2005－DSM－36（16）
2005／3／18

Peer-to-Peer 技術による大規模多人数参加型サービスの分散型構成法 － 信頼性の確保 －

遠藤 慶一† 川原 稔‡ 高橋 豊†

† 京都大学大学院 情報学研究科
‡ 愛媛大学 総合情報メディアセンター

概 要

本論文では，インターネットを通じてリアルタイムで双方向に情報を送受信する大規模多人数参加型サービスを取り扱う．この種のサービスはほとんどの場合，サービス提供者が管理する中央サーバにユーザが直接接続するクライアント・サーバモデルによって提供されているのが現状である．しかしこのモデルでは，計算負荷や通信負荷が中央サーバに集中するため，サービス提供者は高性能のサーバマシンや回線を用意しなければならない．そこで本研究では，Peer-to-Peer モデルに基づき，ユーザが利用するマシンにサーバ機能の一部を委譲することによって，サーバ負荷を分散させる構成法を提案する．

Peer-to-Peer モデルを採用する場合は，いかにしてサービスの信頼性を確保するかということが問題となる．本研究では，同じデータを複数のユーザマシンで管理して多数決方式を適用する．この仕組みによって，ユーザマシンに障害が起こった場合や悪意のあるユーザによるデータ改竄などの攻撃を受けた場合にも正常なサービス提供を続けることができる．

A Distributed Architecture for Massively Multiplayer Online Services with Peer-to-Peer Support: Cheat-Proof Framework

Keiichi Endo† Minoru Kawahara‡ Yutaka Takahashi†

†Graduate School of Informatics, Kyoto University
‡Center for Information Technology, Ehime University

Abstract

This paper deals with Massively Multiplayer Online Services, users of which communicate interactively with one another in real time. At present, this kind of services is generally provided in a Client-Server Model, in which users connect directly with a central server managed by a service provider. However, in this model, since a computational load and a communication load concentrate on the central server, high-performance server machines and high-speed data links are required. In this paper, we propose an architecture for distributing the loads by delegating part of the server's function to users' machines, which is based on a Peer-to-Peer Model.

When we adopt a Peer-to-Peer Model, we face a security problem. In our research, we let multiple machines of users manage the same data and apply a decision by majority rule. This mechanism adds robustness against halts of users' machines and data tampering by malicious users to the service.

1 はじめに

近年，オンラインゲーム，オンラインチャット，オンラインオークション，オンライントレードなど，ユーザがリアルタイムで情報を送受信するようなサービスの利用者が増えている．本論文では，この種のサービスのことを大規模多人数参加型 (Massively Multiplayer Online; MMO) サービスと呼ぶことにする．

現在，MMO サービスはほとんどの場合，サービス提供者が管理する中央サーバにユーザが直接接続するクライアント・サーバモデルによって運営されている（図 1）．しかしこのモデルには，計算負荷や通信負荷が中央サーバに集中するという欠点がある．このモデルで MMO サービスを運営するためには，高性能なサーバマシンや高速な通信回線が必要である．そのため，Peer-to-Peer (P2P) モデルによってこのサービスを運営する手法に関する研究が活発に行われている [1][2]．図 2 は本研究で提案するモデルを示したものである．P2P モデルでは，ユーザの CPU 処理能力，メモリ，通信帯域をサービス運営に利用するので，サーバマシンにかかる負荷は軽減される．しかし，実用のためにはいくつか解決しなければならない問題がある．まず，ユーザマシンの障害に対するロバスト性が必要である．すなわち，あるユーザが突然ネットワークから切断された場合でも，サービスが停止したりデータが消失したりすることがあってはならない．次に，ネットワークの混雑の影響を最小限に抑えなければならない．ユーザマシンは数秒間にわたって通信ができない状態になることも多い．最後に，サービスはユーザの不正行為に影響されてはならない．P2P モデルではサーバ機能の一部をユーザマシンに委譲するため，一般にデータ改竄などの不正を行いやすい．

P2P モデルにおいて，不正行為は非常に深刻かつ難しい問題である．オンラインゲームにおける不正対策に関しては多くの研究があるが [3][4][5]，これらの方法では状態データを改変することによる不正に対応することはできない．P2P モデルでは，状態データが信頼できないマシンに管理されるという構造が，この問題の解決を困難にしている．本論文では，複数のユーザマシンに同じデータを管理させて，多数決方式を適用することにより，上述の問題の解決を試みる．

