社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－DPS－122 (18)
2005－CSEC－ 28 (18)
2005／3／22

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

山本 眞也 村田 佳洋 安本 慶一 伊藤 実
奈良先端科学技術大学院大学情報科学研究科

あらまし ピアツーピア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ゲームを実現することを目的として，我々の研究グループでは，ゲームで発生するイベントの登録・通知を，ゲーム領域を分割してできた領域ごとにゲーム参加者の計算機に担当させ，分散処理させる機構を提案してきた．本手法では，各領域のユーザ数が増加してイベント通知を担当する計算機の負荷が高くなると，複数の計算機からなる負荷分散木を動的に構築し，イベント通知を負荷分散木を経由して行うことで１台あたりの負荷を軽減する．しかし，イベントの配送パスが長くなることによる配送遅延の増加が問題であった．また，従来手法では，ユーザの視野領域が一つのゲーム領域に含まれることを想定しており，複数のゲーム領域に跨る際の具体的なイベント配送について考慮されていなかった．本稿では，(1) 負荷分散木上のノードの動的入れ替えによる，エンド・エンドのイベント配送遅延の短縮方法，(2) 各プレイヤの視界が複数のゲーム領域に跨る場合のイベント配送方法，を提案する．また，ns-2によりシミュレーション実験を行い，提案手法によりイベント配送遅延が 40% 程度軽減できることを確認した．

An Event Delivery Mechanism for Multi-Player Games on P2P Networks with Load Balancing and Short Latency

Shinya Yamamoto Yoshihiro Murata Keiichi Yasumoto Minoru Ito
Graduate School of Information Science, Nara Institute of Science and Technology

Abstract In order to achieve multi-party networked games without specific servers in P2P environments, we proposed a P2P based event delivery method where a shared game space is divided into multiple sub-areas (game areas) and some nodes are selected from all players to deliver game events occurring in their responsible areas to player nodes. This method also includes a load balancing mechanism which allows each responsible node for the congested area to dynamically construct a tree of multiple nodes and deliver events along the tree to reduce event forwarding overhead per node. However, this mechanism introduces larger delivery latency as the event delivery path becomes longer. Also, when each user has visible area over several game areas, he/she must subscribe to multiple responsible nodes to receive events occurring in those areas. In our existing method, this problem was not considered in detail. In this paper, we propose techniques (1) to reduce end-to-end event delivery latency by dynamically replacing nodes in the tree, and (2) to efficiently deliver events to users who have visible areas over multiple game areas. Through simulations with ns-2, we have confirmed that the proposed method can reduce end-to-end event delivery latency up to 40%.

1 まえがき

近年，ピアツーピア (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ゲームを実現する方法が注目されている．現在のネットワークゲームは，主としてクライアント・サーバによる集中制御手法をもとに構築されていることが多い．集中制

御手法はサーバに負荷が集中するという欠点を持つ．この欠点を補うために，ゲーム空間を均等に分割し，分割されてできた複数の領域におけるイベント通知を別々の計算機に担当させる手法が提案されている．このような手法として，固定数のサーバに対応させる手法 [1] や，P2P 環境を用いていくつかのプレイヤの計算機をサーバとみなし処理させる手法 [2] が提案されている．

ゲーム領域を分割する手法においては，分割されたゲーム領域間をプレイヤのキャラクタが移動したとき，移動先のゲーム領域を担当するサーバを見つける手段が必要である．そこで，隣接するゲーム領域を担当するノードへの経路制御表を持たせる方法 [4] や，分散ハッシュテーブルを持たせる方法 [3] により，必要なサーバのアドレスを短時間で求める方法が提案されている．

