAIRVIEW の概要

本節では，FAIRVIEW が提供する機能とアプリケーション例を示し，FAIRVIEW が想定するコンピューティング・ネットワーク環境と，実現のための基本方針を述べる．

4.3.1 FAIRVIEW が提供する基本機能

FAIRVIEW は，図 16 に示すように，実空間と仮想空間を重ね合わせ，実空間のユーザと仮想空間のユーザが現実に近い感覚で協調作業できる環境を提供する[8]．FAIRVIEW が提供する主な機能を以下に示す．

機能 (1) ユーザの視野に応じた空間の観測: 仮想ユーザは，FPS（First Person Shooter） ゲームの要領で，マウスやキーボードを用いて自分の分身である**アバタ**を操作し仮想空間内を移動する．実ユーザは実空間内を普通に移動できる．その際，ユーザの位置，向きに応じて，仮想世界と現実世界の両方のオブジェクトがユーザにより観測される．

機能 (2) ユーザ間の音声会話: ユーザ間で位置関係に基づいた音声会話を可能とする．具体的には，近いユーザの声ほど大きく聞こえ，左側にいるユーザの音声は左側から聞こえるというような臨場感を提供する．この機能は，例えば，文献 [2] の手法を用いることで実現できる．

機能 (3) オブジェクトの共有: 実オブジェクトと仮想オブジェクトをハイブリッド空間に登録し，実ユーザ・仮想ユーザの双方に観測ができるようにする．また，

[8] FAIRVIEW は実空間のオブジェクトを仮想ユーザに見せることができるため，離れた複数の実空間を重ね合わせた視界をそれらの空間の実ユーザに提供することもできる．説明の簡単のため，以後は，実空間と仮想空間の間でのインタラクションに焦点を当て説明する．

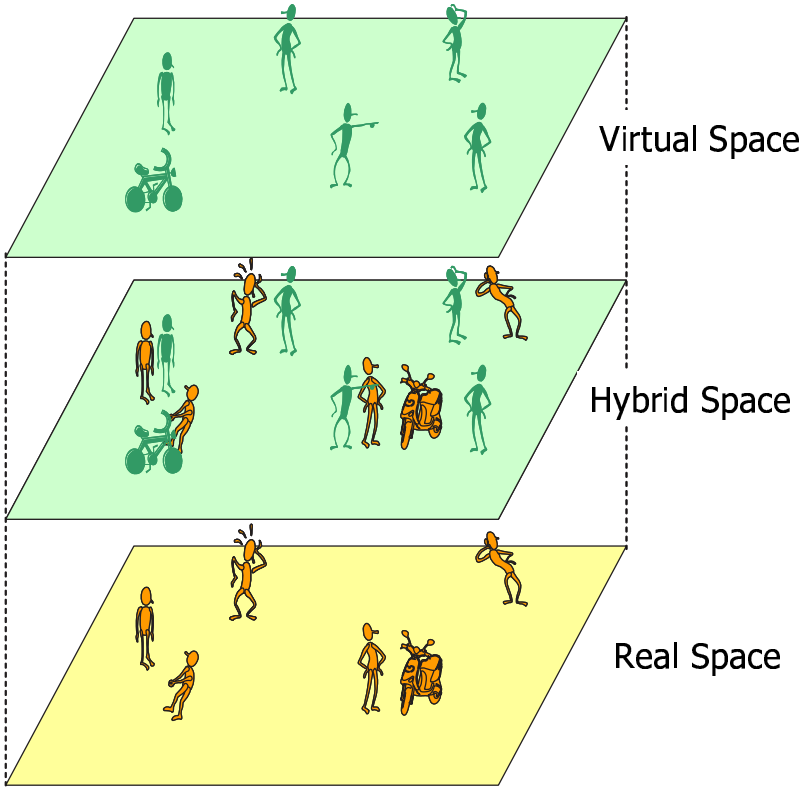

図 16 FAIRVIEW におけるハイブリッド空間

登録済オブジェクトは登録を抹消し，見えなくすることもできる．共有オブジェクトは，そのオブジェクトが視野に入っている全てのユーザにより観測される[9]．登録するオブジェクトの形状データは予めシステムに登録されているものとする．

機能 (4) 登録したオブジェクトの移動・複数オブジェクトの結合: 登録したオブジェクトに対して，近隣ユーザは**アクション** を起こすことができる．実オブジェクトは，実ユーザのみがアクションを起こすことができ，仮想オブジェクトは，実ユーザおよび仮想ユーザの両方がアクションを起こすことができる．例えば，オブジェクトを移動させると，その動きが他のユーザに観測される．アクションとして，掴む，押す，引っ張る，回す，などを対象とする．これらのアクションは，オブジェクトを登録したユーザが管理責任を負う．管理責任を持つユーザの

[9] 実オブジェクトは，登録しなくても実ユーザに見えてしまうが，便宜上，空間に存在しないものとしてあつかう．

端末は，そのオブジェクトの状態の変化を管理する．また，複数のオブジェクトを位置関係を指定して結合することができる．あるオブジェクトを移動させると，それと結合したオブジェクトが一緒に移動する．仮想オブジェクトと実オブジェクトを結合させた場合，実オブジェクトを移動させた時は仮想オブジェクトが連動して移動するが，仮想オブジェクトを移動させると，結合関係が解除される．

4.3.2 FAIRVIEW のアプリケーション

本節では，多数のユーザ間の実時間インタラクションが重要であるアプリケーションの例を 2 種類挙げ，特に必要とされる機能と各アプリケーションに特有の達成するべき要件について説明する．

蚤の市タイプ このアプリケーションでは，実空間に存在する蚤の市に，実ユーザに加えて，仮想ユーザが，売り手・買い手として参加する．例えば，実ユーザである売り手は商品（ここでは実オブジェクト）をハイブリッド空間に登録する．仮想ユーザの買い手は，蚤の市の会場を歩きながら，どんな商品をどんな人が売っているのかを視覚的に概観することができる．また，売り場では，実ユーザと仮想ユーザがお互いの存在を認識しながら音声で会話することができる．さらに，売り手は，商品を移動・回転させるなど，商品の詳細を買い手に見せながら，インタラクティブに商品の説明を行うことができる．以上は，仮想ユーザが売り手で，実ユーザが買い手の場合も実現できる．展示会や見本市，ショッピングセンターなどのアプリケーションがこのタイプに属し，FAIRVIEW により実現できる．このタイプのアプリケーションは，商品・展示品に関する共通の認識によるコミュニケーションによって成り立つため，機能 (3)，(4) が重要である．また，機能 (1)，(2) を採用することによって，よりスムーズなコミュニケーションを実現できる．さらに，これらのアプリケーションでは，多くの商品・展示品を表現する必要があるため，多数のオブジェクトをあつかえることは実現する際の重要な要件となる．

対戦ゲームタイプ 実空間でのサバイバルゲームに，仮想ユーザが参加する．典型的なサバイバルゲームでは，銃および的を持ったプレイヤが互いに対戦相手の的を撃ち，的を撃たれたプレイヤが脱落していくなか，最後まで残ったプレイヤ

が勝利をおさめる．FAIRVIEW において，各プレイヤは自分自身のアバタおよび銃，的を共有するだけで，他のプレイヤが実ユーザか仮想ユーザかを意識することなく，互いの存在および位置関係を認識し，協力したり，戦うことができる．FAIRVIEW を用いることで，実空間では登場させることができない敵キャラクタ，例えば巨大な恐竜などを登場させることが可能になる．また，実ユーザだけでは参加者が少ない場合に，参加者数を容易に補うとともに，仮想ユーザには，より現実に近いプレイ感覚を与えることができる．街角でのイベントや，テーマパークのアトラクションなどのアプリケーションがこのタイプに属し，FAIRVIEW により実現できる．このタイプのアプリケーションは，状況の変化によるインタラクションによって成り立つため，機能 (1)，(3) が重要である．また，機能 (2)，(4) を採用することによって，アプリケーションのエンターテイメント性が向上できる．さらに，これらのアプリケーションでは，状況の変化がユーザの行動に大きな影響を及ぼすため，ユーザの行動をはじめとしたイベントに対する即応性を持つことが求められる．

4.3.3 FAIRVIEW に必要なデバイスとネットワーク

表 2 ユーザの装備

Type	Computer	Display	Network	Other
Real User	PC/PDA	HMD/etc	Wi-Fi/etc	センサ，ウェブカメラ
Virtual User	PC	LCD/etc	インターネット	マウス/etc

