ビザンチン合意を用いたP2Pオンラインゲーム用不正対応プロトコル

和田 大介[†] 小林 洋[‡]

東海大学大学院工学研究科 〒259-1292 神奈川県平塚市北金目 1117
E-mail: [†]7adrm030@ keyaki.cc.u-tokai.ac.jp, [‡]koba@tokai.ac.jp

本研究では，オンラインゲームをピュア型 P2P（peer to peer）ネットワークにおいて行なった場合のプレイヤ間の時間チート(time-cheat)と呼ばれる不正行為の防止を行うプロトコルの一種を提案する．本方式では，分散ゲームにおいて時間チートを防止するための基本的なプロトコルであるロックステップ（lockstep）プロトコルにビザンチン合意(Byzantine agreement)アルゴリズムを組合せて用いており，あるプレイヤが不正行為を行っても他のプレイヤは引き続きゲームを継続することが可能になる．

Cheat Tolerant Online Game Protocol for P2P Using Byzantine Agreement

Daisuke Wada[†] and Hiromi Kobayashi[‡]

Graduate School of Engineering, Tokai University 1117 Kitakaname, Hiratsuka, 259-1292, Japan

We present an online game protocol to prevent dishonest activity called time-cheat for peer to peer (P2P) network. This protocol based on the lockstep protocol which prevents time-cheat in a distributed game and Byzantine agreement algorithm. A game can be continued even if a player performs a cheat action in this protocol.

1. はじめに

インターネットの進歩により，現在オンラインゲームが数多く開発され普及してきている．これらの現行のオンラインゲームでは，ゲームの管理サーバにクライアントがアクセスすることで情報のやり取りが行われゲームが進行する形式である．しかしながら，この方式ではサーバのアクセス許容量によってクライアント数が制限されてしまう．さらには，サーバの処理能力に依存しているため，ゲームの進行が滞ってしまう状況が発生する．オンラインゲームにおいてはデータの遅延を利用した不正も行われる可能性がある．例えば，suppress-correct cheat[4]等がこの種の不正行為として良く挙げられる．suppress-correct cheatとは，クライアント間の通信速度や処理速度を意図的に低速にすることによって，自分の行動を不規則にし，相手に不利になるようにすることである．このような過負荷による遅延の問題を解決するためには，サーバでの処理を各クライアントつまりピア（Peer）に分散することが考えられる[1-3]．分散の方式は，サーバを設けるか否かによっていくつかの方式が考えられるが，本研究では，サーバでの全て処理を分散化し各ピアで行うピュア（pure）型 P2P 方式を用いた処理方式を取ることにする．P2P 方式，特にピュア型 P2P 方式を用いたオンラインゲームでは管理サーバが無いため，あるピアが不正行為を行った場合，その処理を行うための仕組みを分散シ

ステム内に設ける必要がある．そこで本研究では，ピュア型 P2P でオンラインゲームを行うための基本プロトコルの一方式を提案する．本方式では，分散オンラインゲームにおいて各ピアの処理の同期を取り時間チートを防止するためのプロトコルの中で基本的なプロトコルであるロックステップ（lockstep）プロトコル[4-6]に分散コンピューティング上での故障検知技術であるビザンチン合意(Byzantine agreement)アルゴリズム[7-12]を基本としたインタラクティブコンセンサス問題[11]のアルゴリズムを組合せて用いており，あるプレイヤが不正行為を行っても他のプレイヤは引き続きゲームを継続することが可能になる．

なお，文献[13]では，公開鍵暗号と署名の利用及び各プレイヤが持つゲームのシミュレータによりチートに対処しているが，想定しているゲームや対象とするチートが一部を除き本研究とは異なる．

