JAVA のソケットと JXTA を用いた大規模ネットワークゲーム AgentSystem の比較と評価

小 杉 隆 二[†]　河 野 真 治[††]

我々は数百万人規模のネットワークゲームを実現するために、Agent を用いたネットワークゲーム用インフラを提案している。ここでは、ネットワークトポロジーとして Tree 構造を持つゲームを、提案しているインフラ上に作成し、JAVA のソケットを用いた Agent と JXTA を用いた Agent を実装し、通信速度の比較、評価を行う。

Comparison and evaluation of the large-scale network game AgentSystem using the socket of JAVA and JXTA

Ryuji Kosugi[†] and Shinji Kono[††]

We have proposed the infrastructure for network games which used Agent, in order to realize the network game of millions of people scale. In this paper, a game which has Tree structure as a network topology is created on the proposed infrastructure, Agent using the socket of JAVA and Agent using JXTA are mounted, and comparison and evaluation of transmission speed are performed.

1. はじめに

近年、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も世界的規模で増加している。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。

これらのゲームは同時ログイン数は数万人でも同時にインタラクションするのは数人から数十人である。また、現在のネットワークゲームのシステムは、集中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。例えば、数万人が同時にアクセスすることにより集中サーバーへの負荷が大きくなり、重くなることや、集中サーバーが落ちるとゲームも全て止まってしまうことなどが挙げられる。また、開発されているゲーム自身も、特定のゲームマシンに依存して作られており、プレイヤーはソフトウェアが要求するハードウェアを購入しなければならない。

本研究室では、集中サーバーが存在しないネットワーク構成で、数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。

本論文では、現在実装されている Agent システムを、P2P フレームワーク技術である JXTA を用いて、投票ゲーム上に実装してシミュレーションを行い、データ通信においての比較・評価を行った。

2. Agent システムの概要

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。(図 1 参照) 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、ゲーム機と市場 PC との性能差を緩和できる等の利点がある。

我々が目指している Agent を用いたネットワーク

† 琉球大学理工学研究科情報工学専攻
Interdisciplinary Infomation Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†† 琉球大学工学部情報工学科
Information Engineering, University of the Ryukyus.

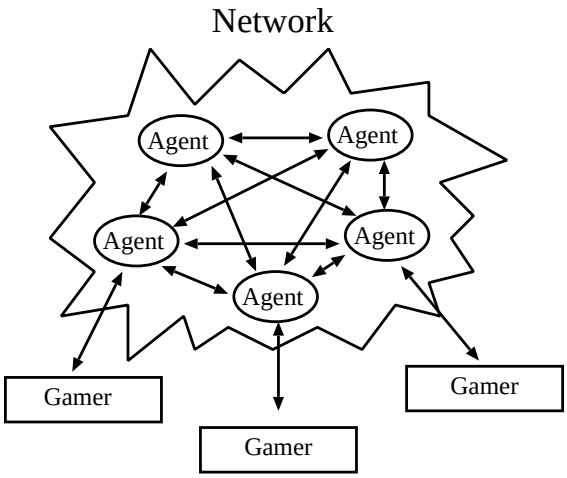

図 1　Agent を用いた通信システム

ゲームとは、まず、数百万人規模のネットワークゲームで、かつ、プレイヤーがゲームを常に監視する必要が無いゲームである。現存するネットワークゲームは、たいてい常にゲーム機となるマシンの前で、そのゲームを操作し進行状況を見ていなければならない。よって、かなりの時間が必要とされる。時間がない人には、そういったゲームを続けていくことは難しい。しかし、我々が目指すゲームは、ゲームの進行方法についての指示を設定しておけば、あとは Agent が指示通りにゲームを進めてくれるというものである。その間、ゲームを常に監視する必要はなく、時間が空いた時にだけゲームの様子を見ればよい。

また、分散 Agent によるネットワークゲームインフラを構築し、そのインフラ上にはゲームが複数個干渉しながら存在するものを目指している。

これらを実現するためには、現状のインフラには頼れず、新しいシステムが必要となった。そこで以下のようなインフラを提案している。

- 集中サーバーを必要としない。
  集中サーバーを用いたネットワークゲームの問題点は Section1 で既に述べた。数百万人規模のゲームを実現するには、分散 Agent によるネットワークゲームインフラの構築が必要となる。
- 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。
  同じインフラ上に複数ゲームが存在する場合、各ゲームに最適なネットワークを構築していかなければならない。また、ネットワークトラフィックなどからトラフィックを抑えるために再構築をしていく必要がある。
- Agent ネットワークがプログラムの流通も担当。
  これにより、User はゲームをするために、必要なハードウェアやソフトウェアを準備する必要がない。