表 2 にユーザが必要な装備を示す．実ユーザは，無線 LAN を備えた小型計算機（PDA など）と HMD（ヘッドマウントディスプレイ）を用いる．その他に，実空間中の位置や向いている方向を計測するセンサ（GPS，トラッキング装置など）およびヘッドセットなどの音声入出力機器を備えるものとする．HMD やセンサは，Wii や PS3 などのゲームコンソール向けの安価なものを流用することを想定

する．仮想ユーザは，PC，LCD などのディスプレイ装置，マウスなどのポインティングデバイス，音声入出力装置を備えるものとする．

4.3.4 FAIRVIEW 実現のための基本アイデア

AR 技術を用いるアプリケーションにおいて，ユーザが観測する視界をどのように生成し描画するかは，コストなどの面で重要な問題である．TEEVE[17] では，3D マルチカメラによりキャプチャした映像に高度な画像処理を適用し，広帯域ネットワークである Internet2 を経由して伝送する．FAIRVIEW は普及環境での実現が目的なので，各オブジェクトの AR 情報をセンサにより計測し，ユーザ端末間でリアルタイムに交換し，各端末で AR 情報をもとにオブジェクトを 3D 描画する方法を採用する．本方式に関して，解決すべき問題は以下の 3 つである．

(i) 実オブジェクトを登録し，それを移動させた際に，空間に対するそのオブジェクトの AR 情報を正確に計測すること

(ii) ユーザ端末で，オブジェクトおよびその動作を違和感なく描画すること

(iii) 普及ネットワーク環境で，AR 情報をリアルタイムにユーザ端末に届けること

上記 (i) の実現において，位置の計測は，屋外では GPS が利用でき，屋内では，無線 LAN の AP を用いた手法 [31] や音源とマイクを用いた手法 [32]，Place Lab [33]，Weavy [13] などが利用できる．また，オブジェクトの向きや傾きについては，姿勢推定機器によって計測するか，ARToolkit [14] や，文献 [12] で提案されている画像処理ベース計測手法を用いることができる．

上記 (ii) については，3D グラフィックスをリアルタイムに描画できる性能を持ったユーザ端末を用いる[10] ．

上記 (iii) は本論文があつかうメイントピックであり，その実現のためには，無線ネットワーク上の実ユーザの端末とインターネット上の仮想ユーザの端末がリ

[10] 近年，3D グラフィックス描画機能を持った携帯電話端末や PDA は標準的であり，普及している．

アルタイムに AR 情報を交換できる通信機構（AR 情報配送機構）が必要である．4.3.2 節で述べた蚤の市や対戦ゲームのようにハイブリッド空間に多数のオブジェクトが存在するアプリケーションでは，単位時間当りに交換すべき AR 情報の総量が大きくなり，利用可能帯域内に収まらなくなる．そのため，単位時間当りに AR 情報を計測・配送する頻度を下げるなどの方策が必要である．しかし，FAIRVIEW が対象とするインタラクティブなアプリケーションでは，AR 情報の計測・配送頻度を極端に下げると，インタラクションの効果が大きく損なわれる．そこで，ユーザ・オブジェクトの位置関係，向きなどに応じて，オブジェクト間で AR 情報の更新頻度を重み付けし，制約のある中で，できるだけユーザ間のインタラクションに関するユーザの満足度が高くなるような QoS 適応機構を考案する．

4.4 AR 情報配送機構

本節では，AR 情報配送機構について述べる．

4.4.1 諸定義

主な諸定義を以下に示し，表 2 にまとめる．R を実空間，V を仮想空間とする．$H = (R, V)$ を R と V を重ねることによって生成されるハイブリッド空間とする．ハイブリッド空間 H は 3D 座標系の x-y 軸で並べられた直方体であると仮定する．$RO = \{ro_1, ..., ro_n\}$, $VO = \{vo_1, ..., vo_m\}$, $RU = \{ru_1, ..., ru_l\}$, $VU = \{vu_1, ..., vu_k\}$ をそれぞれ，R の実オブジェクトの集合，V の仮想オブジェクトの集合，実ユーザの集合と仮想ユーザの集合とする．また，ユーザを含めた全オブジェクトの集合を AO とする．$node(u)$ を，ユーザ $u \in RU \cup VU$ のユーザ端末とする．このとき，$RN = \{node(u) | u \in RU\}$ および $VN = \{node(u) | u \in VU\}$ は，それぞれ実ユーザ端末の集合と仮想ユーザ端末の集合である．さらに，オブジェクト $ao \in AO$ の AR 情報を $AR(ao) = (pos, angle)$ と表記する．ここで，$ao.pos$ は ao の位置，$ao.angle$ は ao の各座標軸に対応する回転角度である．また，AR 情報に色，形状などの属性情報 $attribute = \langle color, form, ... \rangle$ や他のオブジェクト

表 3 諸定義

	表記	定義
実空間	R	
仮想空間	V	
ハイブリッド空間	H	$H = (R, V)$
実オブジェクト	RO	$RO = \{ro_1, ..., ro_n\}$
仮想オブジェクト	VO	$VO = \{vo_1, ..., vo_m\}$
オブジェクト	o	$o \in RO \cup VO$
実ユーザ	RU	$RU = \{ru_1, ..., ru_l\}$
仮想ユーザ	VU	$VU = \{vu_1, ..., vu_k\}$
ユーザ	u	$u \in RU \cup VU$
ユーザ端末	$node(u)$	
実ユーザ端末	RN	$RN = \{node(u) \mid u \in RU\}$
仮想ユーザ端末	VN	$VN = \{node(u) \mid u \in VU\}$
全オブジェクト	AO	$AO = RO \cup VO \cup RU \cup VU$
ユーザを含むオブジェクト	ao	$ao \in AO$
AR 情報	$AR(ao)$	$AR(ao) = \{pos, angle\}$
属性情報	$attribute$	$attribute = \langle color, form, ... \rangle$
アクション	$action$	$action = \langle type, direction, strength, ... \rangle$, $type \in \{push, pull, ...\}$
AR イベント	$ARe(ao)$	$ARe(ao) = (AR(ao), attribute, action)$

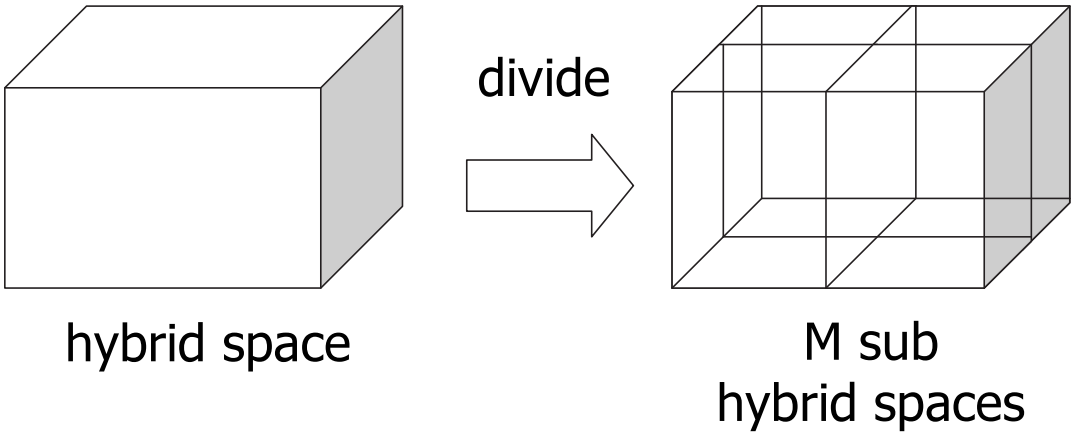

図 17 ハイブリッド空間の分割

へのアクション $action = \langle type, direction, strength... \rangle, type \in \{push, pull, ...\}$ を付加したものを AR イベント $ARe(ao) = (AR(ao), attribute, action)$ と表記する．

本論文では，各実ユーザ ru が 4.3 節で説明した装備を持ち，そのユーザの AR 情報を 1 秒あたり 60 回測ることができ，また，ARToolkit とウェブカメラによって，各実オブジェクト ro の AR 情報を測定できると仮定する．