しかし，ゲーム空間を均等分割し各ゲーム領域でのイベント処理を 1 台の計算機で処理する方法では，特定の領域のプレイヤ数およびそこで発生するイベントの数が多くなると，その領域を担当するノードの負荷が高くなってしまうという問題が生じる．我々は，ゲーム空間を均等分割してできる各ゲーム領域の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）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，各ノードのイベント配送処理のための負荷が予め定めた水準を超えないようにする方法を提案している [4]．しかし本手法は，負荷分散木が大きくなるにつれて，サーバからプレイヤまでの通信遅延が増大するという問題点があった．そのため，通信負荷の分散だけでなく通信遅延が大きくなりすぎないように木のバランスを取るなどの対策が必要である．また，ユーザの視野範囲が複数のゲーム領域に跨る際には，それら領域を担当する複数のサーバにイベント配送要求を出し，イベントを受信する必要がある．しかし，本手法では，ユーザの視野範囲が一つの部分領域に含まれることを想定していたため，複数の領域からの具体的なイベント配送方法について考慮されていなかった．

本稿では，(1) 負荷分散木が用いられる際のエンド・エンドのイベント配送遅延の短縮方法，(2) 各プレイヤの視界が複数のゲーム領域に跨る場合のイベント配送方法，を提案する．(1) は，負荷分散木上の各ノードの故障に備えて用意されるバックアップノードを用いて，そのノードとバックアップノードを入れ替えた場合に配送遅延が改善されるかどうかを判定し，改善される場合には置き換える，といった方法を繰り返し適用することで実現する．また，(2) では，プレイヤの移動に伴い，視野範囲と重なる部分領域を計算し，該当する担当ノードにイベント配送予約の実行，取り消しを行うことで実現する．その際，部分領域間の境界付近を移動することによって，同じ領域に対し新規イベント配送予約の実行と取り消しが振動して行われないよう工夫する．

Tiers により生成した 1011 ノードを持つ階層ネットワークを対象に ns-2 によるシミュレーションを行い，負荷分散木の更新アルゴリズムによる効果を確認した．その結果，更新アルゴリズムを用いない場合よりも，4 割程度通信遅延を軽減できることを確認した．

2 提案するイベント配送機構

本章では，我々が文献 [4] で提案したネットワークゲーム向けイベント配送機構の概要について述べ，その後，本稿で提案する 2 つの改善手法について説明する．

2.1 概要

提案手法では，ネットワークで接続された複数のプレイヤが同一の仮想空間と時間を共有する，鳥瞰型のロールプレイングゲームを想定する．仮想空間は，背景と複数のオブジェクトからなるものと定義する．ここで，オブジェクトは，アバタ（プレイヤ自身を表すオブジェクト）や他の移動体および静止物であり，それぞれ属性を持つ．オブジェクトの属性を変化させる出来事をイベントと呼ぶ．イベントの例として，ユーザの移動や，他のオブジェクトへの働きかけ，オブジェクトの色・形状の変化等がある．このようなネットワークゲームにおいて，複数プレイヤによるゲーム進行の一貫性を保証するには，以下の条件を満たすイベント配送機構が必要である．

(1) 全てのプレイヤは自身を表すアバタの仮想空間上の位置に応じて仮想空間上のある領域（以降，視野範囲と呼ぶ）における最新の状態（その領域内に存在する全てのオブジェクトの最新の位置と属性）を観測できる．

(2) 仮想空間上のある領域内で発生したイベントは，そのイベントを観測可能な（すなわちイベントの発生位置を視野範囲に含む）全プレイヤにリアルタイムで伝達される．

(3) 連続して発生した一連のイベントの発生順序は，そのイベントを観測できる全プレイヤ間で一意に保たれる．

提案手法は，P2P 環境において上記を実現するため，以下の方針を採用する．まず，全プレイヤのリストを保持するロビーサーバ S の存在を仮定する．仮想空間全体を固定数の領域（部分領域と呼ぶ）に分割する (図 1)．

イベントの処理が部分領域で分散して処理されるよう，各部分領域に対し 1 台の計算機が割り当てられる．この各領域のイベント処理を担当する計算機を担当ノードと呼ぶ．担当ノードはゲームに参加している