2.1　Agent のネットワーク構成

Agent が構成するネットワークは物理的なものと論理的なものの 2 種類ある。Agent 起動時に与えられるユニークなノード ID を基に Agent 同士がコネクションを確立し、作られる物理ネットワーク。もう一つは、物理ネットワーク上に仮想的に表現した論理ネットワークがある。(図 2 参照) 実際にゲームの通信は、論理ネットワークを基に行われる。

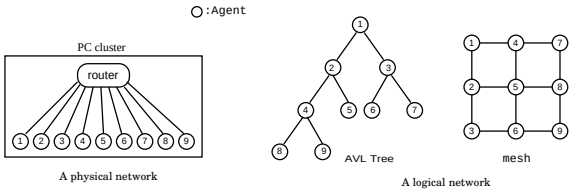

図 2　物理ネットワーク (左) と論理ネットワーク (右)

2.2　Agent が持つ 3 形態

Agent は通信する相手、また通信相手の要求に応じて以下の 3 つの形態を持つ。

(1)　「Portal Agent」･･･ 新規 Agent の受け付け処理を行う

(2)　「Distributed Agent」･･･ ゲーム進行に必要なデータのやり取りを行う

(3)　「Front-end Agent」･･･ Agent とゲーム機のインターフェース部分となる

3.　JXTA の概念

3.1　P2P とは

P2P(Peer to Peer) とは、サーバ・クライアント方式のようにそれぞれに役割があるネットワーク形態とは違い、接続されたコンピュータ間に上下関係が存在しないネットワークの形態のことを指し、つまりサーバとクライアントの区別がなく、全てのコンピュータがサーバとしてもクライアントとしても機能する。

P2P には、大きく分けて「ハイブリッド P2P」と「ピュア P2P」の 2 種類に分けられる。

- ハイブリッド P2P
  各ピア同士は Peer to peer な接続だが、全てのピアの情報を統括しているサーバーが存在する。ピアはサーバーの情報を元に別のピアへの接続や通信を行っている。
- ピュア P2P
  サーバーを一切必要としない純粋な P2P システムで、各ピアがそれぞれ接続している情報を保持している。

本来、P2P は小規模な LAN でファイルやプリンタを接続するのに向いている。Napstar、Gunutella などのようなファイル交換システムのように、インターネット上でユーザ同士のファイルを直接やり取りする技術も現れており、最近では P2P 技術による大規模ネットワークの研究に注目が集まっている。

3.2 JXTA

JXTA とは、サン・マイクロシステムズが提供する P2P のフレームワーク技術で、この JXTA を用いることによって、P2P のアプリケーションなどが容易に作成できるという技術である。JXTA では、ピアの発見、サービスの公開、他のピアとのデータ交換、ピアを集団化してグループを作成等といった機能が提供されている。JXTA を用いた通信の流れとしては、各ピアがピア固有の識別情報、サービスやパイプの設定情報を公開するアドバタイズメントを行う。次に他のピアはその公開されたアドバタイズメントを検索・発見し、そのインスタンスの中の設定情報を元に通信が行われる。