4.4.2 ユーザ同士のコミュニケーションにおける仮定

簡単のため，実空間 R 全体がインターネットに接続している 1 つの AP によってカバーされると仮定する．BW_{AP} を実ユーザ端末と AP の間で利用できる帯域幅であるとする．このとき，すべての実ユーザ端末がこの帯域幅を共有することになる．各仮想ユーザ端末 vn に関して，$bw_{AP}(vn)$ を vn と AP の間で利用できる帯域幅であるとする．各実ユーザ端末 rn について，$bw(vn, rn)$ を vn と rn の間で利用できる帯域幅とする．このとき，$bw(vn, rn) = Min(BW_{AP}, bw_{AP}(vn))$ である．また，仮想ユーザ端末 $vn1, vn2$ に対して，$bw(vn1, vn2)$ を $vn1$ と $vn2$ の間で利用できる帯域幅とする．

4.4.3 AR イベント配送機構

ユーザの AOI に基づき，AR イベントの配送処理・配送量のノードあたりの負荷を減らすため，図 17 に示すようにハイブリッド空間 H を小さな直方体の部分領域に分け，各部分領域に**エリアノード（area node）**と呼ばれるサーバノードを割り当てる．これは，文献 [34] のような既存の P2P ベースの MMOG ゲームアーキテクチャと同様のアプローチである．an_A を部分領域 A に割り当てられるエリアノードとする．エリアノード an_A は，A 内のオブジェクトの AR イベントを受信すると，A や近隣の部分領域でそのオブジェクトを観測しているユーザにその AR イベントを配送する．

FAIRVIEW は，オブジェクトの移動に加え，向きや傾きなどの細かな動きも再現することを目指しているため，処理・伝送する情報量が MMOG に比べ多い．また，ネットワークの資源の制約に合わせて各ユーザに配送する情報量を調整する必要がある．そのため，エリアノードに加え，**帯域制御ノード（bwc-node）**を導入し，ハイブリッド空間の状態の維持・管理を，これらのノード群に分散処理させる．帯域制御ノードは，各仮想ユーザ端末 vn および AP に対し用意される[11]．$bn(u)$ をユーザ u のための帯域制御ノードとする．$bn(u)$ は，$bw(bn(u), node(u))$ をモニタし，an_A から $node(u)$ への AR イベントのストリームを QoS 適応制御する．

図 18 に，FAIRVIEW によるオーバレイネットワークの概要図を示す．これは 1 つの部分領域に対応している．この部分領域に属するエリアノードを 1 個，ユーザノードを N 個，帯域制御ノードを M 個とする．以下，この図を用いて AR イベントの配送手順を説明する．

FAIRVIEW は，次のような機能を持つ：(1) ユーザ u は，オブジェクト ao より，自身も他のユーザに観測される，(2) u は，その視界にある他のオブジェクト ao を見ることができる，(3) u は，他のオブジェクト ao にアクションを起こすことができる．

上記 (1) のために，$node(u)$ は連続的に u の AR 情報を測っており，それが以前

[11] FAIRVIEW では，無線帯域幅の制限のため，それぞれのオブジェクト ao を全ての実ユーザが同じ更新レートで観測するように制限する．

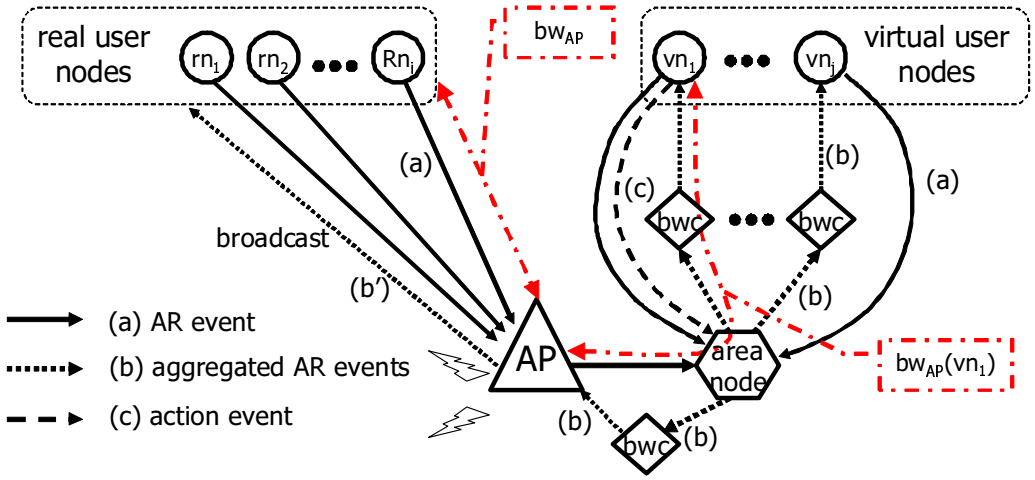

図 18 オーバレイネットワーク

の測定値と異なるならば，$node(u)$ はそれを AR イベントとしてエリアノード an_A に送る（図 18(a)）．

上記 (2) のために，$node(u)$ は，u の視界にあるオブジェクト ao の AR イベントを受信し，オブジェクトの最新の状態を $node(u)$ のディスプレイに表示する．$node(u)$ が AR イベントを受信するために，publish/subscribe 方式 [20] を用いる．$node(u)$ が，エリアノード an_A に AR イベントを送ったとき，an_A は AR イベントによって u の視界のオブジェクト ao を確認する．そして，an_A は帯域制御ノードを通して，$node(u)$ にオブジェクト ao の AR イベントを配送する（Fig. 18 (b)(b')）．

図 18 (b') に示すように，実ユーザ端末は，実空間 R に配置された AP からのブロードキャストにより，帯域制御ノードから AR イベントを受信する．

上記 (3) のために，u が仮想オブジェクト o にアクションを起こすとき，$node(u)$ は an_A に，力の強さ・向きと動作が記述されたアクションを含む AR イベントを送信する（Fig. 18 (c)）．

全体の処理の流れは以下のようになる．

Step(1) ユーザノード $node(u)$ は，AR イベント $ARe(u)$ をエリアノード an に向けて送信する．このとき，ユーザノード $node(u)$ は，オブジェクト o の管理

責任を持つなら，オブジェクト o の AR 情報を取得し，AR イベント $ARe(o)$ を作成し，エリアノード an に向けて送信する．

Step(2) エリアノード an は，4.5 節の QoS 制御手法に従い，AR イベント $ARe(ao)$ に重要度を付加する．

Step(3) エリアノード an は，重要度を付加した AR イベント $ARe(ao)$ を帯域制御ノード bn に送信する．

Step(4) 帯域制御ノード bn は，定期的に 4.5 節の QoS 制御手法に従い，重要度を用いて帯域 $bw(node(u), bn)$ の再分配を行い，それぞれの AR イベント $ARe(ao)$ に対して更新レートを決定する．

Step(5) 帯域制御ノード bn は，Step(4) で決められた更新レートに従い，送信する AR イベント $ARe(ao)$ をまとめ，ユーザノード $node(u)$ に送信する．

Step(6) ユーザノード $node(u)$ は受信した AR イベント $ARe(ao)$ を基に，仮想空間のデータを更新する．

Step(7) ユーザノード $node(u)$ は，更新されたデータを基に，周期的に仮想空間の描画を行う．

Step(8) Step(1) に戻る．

4.5 視界に基づく QoS 適応制御

通信量を利用できる帯域幅以内に抑える手法として，不必要だと思われるデータを推定し送信しない方法や情報の精度を下げてデータサイズを小さくする方法が考えられるが，本論文では，各ユーザがより注目したいオブジェクトを，高品質で見せるための制御をフレキシブルに行うために，以下の手法を採用する．QoS 適応機構の基本アイデアは以下の通りである：(1) オブジェクトがユーザにとってどれくらい重要かについて，各オブジェクトの相対的な**重要度**を決める．そして，(2) 各ユーザのために，通信量の合計が利用できる帯域幅より小さくなるように，重要度に基づいて観察しているオブジェクトの AR イベントの通信量を管理する．