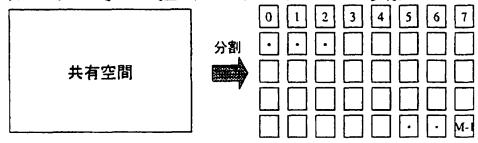

図 1: 仮想空間の分割

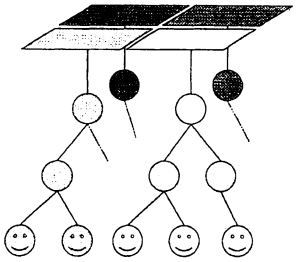

図 2: 部分領域と負荷分散木

プレイヤの計算機（以下，プレイヤノード）の中から割り当て，最初の割り当てはロビーサーバ S が行う．プレイヤの数によっては，一つのプレイヤノードを複数の領域の担当ノードとして割り当てることも可能である．

次に，担当ノード群をリング型トポロジーにより接続する．各担当ノードに Chord[5] に基づいた分散ハッシュテーブルを持たせることで，各プレイヤノードは任意の部分領域の担当ノードを，担当ノードの総数 n に対して $O(\log n)$ で発見できる．また担当ノードに隣接する部分領域の担当ノードのアドレス表を持たせることで，プレイヤが隣接領域に移動したときに必要となる担当ノードを即座に見つけることができる．

各プレイヤノードはパブリッシュ・サブスクライブモデル [7] を用いて，観測可能なすべての部分領域の担当ノードに対してイベント配送予約（サブスクライブ）する．イベント配送予約はイベントの配送要求にあたり，それら部分領域で発生するイベントのみを受信することにより全体としての通信量の削減をはかっている．一方，イベントを起こしたプレイヤは，担当ノードにイベント発生通知（パブリケーションメッセージ）を送信する．イベント発生通知を受け取った担当ノードは全てのイベント配送予約者（サブスクライバ）にそのイベントを転送する．ある部分領域での負荷（イベント配送予約しているプレイヤ数 × 単位時間あたりに発生するイベント数）が予め指定した閾値 C を越えた場合には，その領域の担当ノードを根とする木構造のネットワーク (以降，負荷分散木と呼ぶ) を動的に構築し，負荷分散木に沿ってイベントを配送することでノードあたりの負荷を軽減する (図 2)．

2.2 部分領域と視野範囲

ゲーム内の仮想空間を固定数の部分領域に分割する手法は，これまで，プレイヤの視界を単一の部分領域内に限定していた [1] [2]．このため，プレイヤの操作するキャラクタが部分領域の境界付近にいた場合，どちらか片方の部分領域しか観測することができなかった．このことはプレイヤに仮想空間の境界を認識させ，一つの部分空間を小さく設計することへの妨げとなると考えられる．

提案手法では，プレイヤが複数の部分領域を同時に観測することを許す．視野範囲は複数の部分領域にまたがることができるとする．プレイヤは視野範囲を一部でも含んでいるような仮想空間に対応する部分領域の担当ノードに対してイベント配送予約をし，そこで発生するイベントメッセージを受ける．

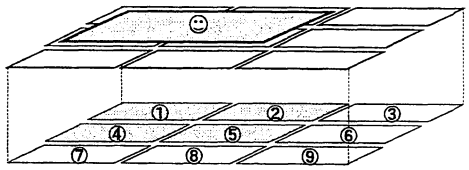

図 3: 部分領域と視野範囲

プレイヤがリアルタイムで影響を受けるイベントは，視野範囲内で発生するイベントに限られる．もし遠隔地が観測可能であるようなゲームならば（自宅に泥棒が入れば感知できる等），その部分領域の担当ノードにもイベント配送予約をする．仮想空間は，先に述べたとおり，図 1 のように部分領域に分けられているとする．基本的にプレイヤの操作するキャラクタの周囲が観測可能であるとする（図 3）．プレイヤはイベント配送予約をしたすべての担当ノードからイベントを受け取るため（図 4），プレイヤの操作するキャラクタの位置が部分領域の境界付近にあっても，境界を感じることなく周囲を観測することができる．キャラクタが移動して視野範囲からある部分領域が外れた場合，その部分領域の担当ノードに対するイベント配送予約は取り消される (取り消しメッセージを送る)．