2. 基本プロトコル

本研究で提案するプロトコルに使用するロックステッププロトコルとビザンチンアルゴリズムについて以下に示す．

2.1 ロックステッププロトコル

ロックステッププロトコル（lockstep protocol）は，オンラインゲームなどで利用されている各ステップの同期のための逐次停止型プロトコルの基本的なものである．オンラインゲーム上ではレスポンスの時間差や通信速度による時間差によって，有利不利が発生してしまうことがある．また，従来オンラインゲーム使われていた単純な逐次停止プロトコルでは lookahead cheat といわれる後出しジャンケンのような不正を防ぐことができない．このような問題を解決するため，相手に自分の送るべきデータのハッシュ（hash）値を送信しておき，全員の同期がとれたことを確認できたところで，本来のデータを送る．その後各自でこのデータのハッシュ値を取り先ほどのハッシュ値と等しいかどうかを確認することにより，この回において後出しなどの時間チート（time-cheat)が起きていないことを確認することができる．この動作についての説明のための図を図 1 に示す．図 1 では，Player1 と Player2 の二人のプレイヤでゲームをする場合である．まず，makeHush()にてお互い共通のハッシュ関数を用いてデータをハッシュ化する．Data1 と Data2 はそれぞれのプレイヤが本来決定していた行動データである．Send()にてゲームの他の参加者にハッシュ値を送信する．各プレイヤは他者のハッシュ値を確認したら，受信確認のメッセージを全員に送る．各プレイヤは他者からの受信確認のメッセージを受けたら，再び Send()にて行動データを送信する．二つのデータ(ハッシュ値，行動データ)が揃ったら送られてきたデータをハッシュ化し，ハッシュ値の比較を行なうことにより，後出しなどの時間チー

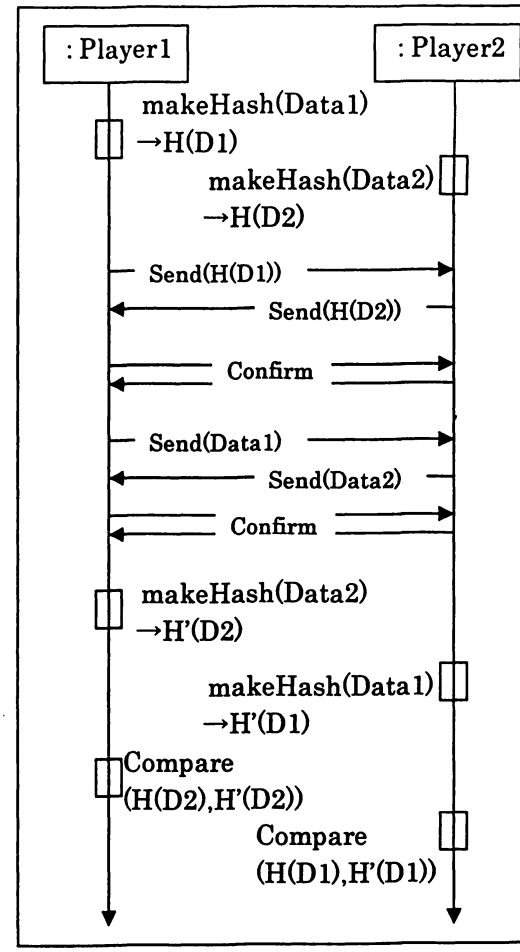

図 1　ロックステッププロトコル

ト（time-cheat)が行われていないことを判断することができる.

2.2 ビザンチンアルゴリズム

分散コンピューティングの環境において，すべてのピアが正確に動作をしていれば問題はないが，時には一部のピアが故障などによって不正な行動をしてしまうことがある．特にビザンチン故障と呼ばれるものは故障したピアの動作が不確定で，正常な値を出力したり，不正な値を出力したりする．この問題を解決するために，ビザンチン合意アルゴリズム（Byzantine agreement algorithm）を用いることにより，そのような状況に陥った時でも，個々のピアが多数決をとることで正常な値を取得し処理を続行することができる．なお，ビザンチンアルゴリズムでは，不正な行動をするピア数を k とした場合，全部で $n \geq 3k+1$ 個のピアがシステム内に必要となる．

本方式では，ビザンチンアルゴリズムの内，OM(Oral Message Algorithms)と呼ばれる再帰的アルゴリズムを使用する．以下にアルゴリズム OM(m)を記述する[10].

アルゴリズム OM(0)

1：親ピアは自分の値を全ての子ピアに送る.

2：各子ピアは親ピアから受信した値を用いる．何も受信しなければ指定された値を使用する.

アルゴリズム OM(m)，m>0

1：親ピアは自分の値を全ての子ピアに送る.

2：子ピア i が親ピアから受信した値を Vi とし，何も受信しなければ指定された値を Vi とする．子ピア i は自分が親ピアであるかのようにアルゴリズム OM(m-1)で値 Vi を残りの(n-2)人の子ピアに送る.

3：ステップ 2 で子ピア i が子ピア j から受信した値を Vj とし，何も受信しなければ指定された値を Vj とする．子ピア i は majority($V_1,V_2,\cdots,V_{n-1}$)の値を用いる.