3.2.1 JXTA Protocols

JXTA は 6 つのプロトコルから構成されており、TCP/IP などのトランスポートプロトコル上に実装されることができる。

以下に 6 つのプロトコルを述べ、プロトコル階層を図 3 を示す。

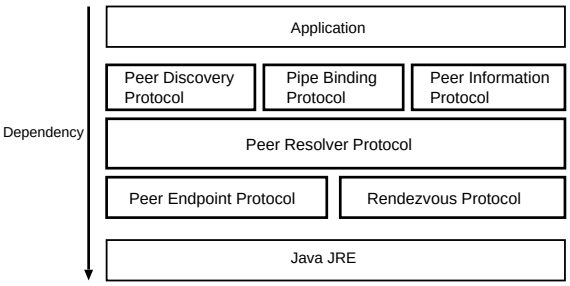

図 3 JXTA のプロトコル階層図

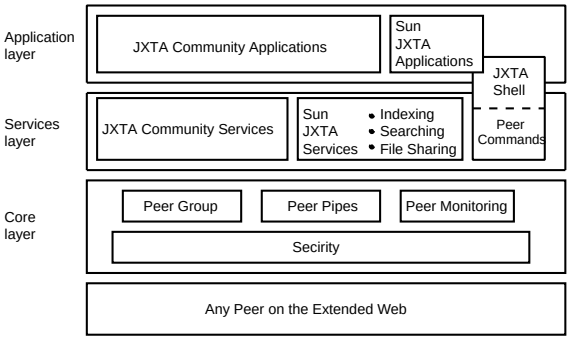

図 4 JXTA の Architecture 図

- Peer Resolver Protocol
  ピアやピア・グループ、パイプ、その他の情報を検索するためのクエリを送受信するためのプロトコル
- Peer Discovery Protocol
  ピアやピア・グループ、告知を発見するためのプロトコル
- Peer Information Protocol
  他のピアの能力や状態を知るためのプロトコル
- Pipe Binding Protocol
  パイプ告知をパイプの終端に特定するためのプロトコル
- Peer Endpoint Protocol
  あるピアから特定のピアまでのルートを発見するためのプロトコル
- Rendezvous Protocol
  ネットワーク上にメッセージを伝播するためのプロトコル

3.2.2 JXTA の基本アーキテクチャ

JXTA のアーキテクチャは Application layer,Services layer,Core layer の 3 つのレイヤーから構成されている。

以下に 3 つのレイヤーの役割を述べ、アーキテクチャ図を図 4 に示す。

- Application layer
  ファイル共有，リソース共有，分散ストレージなどといった複数の統合されたアプリケーションから構成されている。
- Services layer
  P2P アプリケーションに必要なネットワークサービスを提供している。

- Core layer
  JXTA の核となる 4 つの要素で構成されている。あるピアが他のピアを検索する場合に詳細な情報を記述するための「Peer Group」、ピアまたはピアグループが相互に通信する際に利用される通信メカニズムである「Peer Pipe」、ピアの接続状況やコンテンツへのアクセス状況を把握して、その情報をアプリケーションやサービスに通知するための「Peer Monitoring」、特定のセキュリティポリシーを定める代わりに、セキュリティ機構をプラグインするためのセキュリティフレームである「Security 機能」。以上の 4 つである。

4. JXTA での具体的な実装

今回実装した Agent システムの流れは図 6 のようになっている。

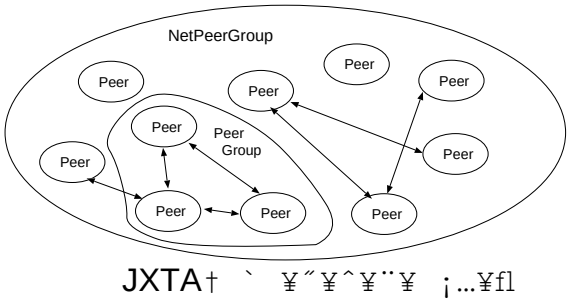

図 5 JXTA のネットワーク概念図

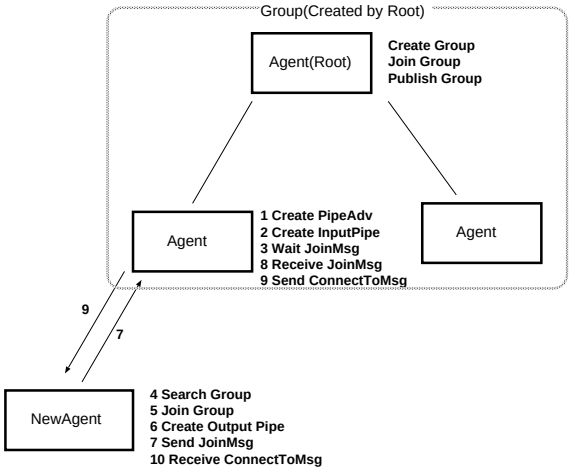

図 6 実装した Agent システムの流れ

まず最初の Agent がピアグループを作成し、自分自身をそこに参加させる。そして、そのピアグループを JXTA ネットワークに公開し、パイプアドバタイズメントを行う。新規 Agent は、Agent システムのピアグループを検索し、見つけたらそれに参加する。公開されているパイプサービスを取得し、入出力用のパイプを生成し、接続のメッセージを送信する。すでに接続されている Agent に新規 Agent から接続メッセージが届いたら、どこが接続先として適しているか判断し、接続可能な Agent を新規 Agent に通知する。新規 Agent はそのメッセージに従って接続を行う。