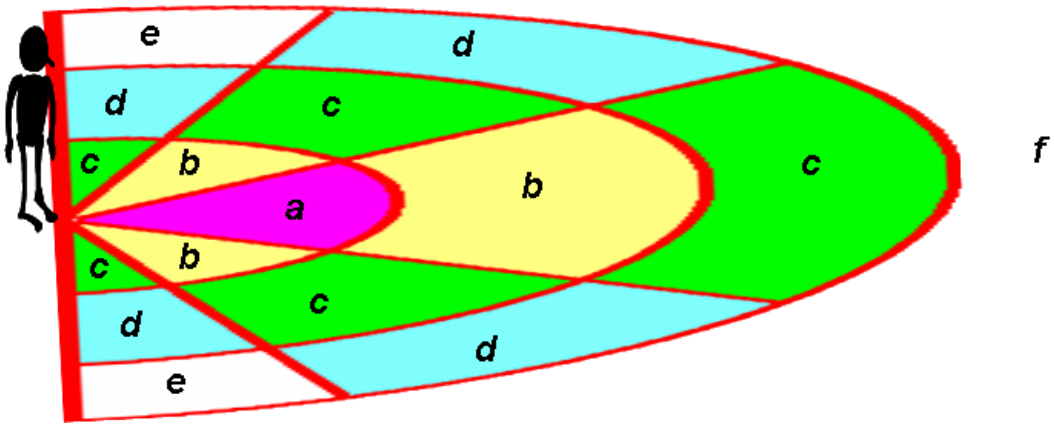

図 19 ユーザの視界と領域ごとの重要度

$Watcher(o)$ を，オブジェクト o を観察できるユーザの集合とする．このとき，$Watcher(o)$ は，以下のように定義できる．

$$Watcher(o) \stackrel{def}{=}$$

$$\begin{cases} \{u|u \in RO \cup VO, o.pos \in View(u)\} & (o \in VO) \\ \{u|u \in VU, o.pos \in View(u)\} & (o \in RO) \end{cases}$$

4.5.1 オブジェクトの重要度の決定

ここで，$View(u)$ は，ユーザ u のハイブリッド空間 H における視野範囲（図 19 の半円全体）を表し，$o.pos$ は o の H 上の位置を表す．各ユーザ $u \in Watcher(o)$ について，o が u にとってどの程度重要かを表す値を重要度と呼び，$Imp(o,u)$ と表記する．このとき，$View(u)$ の形状および重要度の割り当ては，サービス提供者がアプリケーション毎に最適なものにカスタマイズするのが理想であるが，本論文では，ひとつの例として，扇型の視野範囲を設定し，近くのオブジェクトほど，また，視野範囲の中央のオブジェクトほどより詳細に観察できるような視

野の分割を用いて，重要度の割り当てを行う．現実世界の場合に倣うと，重要度 $Imp(o,u)$ は，ユーザ u とオブジェクト o の距離が近いほど，また o が u の視野の中央にあるほど高いのが自然である．視野範囲外のオブジェクトの重要度は 0 にする．すなわち，重要度は，視野におけるオブジェクトの距離と位置により決定する．そのため，提案手法では，図 19 に示すように，ユーザの視野範囲を a–e の 5 つの部分領域からなる半円で表し，部分視野範囲 a に存在するオブジェクトの重要度は最も高く，b，c，d，e の順に，それらの部分視野範囲に存在するオブジェクトの重要度は低くなる．つまり，アルファベット順に小さくなっていき，半円の外の領域 f では 0 となる．

4.5.2 AR 情報配送の更新レートの決定

各オブジェクトからユーザ u に対して配信される AR イベントの通信量を，u の視界の全てのオブジェクトの重要度の合計に対するその重要度の比率に基づいて決定する．このとき，$bw(bn(u), node(u))$ の通信量に収まるように各オブジェクトの更新レートが削られる．提案手法では，更新レートを削るために，帯域制御ノードにおける AR イベントのパケットドロップを行う．以下では，オブジェクトの重要度に基づいた QoS 適応制御の例について，AR 情報を受け取るユーザが，仮想ユーザである場合と，実ユーザである場合に分けて説明する．

仮想ユーザに対する QoS 適応

仮想ユーザ v の端末 $node(v)$ が，v の視野範囲にある 3 つのオブジェクト o_1, o_2, o_3 の AR 情報を受け取る場合を想定する．AR 情報が v に対して割り当てられた帯域制御ノード bn_v を経由して，ユーザ端末 $node(v)$ に受信されるとする．なお，bn_v と $node(v)$ の間で利用可能な通信帯域は 1Mbps であると仮定する．今，$on(o_1), on(o_2), on(o_3)$ が送信する AR 情報配送に必要な伝送速度がそれぞれ 0.5Mbps, 総量は 1.5Mbps とする．この場合，オブジェクト o_1, o_2, o_3 の重要度の比に従い利用可能帯域 1Mbps を比例配分する．o_1, o_2, o_3 の重要度がそれぞれ 10, 25, 15 とすると，それぞれ，$0.2Mbps, 0.5Mbps, 0.3Mbps$ の帯域が割り当てられる．この結

果をもとに，それぞれ割り当てられた帯域と AR イベントの平均データサイズから 1 秒間に送れるパケット数を算出する．このとき，帯域制御ノード bn がユーザノード $node(u)$ に対し，1 秒間に送るパケット数を更新レートとする．更新レートは，

$$\text{AR 情報の計測回数} \geq \text{更新レート} \geq 0$$

である．この更新レートをもとに，帯域制御ノード bn で，パケットドロップを行うことで，各オブジェクトの AR イベントパケットの通信量を調整する．

実ユーザに対する QoS 適応

実ユーザ r の端末 $node(r)$ が，r の視野範囲にあるこの場合，AR 情報が無線 AP に割り当てられた帯域制御ノード bn_W およびその無線 AP を経由して，ユーザ端末 $node(r)$ に受信される．このとき，実ユーザ r に観測される仮想オブジェクト o の重要度 $Imp_r(o)$ を，以下のように定義する：

$$Imp_r(o) = Max_{r \in Watcher(o)}(Imp(o, r))$$

このように定義するのは，実ユーザらの重要度は無線 AP（およびその帯域制御ノード）によってまとめられるため，r の重要度は o を観察している実ユーザ全員の重要度における最大値と同じになるためである．

帯域制御ノード bn_W は，実ユーザの AR イベントの通信の際に QoS 適応制御をする．各オブジェクトの更新レートは，4.5.2 節と同じように，帯域を比例配分することで決定する．

4.6 実験

本論文で提案したフレームワークの有効性を評価するため，オブジェクト数に対するトラフィック量およびユーザが見るオブジェクトの更新レートをシミュレーションにより計測した．また，作成したプロトタイプシステムを用いて，重要度

の算出および重要度による QoS 適応制御を行った場合にかかる処理遅延，AR イベントを送信し，受信するまでのエンド・エンド遅延を，LAN ネットワーク上で計測した．

4.6.1 実験環境

表 4 実験環境

設定項目		実験 4.6.2, 4.6.3	実験 4.6.5
ユーザ数	実	100	–
	仮想	100	1
ハイブリッド空間の広さ（m^2）		50×50	6×6– 120×120
帯域（$Mbps$）	有線	10	–
	無線	5	5
仮想オブジェクト数		50– 5000	8– 1683
パケットサイズ		32	32– 64
最大更新レート（$frame/sec$）		60	30

各実験における，設定を表 3 に記す．フィールドの大きさは，運動場を想定し，$50 \times 50[m^2]$ とする．また，図 19 のように，ユーザの視界は，各角度が $\pi/3$，各領域の距離が 5m となるように，15m の半円を分割した．これは，全体のオブジェクト数が 50 個の時，視野に入るオブジェクトが，1 つから 2 つ，かつ，整数になるように設定している．また，領域 a, b, c, d, e の重要度は，それぞれ，64，16，4，2，1 とした．これは，図 19 において，領域 c の面積が最も大きいため，領域 e から，それぞれ 2 倍毎に設定すると，領域 c が領域 a,b より，オブジェクト数で上回ってしまうため，効果的に重要度の効果を得ることができない．よって，領域 c を基準に，領域 c より遠い領域の重要度は 1/2 倍毎，領域 c より近い領域の重要度は 4 倍毎に重要度を設定した．特別な記述がない限りは，この設定を用いる．