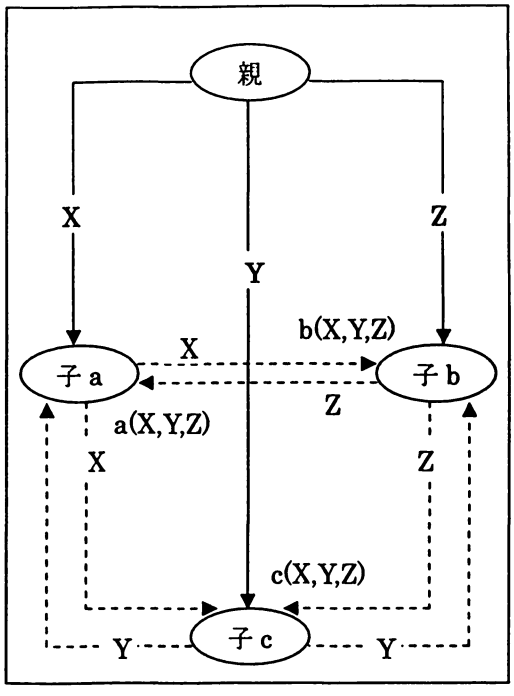

図 2　ビザンチンアルゴリズム OM(1)の振る舞い

この OM(m)を用いることにより，親ピアが故障をして子ピアに対し異なる値を送信しても，子ピア同士の多数決によりある共通の値で合意することができる．また，子ピアが故障をしていても，k 個以内の故障であれば親ピアと残りの子ピアのデータによって多数決を行い，ある共通の値を得ることができる．図 2 では，n=4，k=1 の場合で親ピアが送信した(X,Y,Z)のデータを，それぞれ子ピア a，子ピア b および子ピア c が受け取り，更に子ピア同士での送受信を行う振舞いを表したものである．このような動作をすることで，子ピアは親ピアが送信した全てのデータを確認することができ，通信の途中で一部のデータが改ざんされていても，全ての子ピアが共通のデータを受け取ることができる仕組みになっている.

3. ゲームのモデル

3.1 オンラインゲームと処理

本方式では，オンラインゲームをピュア型 P2P ネットワークで行うモデルを想定している．このオンラインゲームでは，プレイヤ同

士がある一定のタイミングでその時点でのステータスのやり取りを行い，そのデータを基に各自がゲームの進行を進めていくと想定している．例えば，リアルタイムストラテジー(Real-Time Strategy)と呼ばれる種類のゲームである．このRTSゲームでは，プレイヤが神や指揮官などの第三者的視点で兵士などのキャラクタに指示を与え，敵を倒させる展開となるのが特徴である．命令や行動の順番が明確に決まっているいわゆるターン制のストラテジーゲームとは違い，リアルタイムに進行していく状況に対しプレイヤがプランを立て，アクションを起こしていくゲームである．この有名タイトルとしては，「Age of Empires」などがある．これらのゲームではプレイヤは司令官としてゲーム上のキャラクタに対しリアルタイムに命令を下し，その行動をフィールド上で展開させる．その展開を見た他のプレイヤはその状況を判断しながら戦略を考え，自軍のキャラクタに同じくリアルタイムな命令を与えることでゲームの進行を行なうことになる．

このようなゲームをP2Pネットワークで実施するには，自分の行動や状況をステータスとして対戦相手に送信する必要がある．相手はそのステータスを受け取り，自分のゲーム上の相手の行動としてゲーム上に反映させる．この送受信をゲームに参加しているプレイヤ間で繰り返す．この際，時間の進行を遅らせるような不正や，タイムラグによって相手の次の行動をあらかじめ知るような不正を防ぐために，オンラインゲームなどで利用され，各ステップの同期を取るための基本的プロトコルであるロックステッププロトコルを使用する．親ピア（データを送信するピア）は自分のステータスをハッシュ化して送信し，子ピア（データを受信するピア）は送信されたハッシュ化データを受け取る．その後，子ピアは子ピア同士で親ピアから送信されたハッシュ化データを送受信し，ビザンチンアルゴリズムに基づき全プレイヤのハッシュ化データの多数決を取ることでハッシュ化データの同意を得るものとする．これらの行動を全てのゲーム参加者が行なった後，再びステータス自体を送受信する．ハッシュ化データとステータスが揃ったプレイヤは，その二つのデータからデータに改ざんが無いかどうかを確認し，ゲームに反映させる．