そして、論理ネットワークを基に投票データのルーティングを行う。つまり、ここでいう Agent は Portal Agent と Distributed Agent の 2 つの役割を指している。

実際のソースコードを次のページで紹介する。

5. 投票ゲーム

従来の Agent システムが投票ゲーム用に実装されているので、比較・検証を行う為に投票ゲーム用に実装した。

投票ゲームとは、User がある質問に対して「Yes」か「No」を投票するゲームである。質問も各 User が投入することができ、通常 1 つの質問に対して、1 回の投票を行うことができる。

投票した結果は、ツリーに沿って全ての Agent に反映される。(図 7 参照)

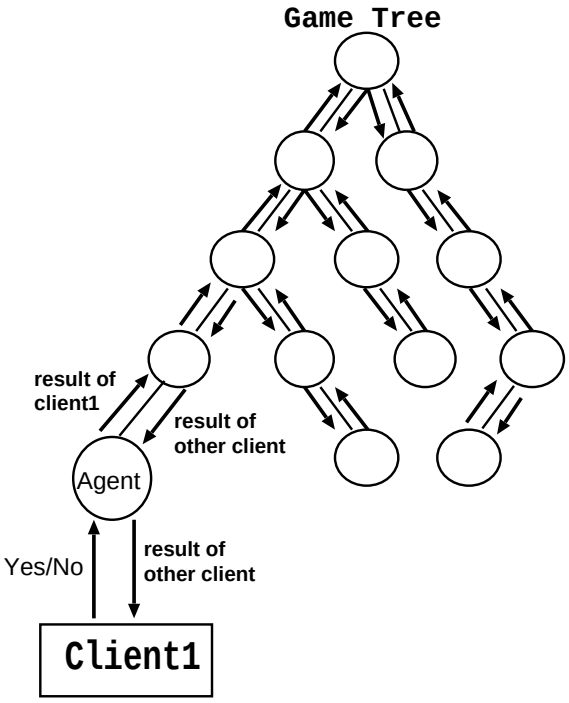

図 7 投票全体の流れ

5.1 Balanced Binary Tree

今回、投票ゲームを作成するにあたってネットワークトポロジーとして Balanced Binary Tree 構造を用いた。Balanced Binary Tree とは全ての Agent がツリーの深さの平均を保持しており、各 Agent は自分の深さと平均を比較し、平均より浅い Agent に新規 Agent を追加していくことにより、偏った Tree が構築されるのを避ける仕組みになっており、常にある程度バランスを保った Tree を維持することができる。

```
PeerGroupManager pgm = new PeerGroupManager(netPeerGroup, discoSvc);

/* 最初に起動した Root となる Agent はピアグループを作成  */
/* 2 番目以降に起動した Agent は既存のグループを検索      */
if (str.equalsIgnoreCase("root")) {
    newGroup = pgm.createGroup(groupName, "JXTA Agent System group");
} else {
    newGroup = pgm.searchGroup(groupName);
}
pgm.joinGroup(newGroup);
/* パイプサービスの取得 */
pipes = newGroup.getPipeService();
--------------------------------------------------------------------------

/*パイプアドバタイズメントの作成*/
PipeAdvertisement pipeAdv = (PipeAdvertisement)AdvertisementFactory
                  .newAdvertisement(PipeAdvertisement.getAdvertisementType());

/*パイプサービスの各種パラメータの設定*/
pipeAdv.setName(peerGroup.getPeerName());
pipeAdv.setPipeID((ID)IDFactory.newPipeID(peerGroup.getPeerGroupID()));
pipeAdv.setType(PipeService.UnicastType);
--------------------------------------------------------------------------

AgentListener[] agentListener = new AgentListener[3];

/*出力用パイプの生成してメッセージを待つ*/
agentListener[i] = new AgentListener(i, inPipeAdv[i]);
pipeIn = pipes.createInputPipe(inPipeAdv, agentListener);
/*入力用パイプ生成*/
pipeOut = pipes.createOutputPipe(outPipeAdv, 10000);

/* パイプを通してメッセージの送信 */
pipeOut.send(ConnectMessage);
--------------------------------------------------------------------------

/* セットしておいたリスナーがイベントを通知し
                  メッセージが来た時のイベント処理 */

try {
    /* メッセージを取り出す */
    msg = event.getMessage();
    if (msg == null) {
        System.out.println("msg is null");
        return;
    }
} catch (Exception e) {
    e.printStackTrace();
    return;
}
```