以下，第 2 章では本論文で提案するモデルについて説明し，第 3 章ではアルゴリズムを詳しく説明する．そして第 4 章でまとめと今後の課題を述べる．

2 MMO サービスにおける P2P モデル

MMO サービスは多数のユーザが同時に利用するが，ユーザは常に限られた範囲しか見ることができない．本論文ではこの範囲のことをサイトと呼ぶ（図 3）．すなわち，MMO サービスにはいくつかのサイトがあり，ユーザはそれぞれ 1 つのサイトに属している．ユーザはあるサイトから別のサイトへ移動することが可能である．ユーザのアクションは，同じサイトにいるユーザのみにリアルタイムで伝えられる．この性質により，サイトのサービス機能をユーザマシンに移すことが可能となる．サイトを管理しているユーザマシンのことを今後はサイトサーバと呼ぶことにする．

図 4 は，サイトサーバがどのように 1 つのサイトを管理するかを示したものである．ユーザは，自分がいるサイトを管理する全てのサイトサーバと接続

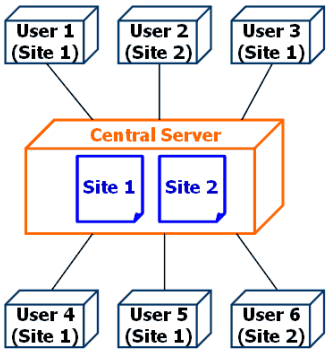

図 1: Client-Server Model.

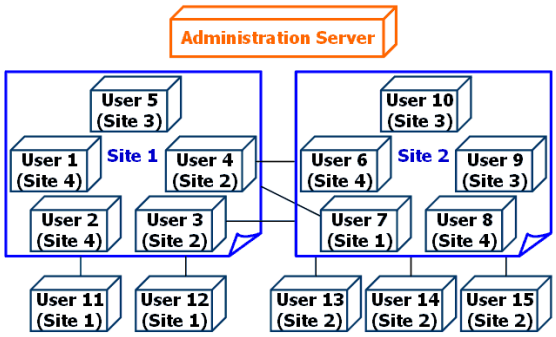

図 2: Example of P2P Model.

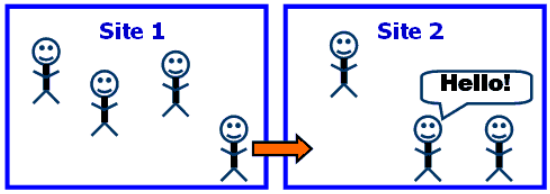

図 3: Image of MMO service.

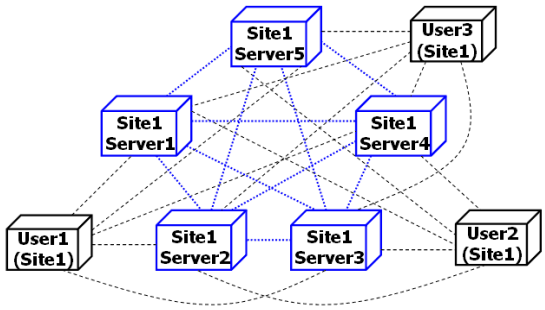

図 4: Model of managing a site.

している．同じサイトを管理している全てのサイトサーバは，完全グラフを形成している．ユーザがアクションを起こしたときは，そのアクションを伝えるために全てのサイトサーバにメッセージを送信する．そして，サイトサーバは状態データを更新する．状態データには，ユーザの状態データとサイトの状態データの２種類がある．たとえばロールプレイングゲームサービスの場合は，ユーザの位置がユーザの状態データに，マップ（サイト）に置かれたアイテムの情報がサイトの状態データにそれぞれ相当する．サイトサーバが状態データを更新する際には同期をとって，他のサイトサーバと同じデータを保持するようにする．最後に，サイトサーバはそのサイトにいるユーザに状態データの更新を伝える．