3.2 不正行為の種類

本方式が対象とする不正行為について，以下に示す．

不正行為1：時間差を利用した不正

自分のデータを送信するタイミングを意図的に遅らせることによって，先に入手した相手のデータを基に自分の対応を決定した後に相手にそのデータを送る行為

不正行為2：異なるデータを送信する不正

相手に応じて異なったデータを送信し，ゲームを混乱させる行為で，1対多又は多対多のゲームではこの行為は脅威となる．

3.3 プロトコル

本方式を用いたゲームモデルについて以下に示す．ゲームでは，まず，ゲームに参加するピアと，親の順番を決定する必要がある．次に親となったピアは自分の行動データをゲーム参加者共通のハッシュ関数にてハッシュ化する．子となったピアは親ピアからのデータ送信を待つ．親ピアはハッシュ化したデータを子ピア全員に送信する．子ピアは送信されたハッシュ化データを受け取り，子ピア同士でそれぞれ送信しあう．子ピアは子ピア同士で送受信したデータを用いて多数決をとり，ハッシュ化データを決定する．子ピアは多数決で決定したデータとあるピアのデータが一致していない時，不正を起こしていると判断しログを取る．次に，親ピアを変更し，同様の処理を行い，全てのピアが親となるまで繰り返し実行する．つまり，インタラクティブコンセンサス問題として処理する．　次に親ピアは行動データを子ピアに送信する．子ピアは先程と同じようにしてデータの送受信を行い，多数決をとり行動データを決定する．ここでも子ピアは多数決で決定したデータとあるピアのデータが一致していない時には不正とみなし，ログを取る．これらの処理をハッシュデータの処理の時と同様に，親ピアを

次々と変更して実行する．最後に，各ピアで行動データをハッシュ化して，先に送られてきたハッシュ化データと比較し，行動データの正当性を検証する．

次に，本稿で提案するプロトコルの基本部分について図3に示す．前処理として，まず初めに親を実行する順番を決定しておく．

なお，使用するハッシュ関数は全てのピアで共通のものを使用する．また，以下においては $n \geqq 3k+1$，$m=n/3$ とする．

(1)～(4)は，行動データのハッシュ値を各ピアに送る部分である．(1)で親はデータをハッシュ化し，以下の部分で送信を行なっている．

(5)～(7)は，行動データを各ピアに送る部分である．

(8)では，初めに受け取っていたハッシュ値と行動データをハッシュ化したものを使用し，ロックステッププロトコルとしての検証を行なっている．

(9)では，入手したデータをゲームに反映させる処理を行なう．

これらの処理を一定時間ごとに繰り返し行なうことで，ゲーム進行に必要なデータの送受信を達成することができる．

更に，不正を起こしたピアをゲーム上から排除するために，データの多数決をする際に送られてきたデータのログを取り，以下のような処理を行う．

ピア p(i)のログから，他のピア p(j)が合意に達していないと判断できる回数がある値（閾値）を超えたのならば，ピア p(j)は不正を起こすピアであるとピア p(i)では判断する．不正ピアの存在を確認したピア p(i)は，グループ内の他のピア p(k)($k=1,\cdots,n$ 但し，$k \neq i,j$)にピア p(j)が不正ピアであるとメッセージを送信し，多数決を取る．多数決が取れた場合，ピア p(j)は不正ピアであると決定し，各ピアはピア p(j)をゲームから排除する．

4. 評価

本研究で提案したプロトコルが正しく動くことを確認するために，シミュレーションプログラムを作成しアルゴリズムのメッセージ数の評価を行った．

```
repeat
  (1)親ピアは自分の行動データを
     ハッシュ関数でハッシュ化する．
  (2)アルゴリズム OM(m)で再帰的に
     ハッシュ値を全てのピアに送る．
  (3)親ピアは全ての子ピアから終了確認の
     メッセージを待つ，子ピアは多数決に
     よって値を得た後に親ピアへ終了の
     メッセージを送る．
  (4)全てのピアで親を次のものにする．
until(全てのピアが親として実行するまで)

repeat
  (5)アルゴリズム OM(m)で再帰的に行動
     データを全てのピアに送る．
  (6)親ピアは全ての子ピアから終了確認の
     メッセージを待ち，子ピアは多数決に
     よって値を得た後に親ピアへ終了の
     メッセージを送る．
  (7)全てのピアで親を次のものにする．
until(全てのピアが親として実行するまで)

(8)全てのピアは行動データをハッシュ化し，
   そのハッシュ値と先に手に入れていた
   ハッシュ値を比較し，チェックを行なう．
(9)行動データをゲームに反映させる．
```