6. シミュレーションによる通信量計測

従来の Agent システムと Jxta を用いた Agent システムを比較するため、以下の条件で計測を行った。

- Agent 数 ：PC クラスタ 1 台あたり Agent1 台で合計 31 台
- Client 数：学科 PC1 台あたり Client1 台で合計 32 台
- 投票数 ：1.5 秒に一回ずつ各 Client から投票が行われる

シミュレーション時の投票データの流れ (図 8 参照) を以下に述べる。

1 投票の対象となる問題が、エミュレータから投入される
2 データを受け取った Agent は問題をハッシュテーブルに格納する
3 同時に、Agent は直接接続している Agent にデータをマルチキャストする。最終的に全ての Agent に反映される
4 投票エミュレータは Agent が保持している問題をとってきて、質問に対して「Yes」か「No」を選択し、Agent に投票データを送信する
5 投票データを受け取った Agent は、またそれをハッシュテーブルに格納する
6 3 と同様にマルチキャストし、Agent に反映させる

Agent が保持しているハッシュテーブルには、質問、Yes の総数、No の総数が格納されている。

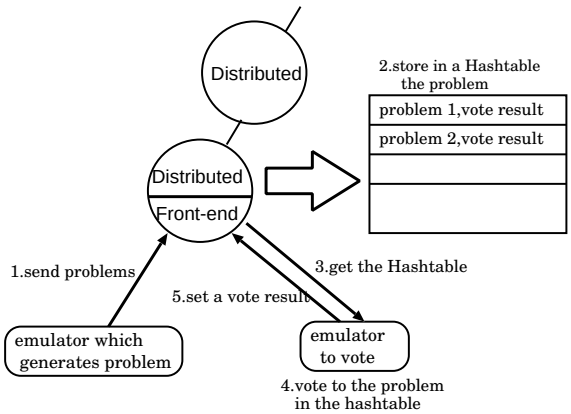

図 8 投票データの流れ

6.1 計測結果の比較・考察

JAVA のソケットを用いた Agent システム (図 9 参照) と JXTA を用いた Agent システム (図 10 参照) の結果を以下に示す。

各 Agent(X) が受け取った総メッセージサイズ (Z) の時間変化 (Y) を測定した。

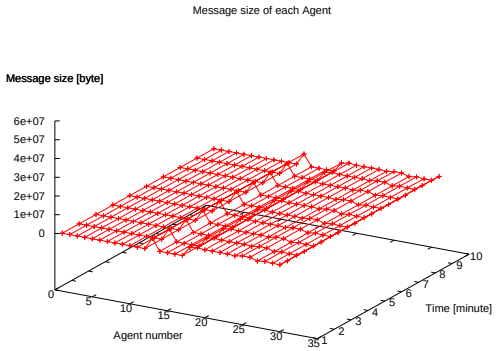

図 9 JAVA ソケットを用いた Agent システム

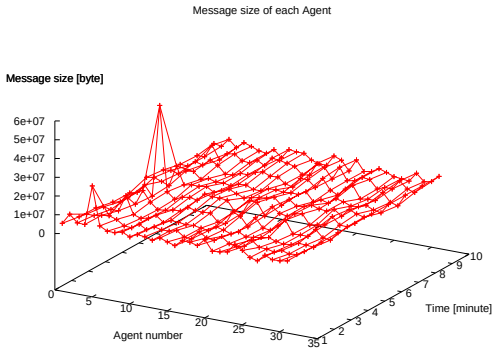

図 10 JXTA を用いた Agent システム

JXTA の通信の一部にピークがあるもののそれ以外の時間変化はほとんどない

どちらも Agent 毎に通信量の多少はあるものの全体的に安定している

各 Agent の平均通信量には、かなりの差が生じた。

JXTA の通信量の方がかなり多い

6.2 総 合 評 価

○ JXTA の利点

接続先の IP アドレスとポート番号を知っておく必要がない。JAVA のソケットプログラミングでは必ず必要である。

○ JXTA の欠点

グループ作成、パイプの作成、アドバタイズメントの公開・検索などソースコードの量が増える。

バージョン 1.0 と 2.0 では仕様が大きく変わっており、今後バージョンが上がった時の移植には大幅な修正が必要と考えられる。また、現在も開発中の技術なので情報が少ない。