提案手法ではイベント配送予約をしておかないとイベントを受けとることができないために，キャラクタが移動してからイベント配送予約を行なうと，移動と同時に境界付近で発生したイベントのメッセージを受けとり損ねるかもしれない．そこで，視野範囲よりもひとまわり大きいイベント配送予約範囲という概念を設け，その範囲内の部分領域の担当ノードにあらかじめイベント配送予約をしておく．

また，イベント配送予約範囲の端辺と部分領域の境界が重なっているときに，キャラクタが極めて近接した 2 地点の往復を繰り返したとする．部分領域がイベント配送予約範囲から外れたときにイベント配送予約を取り消すならば，キャラクタから遠い側の部分領域に対するイベント配送予約とその取り消しが頻繁に繰り返され，メッセージを浪費することになる．そこで，イベント配送予約領域よりさらにひとまわり大きいイベント配送取り消し範囲という概念を設け，イベント配送予約された部分領域がそこから外れたときにイベント配送予約を取り消す．このことにより，イベント配送予約範囲とイベント配送取り消し範囲の間の幅より小さいキャラクタ往復によるメッセージの増加を防ぐことができる．これら範囲は大きく取るほど効果を高めることができるが，反面イベントメッセージの増大を招く．そのためその間のトレードオフを考慮して範囲を決定せねばならない．

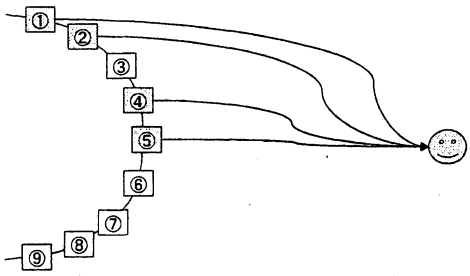

図 4: イベントの受信

2.3 遅延の削減を考慮した負荷分散木の構築

2.3.1 負荷分散木の構築

担当ノードの負荷を，その担当ノードにイベント配送予約しているプレイヤ数 × 単位時間あたりに発生するイベント数として定義する．負荷が少ない間は，担当ノードはイベント配送予約しているすべてのプレイヤに対して直接イベントメッセージを送る．しかしこの負荷が予め指定した閾値 C を越えた場合には，リング上の該当ノードを根とする木構造のネットワークを動的に構築し，その木を通じてイベントメッセージを送ることによって，一つのノードあたりの負荷 (メッセージの配送のための計算パワーと必要通信量) を軽減させる．

担当ノードや負荷分散木における中継ノード (以後，中間ノード) には，離脱に備えて補助のノード (以後，バックアップノード) を用意しておく．このバックアップノードは，同時に新規に追加される中間ノードの候補であり，ロビーサーバから選出されていると仮定する．

担当ノードは，ロビーサーバからバックアップノードの情報を受け取っており，中間ノードを構築する際に，それらのノードから k 個を選び，子ノードとして接続する．担当ノードは，以下の手順で負荷分散木を構築し，子ノードへサブスクライバ (イベント配送予約を行っているプレイヤの集合) の割り当てを行う．

(1) 担当ノードは，現在のサブスクライバの集合を，ほぼ同数になるよう k 分割し，分割された k 個のサブスクライバの集合を，担当ノードに接続された k 個の子ノードに送信する．ここで，k は予め定めた定数である．

(2) 担当ノードがイベント発生通知を受け取ったら，k 個の子ノードに転送する．また，担当ノードがイベント配送予約を受け取ったら，子ノードのうち，最もサブスクライバの集合が小さいノードに転送する．

(3) 負荷分散木のある葉ノードの負荷が高くなったら (サブスクライバの数×単位時間あたりに発生するイベント数が閾値 C を越えたら)，手順 (1) に従い，新たに子ノードを割り当て，サブスクライバの集合を分割し，子ノードに送信する．