図3　提案する基本プロトコル

n 個のノードの内 k 個故障のあるビザンチンアルゴリズムの子同士のメッセージ数は，$(n-1)*(n-2)*\cdots*(n-k-1)$ である（但し，$n \geqq 3k+1$）．しかしながら，今回このプロトコルを現状の RTS ゲームに適応して考えると，ゲームに参加するピア数は平均 4～5 台，上限は高々10 台程度で，不正を働くピアは高々1 台程度ではないかと考えられる．その場合の子同士のメッセージ数は，$(n-1)*(n-2)$となる．これに親から子へのメッセージ数 $n-1$ を加え，更に，インタラクティブコンセンサス問題として処理をしているため全てのピアが親になることから，メッセージ数は，

$$n*\{(n-1)+(n-1)*(n-2)\}= n*(n-1)^2$$

表１．ピア数とメッセージ数

ピア数	メッセージ数
4	72
25	2880
46	186300
67	583704
88	1332144

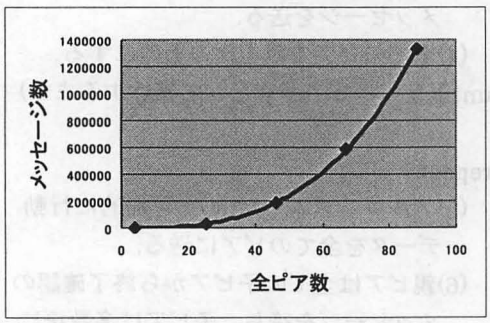

図４　ピア数とメッセージ数

となり，ロックステッププロトコルを用いていることから，メッセージ数の合計は $2n(n-1)^2$ となる．つまり，現実的な RTS ゲームにおける本プロトコルの通信量のオーダは $O(n^3)$ となることが予想される．

k=1 とした場合の，シミュレーションプログラムの実行結果を表１と図４に示す．表１は，ピア数とそれに対応したメッセージ数のシミュレーション結果の一部であり，それをプロットした図に，３次式で最小二乗法に当てはめた曲線を図４に示す．この結果から，このアルゴリズムの計算量の検証を行うことができた．

5. おわりに

本研究では，オンラインゲームをピュア型 P2P ネットワークで行なう際の一方式を提案した．本方式により，データの遅延を利用した不正やデータ改ざんによる不正を防ぐことができる．また，ゲームを進行するグループの中に不正を行なうピアが存在していても，不正者を排除し，他のピアでゲームを継続することが可能となる．

今後の課題としては，ハッシュ関数とその不可逆性の検討や，メッセージ通信回数を減らすための効率的なプロトコルについての研究がある．また，不正を起こしたピアの識別やペナルティの与え方についても検討する必要があると思われる．

参考文献

[1] 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,情報処理学会, pp.79-84, 2003.

[2] 飯村卓司：オーバレイネットワークを用いた大規模マルチプレイヤオンラインゲームインフラストラクチャの構築，インターネットコンファレンス 2004(IC2004)論文集.

[3] P2P ゲームと分散コンピューティング，http://homepage3.nifty.com/toremoro/p2p/trend2.html

[4] N.E. Baughman and B.N. Levine, Cheat-proof playout for centralized and distributed online games, Proc. IEEE INFOCOM2001, pp.104-113,2001.

[5] E. Cronin, B. Filstrup and S. Jamin, Cheat-proofing dead reckoned multiplayer games, Proc. 2nd Int'l conference on ADCOG, 2003.

[6] C. GauthierDickey, D. Zappala ,V. Lo and J. Marr, Low latency and cheat-proof event ordering for peer-to-peer games, Proc. NOSSDAV'04, 2004.

[7] 米田友洋，梶原誠司，土屋達弘，ディペンダブルシステム，共立出版，2005.

[8] 南谷崇，フォールトトレラントコンピュータ，オーム社，1991.

[9] 亀田恒彦，山下雅史，分散アルゴリズム，近代科学社，1994.

[10] L. Lamport, R. Shostak and M. Pease, The Byzantine generals problem, ACM Trans. Program. Lang. Syst., 4, 3, pp.382-401, 1982.

[11] M. Pease, R. Shostak and L. Lamport, Reaching agreement in the presence of faults, JACM, 27, 2, pp.228-234, 1980.

[12] 前田幹夫，中村宏之，須永宏：個人創造コンテンツ流通活性化技術に関する検討，信学技報，NS2004-140, pp29-32, 2004.

[13] H. Yoshimoto, R. Shigetomi and H. Imai, How to protect peer-to-peer online games from cheats, Proc. IEEE CIG2005, 2005.