大規模ネットワークゲームのインフラを自律的に構築するシステムに関する考察

Notes on autonomous construction of large scale network game infrastructure

佐渡山 陽 [2], 小杉 隆二 [1], 河野 真治 [3]
Akira Sadoyama,Ryuji Kosugi, Shinji Kono

現在のネットワークゲームでは、セントラライズドサーバーを用いるのが一般的である。サーバを用いる方法では同時にゲームの中で相互に影響を与える人数には制限があるのが普通である。数百万人以上が参加できるようなネットワークゲームでは、全く別な仕組みが必要となると思われる。そこで、我々は、ゲーム端末と Internet 上で自律的にネットワークを作り協調動作する Agent を用いたフレームワークを提案している。人数的なスケーラビリティを持つネットワークゲームを実現するために、動的に変化する Agent 間のネットワークを実現する通信プロトコルを提案する。

1 はじめに

ここ数年で、ネットワークゲームを取り巻く環境は大きく変化した。PC 用ネットワークゲームの台頭、有名タイトルのネットワークゲームへの進出、小型ゲーム機の増加、そして、家庭用ゲーム機のネットワークへの対応などが挙げられる。また、インターネットの普及・発達と共に、ネットワークゲームの規模も増加している。これにより、数万人規模の多人数同時参加型ゲーム、MMOG(Massive Multiplayer Online Games) と呼ばれるジャンルも生まれた。

しかし、現在のネットワークゲームのシステムは、セントラライズサーバーを用いて管理を行うのが一般的であるため、参加者が増えるに従って様々な障害が起こりやすくなっている。そこで、我々はセントラライズサーバーが存在しないネットワーク構成で、百万人規模のネットワークゲームを実現する事を最終的な目標として、Agent を用いたネットワークゲーム用フレームワークを提案している。(図 1 参照)

これまで、我々は PlayStation 2・Agent 間の通信プロトコルに関する研究発表を行ってきた。[1][2][3][4] 今回は、次のステップとして、Agent・Agent 間の通信プロトコルについて提案を行う。

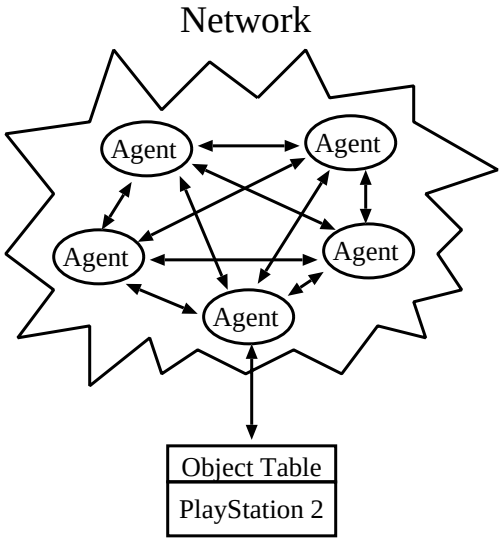

図 1: Agent

2 Agent とは

Agent とは、それ自身が情報処理能力を持ち、送受信されるデータを元に家庭用ゲーム機とネットワークに対して、様々な働きかけを行うものである。Agent 間でネットワークを築くことにより並列分散型のネットワークゲームを構築する要素となる。

Agent は、各家庭用ゲーム機の持つアーキテクチャの特性を吸収する。それにより、プラットフォームに依存しないゲームプログラミングを提供したり、ゲー

[1]琉球大学工学部情報工学科
kein@cr.ie.u-ryukyu.ac.jp
Infomation Engineering, faculty of engineering, University of Ryukyu

[2]琉球大学大学院理工学研究科
meso@cr.ie.u-ryukyu.ac.jp
Intelligent System Engineering, Graduate School of Engineering and Science,University of Ryukyu

[3]琉球大学 科学技術振興事業団さきがけ研究 21(機能と構成)
kono@ie.u-ryukyu.ac.jp
University of Ryukyu,PRESTO,Japan Science and Ethnology Corporation

ム機と市場 PC との性能差を緩和できる等の利点がある。実装には、Java を用いる。

3 ネットワークの形態

我々が目指すネットワークの持つ特徴は、以下の通りである。

- セントラライズサーバーを必要としない
- ゲームの必要に応じて動的かつ自律的にネットワークを再構築する
- Agent ネットワークがプログラムの流通も担当する

Agent が構築するネットワークは、最初にツリーの形態を取る。ノード位置を示すユニークな番号が各 Agent に与えられ、その番号を基に Agent 同士がコネクションを確立し、ツリーを構築していく。ツリーは各ゲーム毎に構築され (これをゲームツリーと呼ぶ)、一つの Agent が複数のゲームツリーのノードとなることもある。ただし、あるゲームツリーの変更が、他のゲームツリーに影響を与えることは無い。