(4) 負荷分散木のある中間ノードを根とする部分木のサブスクライバの数×単位時間あたりのイベント数が閾値 C 以下になったら，子ノードの統合を行う．そのため，各中間ノードでは，イベントの発生頻度，子ノードの担当するサブスクライバの集合をモニタさせる．

以上の手順により，負荷分散木上のノードがイベント配送予約しているプレイヤ群に対し，分散してイベントを配送することができ，各ノードが単位時間あたりに処理するべき配送の回数が削減される．

2.3.2 バックアップノード

提案手法では，プレイヤノードにイベント配送を行わせているが，各プレイヤノードは永続的にゲームに参加するわけではなく，常に離脱する可能性を持っている．従って，イベント配送を行っているプレイヤノードが突然離脱した時でも，他のプレイヤのゲーム進行に支障をきたさず，システム全体が継続して動作できるようにするための機構が必要である．提案手法では，イベント配送の一部を担当するノードに対し，バックアップを設けることでこれに対処する．バックアップノードには，処理能力が高く利用可能通信帯域に余裕があるプレイヤノードが優先的に選ばれる．バックアップノードは，ロビーサーバによって一定個数が選出され，その情報がロビーサーバ上のバックアップノードキューに保持される (図 5)．担当ノード，負荷分散木の中間ノードのバックアップには，このキューから順次取り出され，割り当てられる．あるノード b が負荷分散木上のある中間ノード a のバックアップノードに割り当てられた場合，b はマスターノード a と定期的に交信を行い，a が機能しているかどうかを常に監視する．また，この際 a の親ノード p の情報をもらっておく．ノード a が離脱等によって交信不能になった場合には，a の親ノード p に交代を要請することで，a の子ノード (あるいは，a が葉ノードだった場合は a が管理していたサブスクライバ集合) を貰い，それらに新たなコネクションを張ることで中間ノードとなる (図 5)．なお，ある領域の担当ノード自身が離脱する場合も同様にバックアップノードにより回復が可能である (詳細は，文献 [4] 参照)．

2.3.3 通信遅延の軽減

2.3.1 節で述べたように，負荷分散木は，部分領域の負荷に応じて再帰的に拡張される．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．そのため，再帰的な分散によって遅延が増大してしまうことのないように，できるだけ遅延の少ない負荷分散木を構築する必要がある．提案手法では，欲張り法的なアルゴリズムを用いて，中間ノードを置き換えることにより，負荷分散木における配送遅延を序々に改善して行く方法を取る．