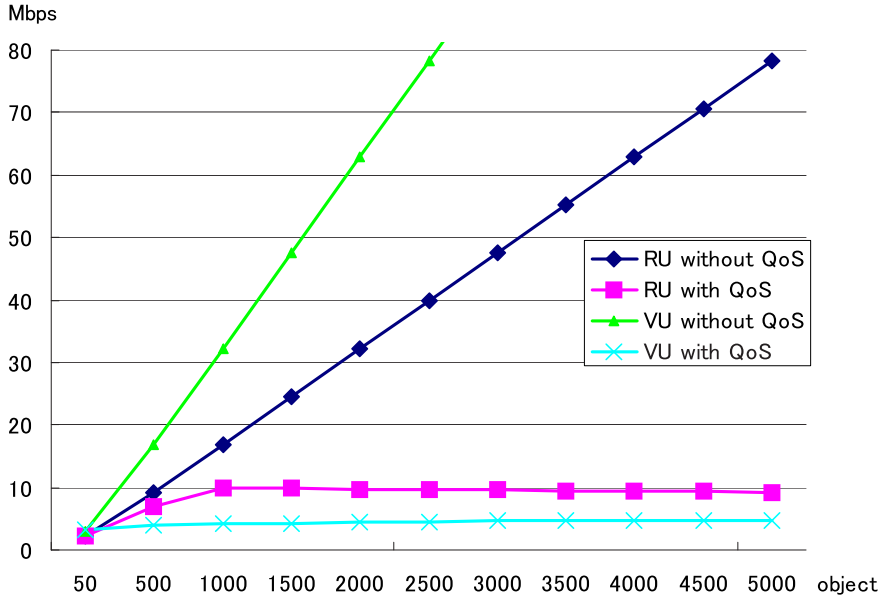

図 20 ユーザ端末にかかるトラフィック

これらの実験環境は，実験 6.2,6.3 では，多数のオブジェクトを配置したときの使用帯域，更新レートを計測するため，蚤の市アプリケーション全体を模した環境に，実験 6.4 では，ユーザが動き回るときの QoS 適応制御が反映されるまでの遅延，パケットが配送されるまでのエンド・エンド遅延を計測するため，ゲームなどのアプリケーションにおいて，参加したユーザに注目した場合の実験環境になっている．

4.6.2 ユーザ端末のトラフィック

提案手法のスケーラビリティを評価するために，QoS 適応制御機構がない場合とある場合の両方について，オブジェクトの数を変化させて，ユーザ端末とそれに対応する帯域制御ノードの間で必要なネットワークの使用帯域幅を計測した．

シミュレーションは表 4 の設定で行った．

また，そのほかの設定は以下の通りである．仮想空間と同じ大きさの実空間が存在する．AP は 1 つとし，AP の伝播範囲は実空間 R 全体をカバーする．無線帯

域 BW_{AP} は，すべての実ユーザがこの帯域を共有する．エリアノード，オブジェクトノード，帯域制御ノードは有線接続の固定サーバ上に割り当てられる．全てのオブジェクトは，ハイブリッド空間に無作為に配置される．ユーザの方向もランダムに決められる．実験 6.2,6.3 では，各ユーザ端末は（X， Y， Z， angleX, angleY，angleZ， オブジェクト名，重要度（各 4[byte]）） = 32[byte] からなる AR イベントを毎秒 60 個 送信する．

実験結果を図 20 に示す．結果は 100 試行の平均である．

QoS 適応制御を行わない場合，オブジェクトの数がそれぞれ，250，500 以上のとき，ユーザ端末のトラフィック量は，各実ユーザ端末（図 20 中 “RU without QoS”）と各仮想ユーザ端末（図 20 中 “VU without QoS”）に必要な帯域幅は限界（すなわち $10Mbps$ と $5Mbps$）を上回った．一方，QoS 適応制御を行う場合，たとえオブジェクト数が 5000 を越える場合でも，必要な帯域幅は限界を越えないように管理することができた（図 20 中 “RU with QoS” および “VU with QoS”）．以上のことから，ユーザ数とオブジェクト数が大きい場合には，QoS 適応機構が有効に働くと考えられる．これにより，蚤の市などの多数のユーザやオブジェクトが存在するアプリケーションでも利用帯域を抑えることができ，本フレームワークは有効であることがわかる．

4.6.3 QoS 適応制御の効果

提案手法では，より重要なオブジェクトの AR イベントは，利用できる帯域幅の範囲内で，他のオブジェクトより大きい通信量で送られる．QoS 適応制御の影響を調べるため，4.6.2 節で述べた領域 a–e にあるオブジェクトの AR イベントの更新レートを計測した．この実験では，オブジェクトが増加したとき，ユーザにとって重要なオブジェクトが，QoS 適応制御により，どの程度まで更新レートを維持できるかを計測した．シミュレーションの設定は，実験 4.6.2 と同様である．結果を，図 21，22 に示す．

図 21，22 において，a, b, c, d, e のラベルは，それぞれ対応する領域の平均更新レートを示す．また，*uniform* のラベルは，帯域幅が一様に分配される場合の平均更新レートを示す．

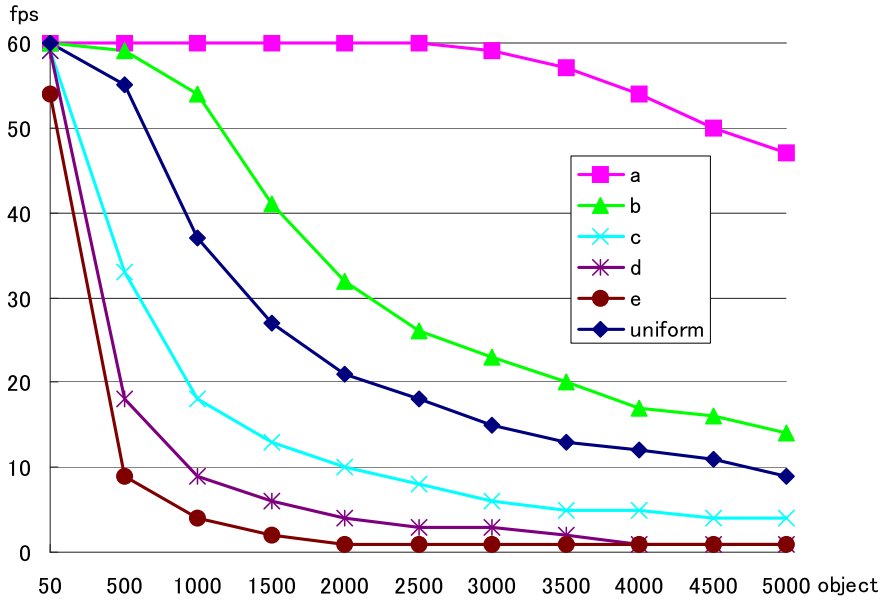

図 21 仮想ユーザが観測するオブジェクトの品質

図 21 より，各仮想ユーザが，領域 a, b の重要なオブジェクトを *uniform* より高い更新レートで見ることができることがわかる．特に，オブジェクト数が 4500 以下では，領域 a のオブジェクトの更新レートは，50fps を保っている．領域 c, d, e の重要でないオブジェクの更新レートは，*uniform* より減らされている．

図 22 より，各実ユーザが領域 a の重要なオブジェクトを *uniform* より高い更新レートで見ることができることがわかる．しかし，その効果は仮想ユーザの場合よりも小さい．他の領域のオブジェクトの更新レートは，*uniform* より減らされている．これは，4.5.1 節で説明したように，実ユーザ全員における重要度の最大値を各実ユーザの重要度とするため，多くのオブジェクトがユーザにとって重要なオブジェクトとみなされてしまうからである．しかし，オブジェクト数が 2000 未満のとき，領域 a のオブジェクトの更新レートは大いに改善されている．この実験結果により，一般的な扇形の視野と各範囲に固定された重要度による設定でも QoS 適応制御によって効果的にデータの更新レートを制御でき，本フレームワークは有効であることがわかる．任意のアプリケーションに特化した重要度の設定法を用いることで，アプリケーションに則した QoS 適応制御を実現すること