3　アルゴリズム

本章では，具体的なアルゴリズムを説明する．

3.1　ユーザのログイン

MMO サービスを利用するには，まず初めに管理サーバに接続し，ユーザ ID とパスワードを送信して認証を受ける．

管理サーバが認証情報を受け取ったときは，指定されたユーザのアカウント情報を参照して，パスワードによって情報の送信元が正当なユーザであるかどうかを確認する．(もしそうでなければ，エラーメッセージを返送する.）認証後，管理サーバはそのユーザの状態データをディスクから読み込んで，そのデータをユーザに返送する．もしそのユーザがいるサイトのサービス機能が他のユーザのマシンに委譲されていれば，管理サーバはここでさらに処理を行う必要がある（たとえばサイトサーバのアドレスを送信するなど）．これについては後に「3.6 ユーザのサイトへの配置」で述べる．

管理サーバから状態データを受け取ることができれば，ユーザは MMO サービスの利用を始めることができる．

3.2　サイトのサービス機能の委譲

同時接続ユーザ数が少ないとき，MMO サービスは通常のクライアント・サーバモデルで運営される．すなわち，全てのサイトのサービス機能は管理サーバの中にある．ユーザ数が増えるにしたがって，管理サーバにかかる負荷が高くなっていく．そこで，ある条件でサイトのサービス機能はユーザのマシンに委譲される．ある条件とは，たとえば「サイトにいるユーザの人数がある値に達したとき」などである．この仕組みによって，ユーザマシン（以下ではピアとも呼ぶ）は徐々に（ハイブリッド）P2P ネットワークを形成する．

管理サーバがサイトのサービス機能を委譲することを決めたときは，サービス機能の委譲先として N_{server} 個のピアを選ぶ．ここで N_{server} はあらかじめ定められた奇数である．ピアの選び方としては様々なものが考えられるが，単純な方法としてはまだサイトを管理していないピアの中からランダムに選ぶというものがある．ただし不正行為の効果を小さくするため，サービス機能を委譲しようとしているサイトにいるユーザを選ぶことは避ける．

委譲先のピアを選んだ後，管理サーバはそのサイトにいる全てのユーザに対して Site Server Address メッセージを送る．Site Server Address メッセージは選ばれたピア（新しいサイトサーバ）に接続するための情報である．このメッセージはサイトサーバの IP アドレスやポート番号だけでなく，サイトサーバとの通信を開始するためのセッション ID も含まれている．セッション ID は，それぞれのクライアントとサーバの組に対してランダムに生成された値であり，認証目的として十分な長さを持っている．新しいサイトサーバに対しては，管理サーバは以下のデータを送信する．

- サービス機能が委譲されるサイトの番号.
- サイトの状態データ.
- そのサイトにいる全てのユーザに対応したセッション ID.
- ユーザ番号とユーザの状態データ.
- 今回選ばれた他の全てのサイトサーバと接続するための情報．IP アドレス，ポート番号，セッション ID から構成される．セッション ID は任意の２つのサイトサーバの組に対して生成される．

これらのデータを受け取った後，ピアはサイトサーバとしてサイトの管理を始める．新しいサイトサーバは他の全てのサイトサーバと接続することによって全体で完全グラフを形成する．

ユーザが Site Server Address メッセージを受け取ったときは，指定されたサーバに接続して，メッセージに含まれているセッション ID を送信する．もしユーザが正しいセッション ID を送らなければ，サイトサーバによって接続が切断される．新しいサイトサーバも，他のサイトサーバと通信するためにセッション ID を利用した同様の手続きが必要である．

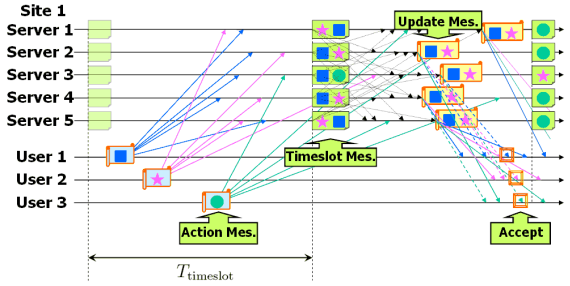

図 5: How to handle Action Messages.