ネットワーク・ゲームは、初期のツリー構造とは別に必要なトポロジーを持つネットワーク構造を自律的に構築する。構築は、実際の通信状況を計測して、データの通信量・通信速度が対象となるゲームに最適になるように行なわれる。

4 Agent の 3 形態

Agent は、通信する相手、あるいは、通信する相手の要求に適した形態で対応する。それにより、動的なネットワーク構築を可能にする。Agent の形態は全部で 3 つあり、以下にその詳細を示す。

4.1 Portal Agent

Portal Agent は、新しく接続要求を送って来た Agent の受け付け処理を行う。(図 2 参照) 接続要求には、以下の 3 種類がある。

1. 新しい Agent として、ネットワークへの接続要求
2. 特定のゲームネットワークツリーへの接続要求
3. 子もしくは親として、ツリー上のノードへの接続要求

1 の要求に対して Portal Agent は、所持しているゲームリストを渡し、初めにどのゲームツリーへ参加するのかを選択させる。参加ゲームが指定されると、実際にそのゲームを実行している Agent を検索して、そのアドレスを返す。接続要求を出した Agent は、そのアドレスへ直接ゲーム参加の申請を行う。(2 の要求を行う)

2 の要求に対して Portal Agent は、ゲームツリーの状態からゲームへ新規参加する Agent の参入位置を算出し、親ノードとなる Agent のアドレスを返す。接続要求を出したた Agent は、そのアドレスへ直接ノード接続要求を行う。(3 の要求を行う)

3 の要求に対して Portal Agent は、ノード接続の処理を行う。その処理とは、コネクションの確立・切断、ゲームツリーデータの更新・伝搬、である。ツリー構築後、新規参加した Agent は、必要ならば以下のデータを隣接するノードから取得する。

- ゲームツリーの状態を表すデータ
- ゲームリスト
- ゲームに必要なデータ

どの Agent からでもネットワークへ参加できるため、ネットワーク全体の構成を管理するセントラライズサーバーを必要としない。Portal Agent では、一つの Portal Agent に複数の要求が起こる可能性が考えられるため、1 対多 (Portal Agent 対 Agent) の通信となる。

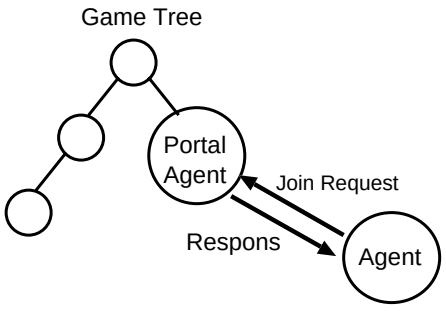

図 2: Portal Agent

4.2 Distributed Agent

Distributed Agent は、ゲーム進行に必要なデータのやり取りを行う。(図 3 参照) 他の Distributed Agent から送られて来たパケットを解析し、自分に

送られたデータなら適所に格納する。もしも、自分宛ではないなら、相手先を確認して適切な Distributed Agent へとパケットを送り、ルーティングを行う。よって、多対多 (Distributed Agent 対 Distributed Agent) の通信となる。

図 3: Distributed Agent

4.3 Front-end Agent

Front-end Agent は、Agent システムと家庭用ゲーム機のインターフェースとなる。(図 4 参照) プレイヤーの操作をゲームの状態へ反映したり、逆に、オブジェクトテーブルのフレームワークを用いて、ゲームの状態に応じた描画制御を行う。[1][2]

1 つの Front-end Agent に対して、複数のゲーム機が接続可能である。よって、1 対多 (Front-end Agent 対ゲーム機) の通信となる。

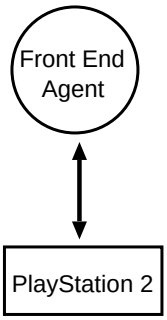

図 4: Front-end Agent

5 システムの流れ

家庭用ゲーム機の始動から、ゲーム終了までの簡単な流れを以下に示す。

1. 家庭用ゲーム機が Front-end Agent へアクセス
2. Front-end Agent が Portal Agent へアクセス
3. Front-end Agent がゲームのリストを受け取りプレイヤーへ提示
4. プレイヤーが遊ぶゲームを選択
5. Front-end Agent が Portal Agent を介して、選択されたゲームを実行している Agent を検索
6. 検索にヒットした Agent へ Front-end Agent が接続要求を行う
7. Portal Agent がツリーの状態から、新たに参加する Agent の親ノードを決定
8. Front-end Agent が、指定された親へ接続要求を行う
9. 必要に応じて、Front-end Agent が、各種必要なデータを隣接ノードから取得
10. ゲームスタート
11. 状況に応じて、Agent はゲームツリーの構成を変化させる
12. ゲームが終了したらツリーから抜ける