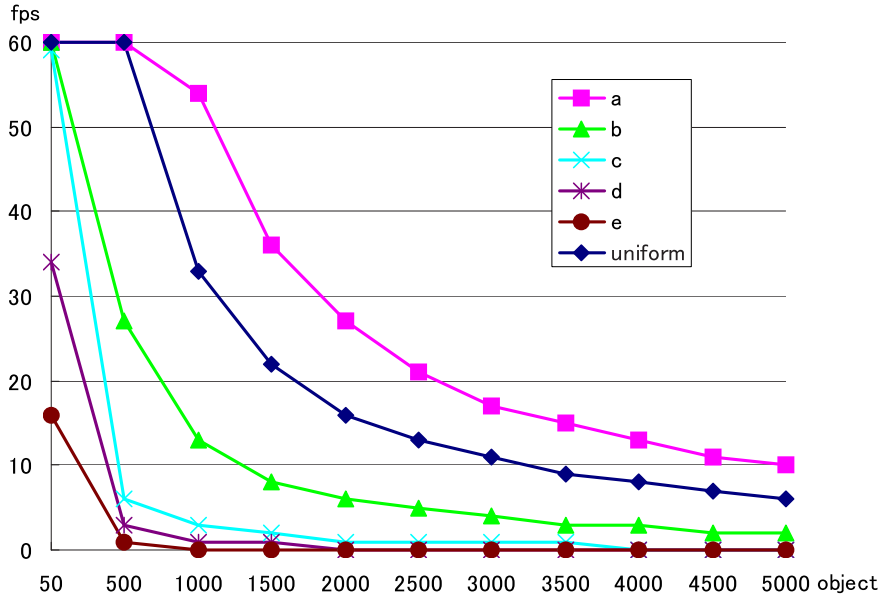

図 22 実ユーザが観測するオブジェクトの品質

ができる．

4.6.4 QoS 適応制御に関するアンケートによるユーザ評価

QoS 適応機構によって，ユーザが求める動画品質が確保されているかを評価するため，4.6.3 節の結果に基づくアンケートによって，提案手法を評価した．領域 a, b, c にユーザがいるという仮定のもと，FAIRVIEW によって生成される仮想的な視界を実験結果に基づいて再現するような動画を作る．まず，あるユーザがジェスチャで，空中に絵を描く動画を用意する．絵は丸，螺旋，上三角，下三角，四角，菱形，ハート，星，上矢印の 9 つである．これらの動画を，領域 a, b, c にユーザが居るように見えるように配置，合成する．このとき，動画のサイズとフレームレートをそれぞれ，領域 a の動画を 368x488/24fps，領域 b の動画を 184x244/7fps，領域 c の動画を 92x122/2fps とした．また，*uniform* では，それぞれ同じサイズで 4fps の動画による合成動画を作成した．これらのフレームレートは，図 21 で示すように，オブジェクト数が 5000 のときのケースに基づいて決定されている．

表 5 アンケートによる評価

Type		fps	ratio of right answer	comprehensi-bility	time to answer
With QoS	a	24	100%	4.5	5.2 sec
	b	7	100%	3.5	7.2 sec
	c	2	83%	2.0	10.5 sec
uniform	a	4	100%	3.3	10.7 sec
	b	4	83%	3.0	6.8 sec
	c	4	66%	2.3	11.1 sec

この動画を被験者に見てもらい，動画の中の人が描いているものを識別するまでの時間，5 レベル（より大きく，より見易い）の主観的なわかり易さ，正解の割合を測定した．その，5 人の被験者の評価の平均を表 5 に示す．表 5 より，アンケートの結果，被験者が提案する QoS 適応制御手法で，より正確に，より速く重要なオブジェクトを認識できることがわかった．

4.6.5 QoS 適応制御における処理遅延

提案する QoS 適応制御を行った場合にかかる遅延を評価するために，LAN 環境において，ユーザの視野が変化した場合に QoS 適応制御によって更新レートが変化するまでの遅延時間を計測した．

実験は表 4 の設定で行った．また，そのほかの設定は以下のとおりである．ユーザは仮想ユーザ 1 人とし，実空間および AP は存在しないものとする．ユーザはまず，空間の中心に配置され，ランダムに決められた目的地に向って，$1m/sec$ で前進する．目的地に到着すると，次の目的地が再びランダムに決められる．仮想空間において，n 個の仮想オブジェクトがユーザを中心として 3 メートル間隔に配置される．このとき，ハイブリッド空間は最も遠くに配置されたオブジェクトが入るだけの広さとする．実験 4.6.5 では，各ユーザ端末は位置情報（8[byte]× 3），回転角度（8[byte] × 3），重要度（4[byte]），ユーザ ID（4[byte]），オブジェクト名（4[byte]），アクション（4[byte]） = 64[byte] からなる AR イベント

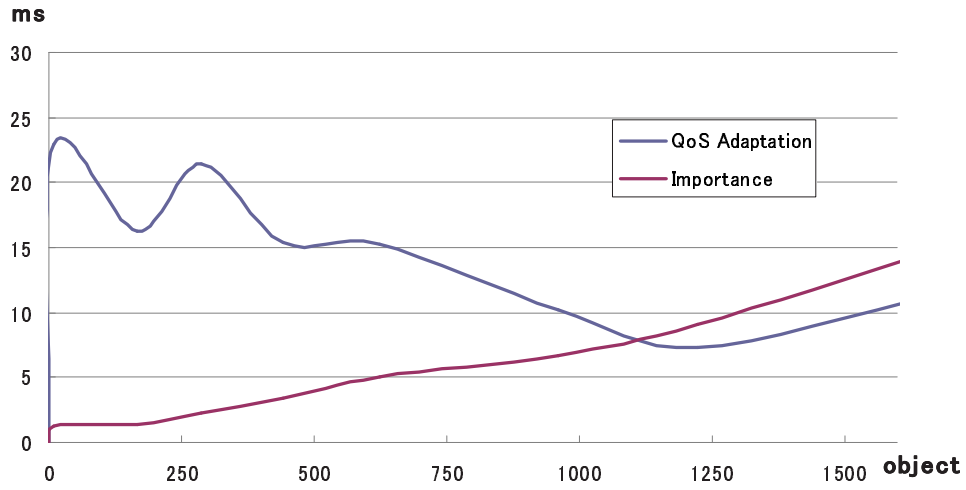

図 23 QoS 適応の遅延時間

を毎秒 30 個 送信する．実験に用いたプロトタイプでは，AR イベントによる各オブジェクトのデータの更新，パケットの送信，パケットの受信，重要度の変更および QoS 適応制御は，1 秒間に 30 回の頻度で並列処理される．このとき，それぞれのプロセスは個別のタイムスロットを使い非同期で処理される．重要度の変更はエリアノードが，QoS 適応制御は帯域制御ノードが行う．通信プロトコルには TCP を使用した．実験には，エリアノード，帯域制御ノード，ユーザノードにそれぞれ，CPU:Athlon64 X2 4200+，メモリ:2GB，OS:Debian Linux（kernel 2.6.8），CPU:Athlon64 X2 4200+，メモリ:2GB，OS:Debian Linux（kernel 2.6.8），CPU:Opteron 242（1.6 GHz）×2，メモリ:8GB，OS:Debian Linux （kernel 2.6.8）のマシンを使用した．プログラム言語には，JAVA SE 1.6.0_05 を用いた．

重要度の変更と QoS 適応制御に関する処理遅延は，計算処理を始める直前と終了した直後のタイムスタンプの差分によって計測した．また，エンド・エンド遅延は，ユーザノードが送信する直前とデータを更新した直後のタイムスタンプの差分によって計測した．これらに，スレッドの生成遅延は含まない．実験結果を図 23，24 に示す．結果は 5 試行の平均である．

図 23 は，アバタの位置に従い変化する重要度の再計算（図 23 中”importance”），

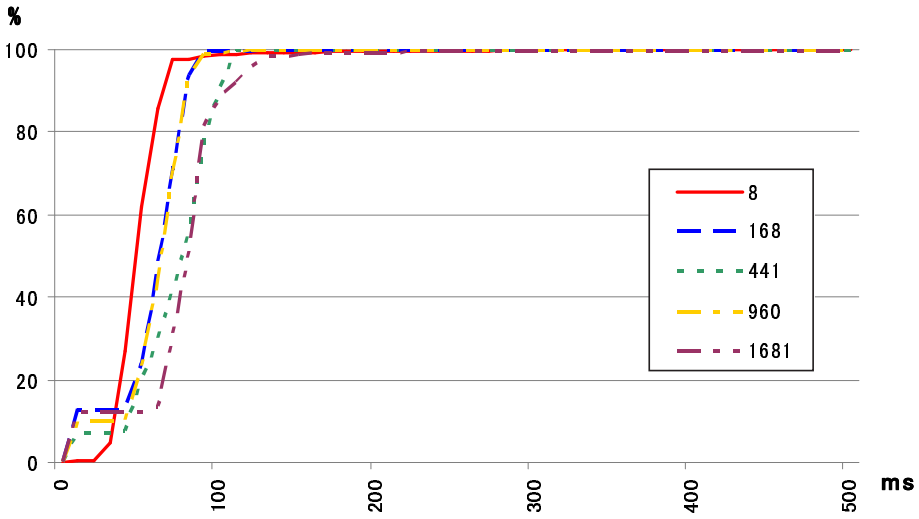

図 24 AR イベントが伝わるまでの処理遅延の累積分布

および，それを基にした QoS 適応制御にかかる再計算（図 23 中”QoS Adaptation”）の処理遅延のグラフである．これより，重要度の再計算はオブジェクト数に従い計算量が増加するため，処理遅延が大きくなることがわかる．この処理遅延の主な原因は，アバタ対オブジェクトの距離・角度計算である．これにより，オブジェクト数が多い環境では，短い周期（例えば，フーレム毎）で計算するのは難しいことがわかる．このとき，CPU を効率的に使うために，再計算の頻度を下げるか，エリアノードの数を増やす必要がある．一方，QoS 適応制御にかかる時間は処理のタイミングによって振動するものの，25ms 以下に収まっており，オブジェクト数に依存しないことがわかった．