3.3 アクションメッセージの処理

ユーザがログイン・ログアウト以外のアクションをとったときは，アクションの内容を表すアクションメッセージを作成する．そのユーザが管理サーバによって管理されているサイトにいた場合は，管理サーバにそのメッセージを送信する．管理サーバがそのメッセージを受信するとすぐにメッセージを処理して状態データを更新する．アクションメッセージによって変更されるのはそれを送信したユーザの状態データだけに限らず，サイトの状態データやそのサイトにいる他のユーザの状態データが変更される場合もある．そして，アップデートメッセージをそのサイトにいる全てのユーザ（またはその状態更新を知る必要のあるユーザ）に対して送ることによって，管理サーバは状況の変化を知らせる．

一方，そのユーザがサイトサーバによって管理されているサイトにいた場合は，以下の手順が実行される（図 5）．まずユーザはそのサイトを管理している全てのサイトサーバにアクションメッセージを送信する．そのメッセージには識別のためのアクション番号が含まれている．それぞれのサイトサーバはそのメッセージを受信した後，ユーザ番号，メッセージの内容，およびメッセージの受信時刻を記憶する．サイトサーバの時計は NTP (Network Time Protocol) のような仕組みによってあらかじめ同期がとられているものとする．管理サーバの場合と異なり，サイトサーバはこの時点ではメッセージの処理を行わない．サイトサーバ間でアクションメッセージの処理順序を合わせる仕組みが必要である．本研究のシステムでは，それぞれのサイトサーバへの到着時間の中央値を基準としてアクションメッセージの処理順序を決定する．中央値をとる理由は，不正行為の影響を抑えるためである．次段落で詳細を示す．

定められた時間間隔 T_{timeslot} ごとに，サイトサーバは同じサイトを管理する他の全てのサイトサーバにタイムスロットメッセージを送信する．そのメッセージは次の情報から構成される．

- サイト番号．
- そのときの時刻（タイムスタンプ）．
- 前回タイムスロットメッセージを送信してから受信した全てのアクションメッセージに関する { ユーザ番号, アクション番号, 到着時刻 } の組．

サイトサーバがタイムスロットメッセージを受け取ったときは，そのメッセージを記憶し，以下に示すアルゴリズムで状態データを更新する．

1. N_{majority} 個以上のサイトサーバ（自分自身を含む）から情報を受け取っているアクションの中から，N_{majority} 番目に早い到着時刻が，各サイトサーバからの最新のタイムスロットメッセージのタイムスタンプよりも古いアクションを選ぶ．N_{majority} は，同じサイトを管理するサイトサーバの数の半分より大きい最小の整数である．すなわち，
$$N_{\text{majority}} = \frac{N_{\text{server}} + 1}{2}. \qquad (1)$$
このステップは，到着時刻の中央値が確定したアクションを選ぶことを意味する．もしそのようなアクションがなければ，状態データの更新は行われない．
2. サイトサーバは N_{majority} 番目に早い到着時刻をもとにした順序で，選ばれたアクションに対応するアクションメッセージを処理することにより，状態データを更新する．もしまだ受信していないアクションメッセージがあれば，そのアクションを起こしたユーザからそのメッセージが到着するまで待つ．
3. ユーザにアップデートメッセージを送信する．

悪意のあるユーザやネットワークの混雑がなければ，ユーザはそのサイトの全てのサイトサーバから同じアップデートメッセージを同じ順序で受け取ることになる．不正を行うユーザの影響を受けないようにするため，ユーザは N_{majority} 個の同じアップデートメッセージを異なるサイトサーバから受け取った後に，アップデートメッセージの内容を受け入れ，画面を更新する．

3.4 ネットワーク混雑の対策

前節で述べたアルゴリズムは，ネットワークの混雑に対応できるような改良を必要とする．このままではアクションメッセージ遅延（アクションメッセージが送信されてからアップデートメッセージがユーザに受け入れられるまでにかかる時間）がネットワークの混雑に大きく影響されるからである．たとえば，あるサイトサーバから N_{majority} 個以上の

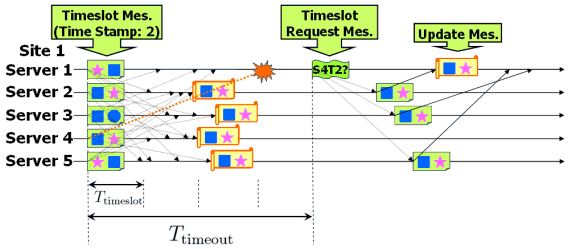

図 6: How to settle the long delay of the Timeslot Message from Server 4 to Server 1.