負荷分散木を更新するにあたって，プレイヤが仮想空間上を部分領域を超えて移動すると担当ノードも変わるので，プレイヤノードは，現在の負荷分散木から

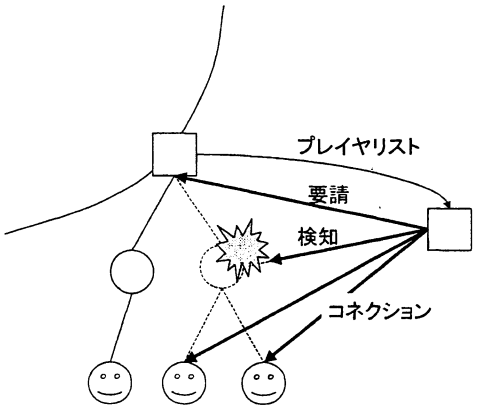

図 5: バックアップノード

離脱する可能性が最も高い．一方，担当ノードはゲーム上から離脱しない限りは不変である．よって，負荷分散木における配送遅延の改善は担当ノードを基準に行うこととする．

具体的な手順について説明する．担当ノードが複数の子ノードを持ち，それぞれの子ノードは複数のプレイヤをサブスクライバの集合として持つとする．担当ノードは，ロビーサーバから複数のバックアップノードの集合を受信し，図 6 に示すバックアップノードキューに管理するものとする．担当ノード a は，イベントメッセージにタイムスタンプを付与して各子ノード c_i に送信することによって，a から c_i への通信遅延を c_i で測定できるようにする（ノード間の時刻の同期は NTP などのプロトコルで確保されていると仮定している）．また同様の方法で，キューの先頭にあるバックアップノード b までの通信遅延を b で計測する．バックアップノード b に対する遅延時間より大きい通信遅延を持つ子ノード c_j が存在した場合には，c_j と b を交換し，b を a の子ノードとする．b が交換されなかった場合，b をキューの先頭から削除し，最後尾に追加する．交換された場合は，c_j をキューの最後尾に追加する．以上の操作がバックアップキューの全てのノードに対し順次繰り返すことで，負荷分散木を更新していく．

以下，バックアップノードと中間ノードのシームレスな交換のための負荷分散木の更新アルゴリズムを以下に示す．

(a) 担当ノード a は，各イベント通知のためのパケットに現在時刻に相当するタイムスタンプを付与し，子ノード c_i に送信する．このとき，同じパケットをバックアップノード b にも配送する．

(b) a からのパケットを受け取ったバックアップノード b は，タイムスタンプと受信時刻の比較から，a-b 間のパケットの遅延時間 T_{a-b} を測定し，a の全ての子ノードからパケットを受け取るまで待つ．

(c) a の子ノード c_i は，a から受信したパケットに付与されたタイムスタンプと受信時刻から，a-c_i 間のパケットの遅延時間 T_{a-c_i} を計測し，それをバックアップノード b へ送る．

(d) b は，T_{a-b} と $\{T_{a-c_1}, ..., T_{a-c_k}\}$ を比較し，b 自身も含め a との間で最も大きな遅延時間を持つノード c_m を計算する．c_m が b 自身でない場合は，b が c_m と交代することを伝えるため，交代依頼メッセージを担当ノード a に送る．

(e) 交代依頼メッセージを受け取った担当ノード a は，c_m が持つサブスクライバの集合を b に送り，以降のイベント通知の送信先を c_m から b に変更する．

(f) b は，サブスクライバの集合を受け取ると，交代完了メッセージを c_m に送り，以後，a の子ノードとして振舞う．

(g) 交代完了メッセージを受け取った c_m は，負荷分散木から離脱する．

上記のアルゴリズムは，2.3.1 節で述べた手順のうち (3) 子ノードの分割時，(4) 部分木の統合時など，負荷分散木の中間ノードが増減する操作に対して排他的に適用する必要があるが，それ以外では，どのタイミングでも適用することができる．

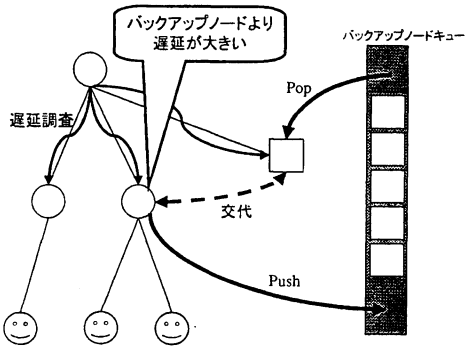

図 6: 負荷分散木の更新

3 実験と評価

2.3 節の負荷分散木の更新アルゴリズムによる配送遅延の改善度合いを調べるため，ns-2 によるシミュレーションを行った．シミュレーションにおける初期設定では，負荷分散木の担当ノードと中間ノード，それぞれに対するバックアップノードを無作為に選択した．評価する項目は，更新回数によるエンド・エンド配送遅延の推移と遅延時間がほぼ最小に収束するまでの時間の 2 点である．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 を用いて生成し，各リンクの遅延時間は，同一 LAN のノード間で 1ms，LAN–MAN 間が 10ms，MAN–WAN 間が 50ms となるよう設定した．総ノード数は 1011 である．このトポロジを用いて，ある領域のプレイヤの数を 100，担当ノー