さらに，重要度の再計算と QoS 適応制御にかかる再計算の合計により，ユーザが振り向くなどの急激なアクションへの耐性をみることができる．つまり，ユーザが振り向きなどの動作によって，各オブジェクトの重要度が大幅に変化した場合，その影響がどの程度の遅延で反映されるかがわかる．

図 24 は，図 23 の結果を含め，ユーザノードが AR イベントを送信し，その AR イベントがエリアノードに受理され，帯域制御ノードを経由して戻ってくるまでのエンド・エンド遅延を測ったものである．およそ 200ms 以内に収まっている．

遅延の主な原因は，重要度の再計算処理，QoS 適応制御にかかる再計算処理，非同期の計算プロセスによる待機時間が考えられるが，図 23 より，非同期による待機時間が最も大きな原因であることがわかる．図 23，24 からオブジェクトが 1500 個を越えた状況でも 30ms 程度の計算遅延で処理され，その影響は遅くとも 200ms 以内にはユーザに反映されることがわかる．人間が振り向き，対象にフォーカスをあわせる，という動作にかかる時間を考慮すれば，この遅延時間は，実用上，十分に許容範囲内である．これにより，本フレームワークは，即応性の求められるゲームなどにも，十分に対応できることがわかる．

4.7 まとめ

本章では，多人数参加型の実・仮想空間のインタラクションを実現するためのフレームワーク FAIRVIEW を提案した．提案手法では，共有空間を分割した上で，Publish/Subscribe 方式を用いることで，ユーザ自身を含む動くオブジェクトに関する情報の共有を行う．また，普及ネットワーク環境での利用を目的に，インタラクションの際のユーザ満足度を高く保ちながら通信量を削減するための QoS 適応機構を提案し，実験による評価を行った．

5. 結論

本論文では，筆者が多人数参加型の実・仮想空間のインタラクションを実現を目指し，おこなった 2 つの研究について記述した．研究内容は，(1) 多人数参加型のネットワークゲームを対象とした分散型イベント配送方式の研究および，(2) 多人数参加型の実・仮想空間のインタラクションを実現するためのフレームワーク FAIRVIEW および普及ネットワーク環境での利用を目的に，インタラクションの際のユーザ満足度を高く保ちながら通信量を削減するための QoS 適応機構の研究である．以下，得られた成果および今後の課題をまとめる．

多人数参加型ネットワークゲームの分散型イベント配送方式では，プロトタイプシステムと ns-2 でのシミュレーションによる実験により，提案方式が，各ノードの計算負荷および通信量を負荷分散木を用いない場合より小さく抑えられること，一般的なネットワークにおけるメッセージ配送遅延をある部分領域のプレイヤ密度が高くなっても，十分に使用可能な範囲に抑えることができることを確認した．

多人数参加型実・仮想空間混合インタラクション用フレームワークでは，試算，アンケート，プロトタイプシステムによる実験により，ユーザの視界にあるオブジェクトの動きを十分に実用可能な範囲の更新レートに保ちつつ，通信量を利用可能な帯域に収めることができることがわかった．また，その際にかかる計算処理にかかる遅延時間も実用上，十分に許容範囲であることを確認した．

本論文で提案したフレームワークを用いることで，一般的なネットワーク環境および安価な装置などの普及環境において，様々な社会活動に遠隔ユーザがネットワークを介して仮想的に参加するような実・仮想空間混合対話型アプリケーションを容易に構築することができる．本論文において，基本的なシステムの構築技術は提案できた．今後，無線環境下において，同じエリアでアクセスポイントが 2 つ以上ある場合について実験をおこない，実ユーザのオブジェクトの観測性能の向上効果を調査したいと考えている．そして，その調査結果を踏まえ，インターネットに比べ，遅延が大きく変動する傾向が無線通信環境下に特化した，遅延を小さく抑えるような対策を提案したい．また，提案した QoS 適応制御手法では，オブジェクトを一律に重要度であつかうことによって，ゲームにおいて，標的と

障害物が同等にあつかわれてしまうため，今後，オブジェクトの種類による重要度の調整方法についても検討・新規提案したい．最終的には，提案手法をミドルウェアとして実装し，テストアプリケーションを構築して評価をおこなった後，一般公開したいと考えている．

実・仮想空間における多人数インタラクションの実現にむけての今後の課題として，ユーザが得る情報をユーザ満足度が最大化するように効率的かつ自動的に取捨選択することが挙げられる．現在，VE 技術に関する研究は，如何に効率的なネットワークアークテクチャを構築するかというトピックから，如何にユーザの AOI におけるインタラクションの高品質化をおこなうかというトピックに移りつつある．ユーザが関心のある事象を自動で推測し，それに沿う処理をおこなうという目標は，AR 技術や MR 技術における基本的な概念の 1 つである．それぞれの研究は，互いの弱点を補えるものが多く，VE 技術と AR/MR 技術をより融合させる研究が今後発展していくと予想される．

VE 技術，AR 技術，MR 技術の普及のためには，キラーコンテンツの提案が重要な課題となる．VE 技術，AR 技術，MR 技術は，エンターテイメントやデモンストレーションと相性が良く，ゲームや展示などに今後ますます利用されていくことが予想される．現在でも，様々な映画作品において，VE 技術，AR 技術，MR 技術の基本的な概念を提示する場面が増えてきている．今後は,「見せる」だけでなく「触らせる」ことが重要となってくるであろう．これを踏まえて，無線環境下をはじめとした様々な状況に対応し，用意した会場などの特定の環境下ではなく，一般的な生活環境に対応することも，重要な課題となると思われる．

謝辞

指導教官である伊藤実 教授には，素晴らしい研究環境を与えて頂くとともに適切な御指導を頂きました．ここに深く感謝致します.

加藤博一 教授には，本研究の副指導教官として，本論文の作成および研究にわたり有益な情報や御助言を賜わり，御指導頂きました．ここに深く感謝致します.

安本慶一 准教授には，本研究の方向性をはじめ，様々な面で御指導して頂きました．心より感謝致します.

柴田直樹 滋賀大学准教授， 村田佳洋 広島市立大学准教授には，本研究を進めるにあたり，多くの御助言を頂きました．ここに深く感謝致します.

また，研究に関してのみならず，研究室の生活において手助けして頂いた木谷友哉 助教，孫為華 助教，ATR 適応コミュニケーション研究所 玉井森彦 博士に深く感謝します.

尾川恵理 事務補佐員 には，研究費管理などの事務手続き等をお世話して頂きました．ここに深く感謝致します.

研究室の皆様には，日頃から公私に渡りお世話になりました．ここに深く感謝致します.

最後に，研究生活を支えてくれた家族，友人に感謝します.

参考文献

[1] B. D. Vleeschauwer, B. V. D. Bossche, T. Verdickt, F. D. Turck, B. Dhoedt, P. Demeester: *"Dynamic Microcell Assignment for Massively Multiplayer Online Gaming," Proc. of 4th ACM Workshop on Network and System Support for Games (NETGAMES '05)* , pp.1–7, (2005).

[2] K. Yasumoto and K. Nahrstedt: *"Realistic Voice Chat Framework for Cooperative Virtual Spaces," Proc. of IEEE 2005 International Conference on Multimedia and Expo (ICME 2005)* , CD-ROM, (2005).

[3] M. Eraslan, N. D. Georganas, J. R. Gallardo and D. Makrakis: *"A Scalable Network Architecture for Distributed Virtual Environments with Dynamic QoS over IPv6," Proc. of IEEE Symposium on Computers and Communications (ISCC'2003)* , vol.1, pp.10–15, (2003).

[4] 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ネットワーク環境における利用実験，" 信学技法，IN2000-121，pp.21–28，(2000).

[5] A. Bharambe, S. Rao and S. Seshan: *"Mercury: A Scalable Publish-Subscribe System for Internet Games," Proc. of 1st Workshop on Network and System Support for Games (NetGames2002)* , pp.3–9, (2002).

[6] A. P. Yu and S. T. Vuong: *"MOPAR: A Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games," Proc. of ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV '05)* , pp.99–104, (2005).

[7] B. Knutsson, H. Lu, W. Xu, and B. Hopkins: *"Peer-to-Peer Support for Massively Multiplayer Games," Proc. of INFOCOM 2004* , pp.1–12, (2004).