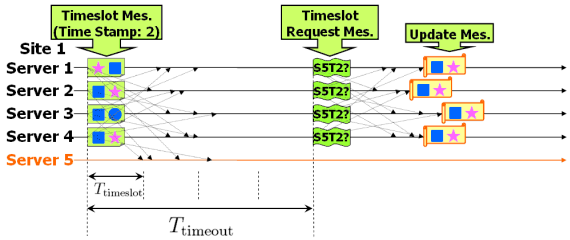

図 7: How the mechanism works when Server 5 suddenly halts.

サイトサーバへのタイムスロットメッセージが遅れると，そのサイトのサービス機能は停止してしまう．以下に述べる方法でこの問題を解決する．

T_{timeout} を T_{timeslot} より大きい定数とする．もしあるサイトサーバが，他のサイトサーバからのタイムスロットメッセージを，最後に来たタイムスロットメッセージのタイムスタンプより T_{timeout} だけ後の時刻になっても受け取っていないときは，同じサイトを管理するサイトサーバのうち，タイムスロットメッセージが来ていないサーバ以外の全サーバに Timeslot Request メッセージ (TRM) を送信する (図 6)．TRM はサイト番号，タイムスロットメッセージが欠けているサイトサーバの識別番号，そして欠けているタイムスロットメッセージのタイムスタンプから構成される．

サイトサーバが TRM を受け取ったときは，要求されているタイムスロットメッセージを以前に受け取ったかどうかを調べる．もし受け取っていれば，要求されているメッセージと全ての後続のタイムスロットメッセージを Timeslot Request Reply メッセージ (TRRM) として返送する．

TRM を送信したサイトサーバが TRRM を受け取ったときは，そのメッセージに含まれているタイムスロットメッセージを使って状態データの更新を行う．TRM を送信した後に，到着が遅れていたタイムスロットメッセージが元来の送信元から届いた場合は，そのタイムスロットメッセージを無視する．TRM を送信した全てのサイトサーバから TRM を受信したとき，または TRM を送信してからある時間 $T_{\text{req\_timeout}}$ が経過したときは，タイムスロットメッセージが来ていないサイトサーバをある時間 T_{ignore} だけ無視する（図 7）．サイトサーバを無視するとは，そのサーバからのタイムスロットメッセージを無視し，状態データ更新時にもそのサーバを除外してユーザアクションの順番付けを行うという意味である．このアルゴリズムによって，あるサイトサーバから無視されたサイトサーバは，通常同じサイトを管理する他の全てのサイトサーバにも無視されることになる．もし時間 T_{ignore} が経過する前に到着が遅れていたメッセージが無視されているサーバから届いたときは，その時間が経過した後そのサーバはサイト管理に復帰する．サイト管理に復帰するときは，同じサイトを管理する他の全てのサイトサーバから状態データを集めて，それらの多数派を自分が保持する状態データとする．もし時間 T_{ignore} が経過しても到着が遅れているメッセージが届かないときは，管理サーバに代役を選ぶよう要求する．この仕組みによって，サービスはユーザマシンの停止に対するロバスト性を持つ．

3.5 サイトの移動

ユーザがサイトを移動するようなアクションをとったときは，そのときにいるサイトと移動先のサイトが両方管理サーバによって管理されている場合を除いて，管理サーバがサイト移動のための処理を行う必要がある．そのときにいるサイトがサイトサーバによって管理されている場合は，サイトサーバが管理サーバに要求メッセージを送る．その要求メッセージにはユーザ番号とそのユーザの状態データが含まれている．また，移動先のサイト番号も含まれている．不正に対するロバスト性を確保するため，移動元サイトを管理する全てのサイトサーバからの要求メッセージの多数決がとられる．すなわち，管理サーバは移動元サイトを管理するサイトサーバから N_{majority} 個の同じ要求メッセージを受け取るまでは何もしない．そして次に，移動元サイトがサイトサーバに管理されているかどうかにかかわらず，管理サーバは次節で述べる方法でユーザを移動先のサイトに配置する．

3.6 ユーザのサイトへの配置

本節で述べる手順はユーザがサイト移動を伴うアクションをとったときだけでなく，ユーザがログインしたときにも実行される．もし配置先サイトのサービス機能が管理サーバの中にあるとき，管理サーバは単にサイトが管理サーバによって管理されていることをユーザに伝える．そのサイトがサイトサーバによって管理されている場合は，管理サーバはサイトサーバに接続するための Site Server Address メッセージをユーザに送る．Site Server Address メッセージの内容はすでに述べたとおりである．また，