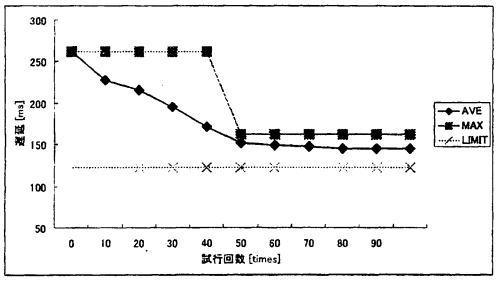

図 7: 負荷分散木の更新による遅延の推移

ドの数を 1，中間ノードの数を 10($k = 10$) としたオーバーレイネットワークを構築した．また，各プレイヤのイベント発生頻度を $e = 5/sec$，閾値 $C = 50$ と想定した．すなわち，担当ノードは，100 人のプレイヤから，毎秒 500 個のイベント通知を受け取り，負荷分散木を通してそれらのイベント通知を 100 人のプレイヤに配送する．$k = 10$ であるので，負荷分散木の段数は 1 段であり，各ノードのイベント配送数は毎秒 50 となる．

負荷分散木の更新による遅延の推移を図 7 に示す．10 個の中間ノードが全て更新されるには約 50 回程度の更新回数を必要とした．また，およそ 100 回程度の更新で配送遅延の最大 (図 7 の MAX) および平均 (図 7 の AVE) が最小値に収束している．今回の実験では，エンド・エンド配送遅延を 4 割程度削減することができている．また，中間ノードを経由することによる遅延の増加分（担当ノードからプレイヤノードに直接イベント配送する場合の遅延からの増加分）は，負荷分散木を更新しない場合に比べ 8 割以上削減できており，理論値に近い遅延時間（図 7 の LIMIT）まで削減することができている．物理ネットワークにおけるプレイヤノードと担当ノードの距離は担当ノードを入れ替えない限り改善できないため，提案手法によるさらなる劇的な遅延改善は難しいと考えられる．

プレイヤノードにとって，最適な中間ノードが既に同じ木の他の中間ノードとして使われている場合もあるので，中間ノードだけではなく，プレイヤノードも考慮した更新方法を考えることである程度の遅延改善は可能である．また，今回の実験では，交換用のバックアップノードを無作為に選んだため，あまり効果の高くない更新試行が多かった．ある程度以上の効果が見込める場合にのみ，バックアップノードと中間ノードの入れ替えを行うなどの方法で，より少ない更新回数で負荷分散木を効果的に更新することが可能と思われる．

4 あとがき

本稿では，多人数参加型ネットワークゲームのイベント配送機構における (1) 負荷分散木が用いられる際のエンド・エンドのイベント配送遅延の短縮方法，(2) 各プレイヤの視界が複数のゲーム領域に跨る場合のイベント配送方法，を提案した．

提案したイベント配送遅延の短縮方法は，プレイヤノードの離脱に備えたバックアップノードをイベント配送の中継ノードと動的に交換するもので，プレイヤの離脱や通信状況の変化による通信トポロジの変化にも柔軟に対応することができるという特徴を持つ．1011 ノードを持つ階層ネットワークを対象にシミュレーションを行い，更新アルゴリズムを用いない場合よりも 4 割程度通信遅延を削減できることを確認した．

今後の課題として，さらなるイベント配送遅延の削減のため，担当ノードの入れ替え，および中間ノードからプレイヤノードまでの遅延時間短縮を考慮した中間ノードの入れ替えを考慮したアルゴリズムを考案し，評価を行いたい．

参考文献

[1] 井芹，堀，藤川，下條，宮原：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).

[2] 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)* (2002).

[3] T. Iimura, H. Hazeyama and Y. Kadobayashi: "Distributed Scalabale Multi-player Online Game Server on Peer-to-Peer Networks"m IPSJ Digital Courier, Vol. 1, (2005).

[4] 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)

[5] 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).

[6] T. Henderson: "Latency and Behaviour on a Multiplayer Game Server", *Proc. of 3rd Intl. Workshop on Networked Group Communication (NGC2001), LNCS2233*, pp. 1–13 (2001).

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