6 データフォーマット

ゲームに必要なデータの送受信には、以下のフォーマットを使用する。

- ゲームツリーの状態 … XML フォーマット
- 制御コマンド … XML フォーマット
- 各種パラメータ … XML フォーマット
- 3D オブジェクト … XML フォーマット
- 2D イメージ … png フォーマット
- binary File … base64 フォーマット

7 ゲームの例題

Agent システムを利用したネットワークゲームの例として、「投票ゲーム」を挙げる。このゲームでは、ツリー構造のネットワーク・トポロジーを採用する。投票ゲームとは、単純にゲーム参加者が「Yes」か「No」を選択し、送信するというゲームである。投票の対象となる質問は、ゲーム端末あるいは Agent から投入され、ツリーの根に集められ、そこで質問の選択と番号づけが行なわれる。質問は、Distributed Agent を通して、ゲーム端末にマルチキャストされる。ゲーム端末上でユーザから時間内に回答を得る

と、ゲーム端末は回答情報を Front-end Agent に送信する。データを受け取った Front-end Agent は、親ノードの Distribued Agent に回答情報を送信する。データを受け取った親ノードは、その子ノードからの情報を集計した単一の情報を親ノードに送信する。最終的にツリーのルートに全ての情報が集められ、そこで解答情報を集計し、ルートからまた各子ノードへと集計結果を送信する。各 Agent は、ゲーム参加者に集計結果を送信する。(図 5 参照)

このゲームにおいて、Agent 間でやり取りされるゲームパラメータデータは、以下の通りである。

- 質問の番号
- Yes の数
- No の数
- 集計結果

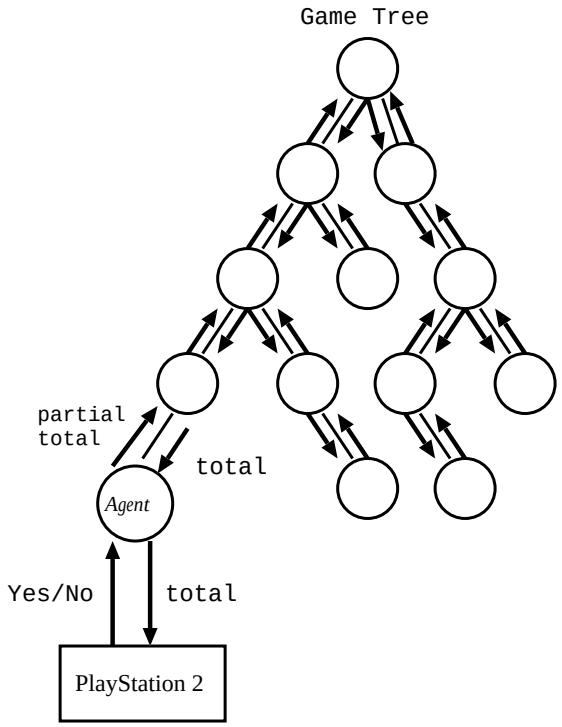

図 5: 投票ゲームにおけるデータの流れ

この方法では、投票時間制限はゲーム端末上で有効であるが、それが有効であるかどうかは、集計に間に合うかどうかによって決まる。この非決定性があるために、正確さを犠牲にしたユーザ数のスケールあっぷが可能となる。また、集計された結果のみを上位に伝えるためにツリーの上位にいくにつれトラフィックが増えることはない。

8 まとめ

本研究では、動的に変化する Agent 間のネットワークにおいて、トラフィックを抑えつつ、ネットワークゲームを実現するための通信プロトコルの提案を行った。今後の課題として、以下のものが挙げられる。

- ノード最適化に関する更に細かい仕様の設計
- Agent の実装
- 実際に運営した際のデータ計測 (もしくはシミュレーション)
- 計測したデータによる評価
- 最大子ノードの数等の各種パラメータのチューニング
- 更に効率の良いネットワーク構築のプロトコル開発

参考文献

[1] 河野 真治, 佐渡山 陽：PlayStation 2Linux 上のネットワークゲーム・フレームワークの提案, 日本ソフトウェア科学会第 19 回大会論文集 September, 2002

[2] 河野 真治, 佐渡山 陽：PlayStation 2 Linux におけるネットワークゲーム用フレームワークの実装, 第 93 回 OS 研究発表会 May, 2003

[3] 河野 真治, 村吉 政登：PS2 向きの並列ゲームオブジェクトシステムの提案, SwoPP 2001, July, 2001

[4] 河野 真治, 村吉 政登：VRML と他言語を使用した PlayStation のゲーム開発システムの提案, 日本ソフトウェア科学会第 17 回大会論文集 September, 2000

[5] P. パチェコ (著), 秋葉 博 (訳)：MPI 並列プログラミング, ISBN 4-563-01544-X

[6] Java で学ぶアルゴリズムとデータ構造：Robert Lafore(著), 岩谷 宏 (訳), ISBN 4-7973-0694-7