[8] R. Chertov and S. Fahmy: *"Optimistic Load Balancing in a Distributed Virtual Environment," Proc. of ACM Network and Operating Systems Support for Digital*

Audio and Video (NOSSDAV '06) , pp.74–79, (2006).

[9] S. Y. Hu, G. M. Liao: "*Scalable peer-to-peer networked virtual environment*," *Proc. of ACM 3rd Workshop on Network and System Support for Games (NETGAMES '04)* , pp.129–133, (2004).

[10] T. Iimura, H. Hazeyama and Y. Kadobayashi: "*Zoned Federation of Game Servers: a Peer-to-peer Approach to Scalable Multiplayer Online Games*," *Proc. of the 3rd Workshop on Network and System Support for Games (NETGAMES 2004)* , pp.1–5, (2004).

[11] R. Ichikari, K. Kawano, A. Kimura, F. Shibata and H. Tamura: "*Mixed Reality Pre-visualization and Camera-Work Authoring in Filmmaking*," *Proc. of 5th International Symposium on Mixed and Augmented Reality* , pp.239–240, (2006).

[12] K. Fudono, T Sato. and N. Yokoya: "*Interactive 3-D modeling system using a hand-held video camera*," *Proc. of 14th Scandinavian Conference on Image Analysis (SCIA2005)* , pp.1248–1258, (2005).

[13] M. Kourogi and T. Kurata: "*A method of personal positioning based on sensor data fusion of wearable camera and self-contained sensors*," *Proc. of IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2003)* , pp.287–292, (2003).

[14] ARtoolkit,
http://www.hitl.washington.edu/artoolkit/.

[15] M. Fujimoto and Y. Ishibashi: "*Packetization Interval of Haptic Media in Networked Virtual Environments*," *Proc. of 4th ACM Workshop on Network and System Support for Games (NETGAMES '05)* , pp.1–6, (2005).

[16] Z. Yang, B. Yu, K. Nahrstedt and R. Bajscy: "*A Multi-stream Adaptation Framework for Bandwidth Management in 3D Tele-immersion*," *Proc. of ACM Network*

and Operating System Support for Digital Audio and Video (NOSSDAV' 06) , pp.1–6, (2006).

[17] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nahrstedt, S. Jung and R. Bajscy: *"TEEVE: The Next Generation Architecture for Tele-Immersive Environments," Proc. of 7th IEEE International Symposium on Multimedia (ISM'05)* , pp.1–8, (2005).

[18] 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，(2003).

[19] 公原勝彦： *"P2P* 技術を用いたネットワークゲーム向け負荷分散機構の提案，" 奈良先端科学技術大学院大学情報科学研究科修士論文，修第 2461 号, (2004).

[20] Netowork Simulator 2,
http://www.isi.edu/nsnam/ns/index.html.

[21] PlanetLab,
http://www.planet-lab.org/.

[22] S. Yamamoto, Y. Murata, N. Shibata, K. Yasumoto and M. Ito: *"QoS Adaptation for Realizing Interaction between Virtual and real Worlds in Pervasive Network Environment," Proc. of ACM Network and Operating Systems Support for Digital Audio and Video (NOSSDAV' 07)* , pp.119–124, (2007).

[23] S. Benford, and L. Fahl'en: *"A Spatial Model of Interaction in Large Virtual Environments," Proc. of Third European Conference on CSCW (ECSCW'93)* , pp.107–122, (1993).

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan: *"Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," Proc. of ACM SIGCOMM'01* , pp.149–160, (2001).

[25] A. S. Tanenbarm and M. V. Steen: *"Distributed Systems - Principles and Paradigms," Prentice Hall* , (2002).

[26] D. L. Mills: *"RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation,"* 1992, http://www.faqs.org/rfcs/rfc1305.html.

[27] J. Kim, J. Choi, D. Chang, T. Kwon and Y. Choi: *"Traffic Characteristics of a Massively Multiplayer Online Role Playing Game and Its Implications," Proc. of the 4th Workshop on Network and System Support for Games (NETGAMES 2005)* , pp.1–8, (2005).

[28] Tiers Topology Generator, http://www.isi.edu/nsnam/ns/ns-topogen.html.

[29] T. Henderson: *"Latency and Behaviour on a Multiplayer Game Server," Proc. of 3rd Int'l. Workshop on Networked Group Communication (NGC2001)* , LNCS2233, pp.1–13, (2001).

[30] A. Rowstron and P. Druschel: *"Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* , pp.329–350, (2001).

[31] T. Kitasuka, T. Nakanishi and A. Fukuda: "Wireless LAN based Indoor Positioning System WiPS and Its Simulation," *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'03)* , pp.272–275, (2003).

[32] J. Scott and B. Dragovic: *"Audio Location: Accurate Low-Cost Location Sensing," Proc. of 3rd International Conference on Pervasive Computing(Pervasive 2005)* , LNCS 3468, pp.1–18, (2005).

[33] Intel Research: Place Lab,
http://www.placelab.org/.

[34] S. Yamamoto, Y. Murata, K. Yasumoto and M. Ito: *"A Distributed Event Delivery Method with Load Balancing for MMORPG," Proc. of 4th ACM Workshop on Network and System Support for Games (NETGAMES '05)* , pp.1–8, (2005).

業績リスト

論文誌

1. 山本眞也，村田佳洋，安本慶一，伊藤実：マルチユーザネットワークゲームにおける負荷分散および遅延時間を考慮したイベント配送方式の提案，情報処理学会論文誌，Vol.47，No.2，pp.475-483，(February 2006).
（3 章）

2. 山本眞也，村田佳洋，柴田直樹，安本慶一，伊藤実：実・仮想空間を跨るインタラクションを実現するための QoS 適応機構とその評価，情報処理学会論文誌，Vol.50，No.2，pp.765-776 (February 2009).
（4 章）

国際会議 (査読有り)

1. Shinya Yamamoto ○, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito: A Distributed Event Delivery Method with Load Balancing for MMORPG, Proceedings of ACM 4th Workshop on Network and System Support for Games (NetGames 2005), on-line proceedings, New York Hawthorne, (October 2005). （口頭発表）
（3 章）

2. Shinya Yamamoto ○, Yoshihiro Murata, Naoki Shibata, Keiichi Yasumoto, and Minoru Ito: QoS Adaptation for Realizing Interaction between Virtual and Real Worlds in Pervasive Network Environment, Proceedings of ACM 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2007), pp.119-124, America (June 2007). （口頭発表）
（4 章）

3. Shinya Yamamoto ○, Yoshihiro Murata, Naoki Shibata, Keiichi Yasumoto, and Minoru Ito: QoS Adaptation in Streaming 3D Graphics for FAIRVIEW, Proceedings of ACM 18th International Workshop on Network and Operating Sys-

tems Support for Digital Audio and Video (NOSSDAV2008), pp.125-126, Germany (May 2008).（デモ発表）
（4 章）

国内学会 (査読有り)

1. 山本眞也○，村田佳洋，柴田直樹，安本慶一，伊藤実：現実空間ユーザと仮想空間ユーザのリアルタイム情報共有のためのフレームワークの提案，マルチメディア，分散，協調とモバイル (DICOMO2006) シンポジウム， 香川県 琴平温泉 琴平グランドホテル，pp.609-612，(July 2006). （口頭発表）
（4 章）

2. 山本眞也○，村田佳洋，柴田直樹，安本慶一，伊藤実：実空間と仮想空間におけるインタラクションを一般的なネットワーク環境で実現するためのフレームワークの提案， 情報処理学会 マルチメディア，分散, 協調とモバイル (DICOMO2007) シンポジウム論文集，pp.1690-1699, (July 2007).（口頭発表）
（4 章）

3. 山本眞也○，村田佳洋，柴田直樹，安本慶一，伊藤実：3D 仮想空間におけるユーザ間インタラクションを実現するためのユーザ視野に基づく QoS 適応機構，第 16 回マルチメディア通信と分散処理ワークショップ (DPSWS2008), （December 2008）.（デモ発表）
（4 章）

国内学会 (査読無し)

1. 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，私立 同志社大学 京田辺キャンパス，CD-ROM，(September 2004).

（口頭発表）
（3 章）

2. 山本眞也○，村田佳洋，安本慶一，伊藤実：マルチユーザネットワークゲームにおける負荷分散および遅延時間を考慮したイベント配送機構の提案，情報処理学会研究報告，2005-DPS-122，国立大阪大学 吹田キャンパス，pp. 99-104，(March 2005).（口頭発表）
（3 章）