管理サーバはサイトサーバに対しても以下のデータを送る．

- サイト番号．
- 配置するユーザの状態データとユーザ番号
- そのユーザと通信するためのセッション ID

ユーザは Site Server Address メッセージを受信後，指定されたサーバに接続することによって新しいサイトで活動を始めることができる．

すでに述べたように，サイトがそのサイトにいるユーザによって管理されることは望ましくない．そこで，もしユーザがそのユーザによって管理されているサイトに移動するときは，管理サーバが代替のサイトサーバを選び，新しいサイトサーバにそのサイトに関する全てのデータを送信するようサイト移動するユーザに対して命じる．管理サーバは他のサイトサーバやそのサイトにいる全ユーザに対しても，サイトサーバが変わったことを知らせる．

3.7 ユーザのログアウト

ユーザが MMO サービスの利用を終了するときは，管理サーバにその意向を伝える．そのユーザがサイトのサービス機能を持っている場合は，そのサイトに関する全データも管理サーバに送信する．そしてユーザは全サーバとの接続を切断し，サービスのソフトウェアを終了させる．

管理サーバはさらに次のような処理を行う必要がある．

1. ユーザによって管理されていたサイトに関するデータを受け取ったときは，新しく選ばれたサイトサーバにそのデータ（および新しいセッション ID）を送信する．さらに，そのサイトの他のサイトサーバや，そのサイトにいる全ユーザにサイトサーバの変更を伝える．もし総ユーザ数が少なすぎて代替のサイトサーバを見つけることができない場合は，サイトサーバからそのサイトに関する状態データを収集し，それらの多数決をとる．そして，そのサイトにいる全ユーザにサービス機能の移動を伝えた後，管理サーバがそのサイトを管理し始める．
2. 管理サーバがログアウトしたユーザの状態データを保存する．サイトサーバがそのデータを持っている場合は，サイトサーバにそのデータを送り返すよう要求し，それらのうちの多数派を保存する．

4 まとめ

本論文で提案したアーキテクチャには既存の研究と比較して以下のような特徴がある．

1. 同じデータを管理する複数のマシンで多数決をとるという方法を MMO サービスに適用したこと．この仕組みによってデータの改竄や消失に対する信頼性を高めることができる．さらに，サービスはネットワーク混雑に対するロバスト性を持っている．
2. サイトを管理するユーザは，そのサイトにいないユーザの中から管理サーバが選択すること．これによって不正を行う利点が減り，結託も難しくなる．
3. アクションメッセージの処理順序が，それぞれのサイトサーバへの到着時刻の中央値を基準として決定されること．これは，メッセージに含まれるタイムスタンプの改竄による影響を少なくする効果がある．
4. 全てのユーザマシンがサイトを管理しなければならないわけではないということ．したがって，サイトのサービス機能を委譲するルールを適切に決めれば，低速回線，低性能マシンを利用しているユーザや，ファイアウォール（または NAT ルータ）の裏にいるユーザでもサービスを利用することができる．

今後の課題としては，まず本論文で提案したモデルを用いたシミュレーションを行いたい．そして，実際の MMO ゲームサービスを利用した実験により，実用性を検証したいと考えている．

参考文献

[1] Christophe Diot and Laurent Gautier: “A Distributed Architecture for Multiplayer Interactive Applications on the Internet,” *IEEE Networks Magazine*, Volume 13, Issue 4, pp. 6-15, 1999.

[2] Bjorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins: “Peer-to-Peer Support for Massively Multiplayer Games,” *Proceedings of IEEE INFOCOM 2004*, Hong Kong, China, March 2004.

[3] Nathaniel E. Baughman and Brian Neil Levine: “Cheat-proof Playout for Centralized and Distributed Online Games,” *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[4] Eric Cronin, Burton Filstrup, and Sugih Jamin: “Cheat-Proofing Dead Reckoned Multiplayer Games,” *Proceedings of ADCOG 2003*, Hong Kong, January 2003.

[5] Chris GauthierDickey, Daniel Zappala, Virginia Lo, and James Marr: “Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games,” *Proceedings of NOSSDAV 2004*, Kinsale, County Cork, Ireland, June 2004.