このように JAVA のソケットを用いた場合と JXTA を用いた場合の両方ともコーディングをする上では一長一短であるが、JXTA の欠点でも述べたように JXTA はこれからも次々と改良されていくため、プログラムを組む上での情報が少ないため、コーディングに時間がかかる。

7. まとめと今後の課題

7.1 ま と め

本研究では、JXTA を用いた並列分散型のネットワーク構築システムを提案し、Agent システムを実装した。また、JAVA のソケットと JXTA それぞれで実装した Agent システムを、投票ゲーム上に作成しシミュレーションを行うことによって比較・評価を行った。これにより、前セクションの考察で述べたように JXTA の利点・欠点がわかった。

さらに、通信量の比較を行うことによって JXTA の方に大きなトラフィックが生じた。しかし、時間当たりのメッセージ処理率を測っていないため、その違いの可能性も考えられる。現段階ではソースコードの量や通信量の点においては、JAVA のソケットを用いた方が有効であることがわかった。

今回実装した Agent システムは、Tree 構造を用いながらもリアルタイム性を考慮し、メッセージを受け取った Agent は即座に他の Agent にマルチキャストする仕様だった。しかし、Tree を上がって行くに従って投票結果を集計していく設計にして実験を行うことによって違った結果が得られるかもしれない。

7.2 今後の課題

今後の課題として以下のことが挙げられる。

- メッセージが Tree の上に行く従って、集計していくような設計と実装
- その際の各ノードにおけるパケット数の計測
- Agent 台数の増加に伴った通信量の変化の計測

また、Agent システムとは数百万人規模を目的としており、現在の学科の PC クラスタ 50 台で百万台の Agent を起動するためにはクラスタ１台あたり２万台の Agent を起動する必要がある。単純に考えてプロセス２万個起動するにはクラスタの性能により限界がある。現在クラスタ１台あたり１００プロセス (１００ Agent) を起動し、最低でも５０００台以上の Agent を起動することを目標としている。JAVA の Thread

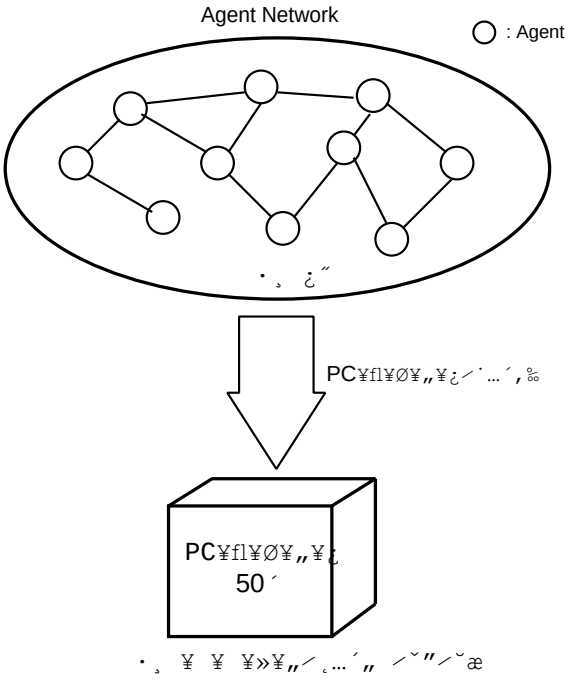

図 11 Agent 百万台の実行環境の問題

を用いて１プロセスあたり１０ Thread 起動することによって１台あたり１０プロセスに抑えることも考えられる。

参 考 文 献

1) 佐渡山 陽, 河野真治. 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) 佐渡山 陽, 小杉 隆二, 河野 真治. 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. 日本ソフトウェア科学会第 19 回大会論文集 September, 2003
3) 小杉 隆二, 河野 真治. Tree 構造と Mesh 構造に対応した大規模ネットワークゲーム AgentSystem を用いたシミュレーション. 日本ソフトウェア科学会第 20 回大会論文集 September, 2004
4) http://www.jxta.org/ Project JXTA
5) Joseph D.Gradecki(著) Mastering JXTA, ISBN 0-47L-25084-9
6) Daniel Brookshier,Darren Govoni,Navaneeth Krishnan(著) JXTA:Java P2P Programming, ISBN 0-672-32366-4