2003－OS－93 (16)
2003／5／9

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

佐渡山 陽[†] 河野 真治[☆,††]

我々は、百万人規模のネットワークゲームを実現するためのフレームワークを提案している。汎用機上で Agent がゲームを運用し、PlayStation 2 上では Agent からの指示によって描画処理のみを行うというものである。ここでは、Agent と PlayStation 2 間における通信用のプロトコルを実装し、これまで使用していた Linda サーバーとの比較を行い、その有効性を示す。

Implementing of the frame work for online network games in PlayStation 2® Linux

Akira Sadoyama[†] and Shinji Kono[☆,††]

We have proposed the frame work for realizing the online network game of millions people scale. Agent operate a game on same high spec PC, and the directions from Agent perform only drawing processing on PlayStation 2. We implement the protocol for communication between Agent and PlayStation 2. Then compare with the Linda server which was using it until now, and show the validity.

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、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。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考案した。

† 琉球大学理工学研究科情報工学専攻
Interdisciplinary Information 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.
†† 科学技術振興事業団さきがけ研究 21(機能と構成)
PRESTO, Japan Science and Technology Corporation.

2. Agent

2.1 Agent とは

コンピュータ技術は進歩が早く、数年でマシンの性能に 10 倍以上の差が出てくる。これはゲーム機についても同様である。また、グラフィックの処理は比較的重いものである上、画面の垂直同期のタイミングに合わせて、毎回 1/60 秒以内に描画処理を終えなければならないという制限もある。PlayStation 2 は高い演算処理能力を有しているが、通信に関わる全ての処理まで同時に行うには力不足である。

そこで、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 PlayStation 2 とネットワークに対して、様々な働きかけを行う Agent を用意してやる。エージェントを使用することで、PlayStation 2 側のリソースの節約と、広域ネットワーク側のデータの送受信の反応速度を上げることが可能となる。また、PlayStation 2 がネットワークから切断されている場合もエージェントが代わりにネットワークゲームを進行することができるようになる。エージェントは、PlayStaion 2 の本体とは別にハードを用意してやって、その上で動かす。また、エージェント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)

2.2 Agent の運営・管理

Agent には、以下の二種類が考えられる。

- ユーザーの管理する汎用機上で運営される Agent
- プロバイダが管理する汎用機上で運営される

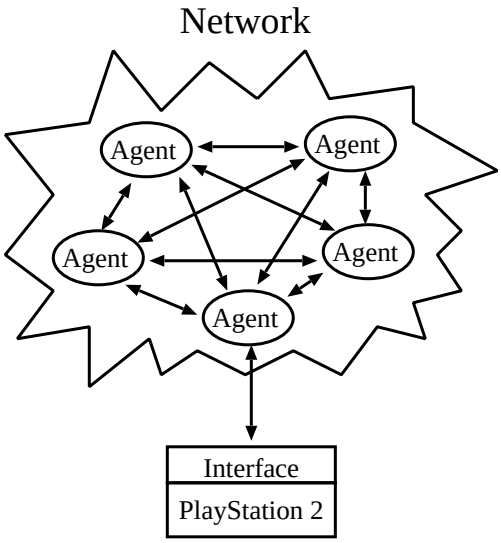

図 1 Agent を用いた通信システム

Agent

ユーザーが管理する汎用機上の Agent 同士がネットワークゲームを進行するとなった場合、各ユーザーが使用している回線のスピードの差が、直接ゲーム進行に影響を与えるため、大きな問題である。回線速度の差は、現在のネットワークゲームにおける問題点の一つでもある。また、ユーザーの手元にゲームプログラムが存在しているため、勝手に改造されて正しくスムーズなゲームの運営ができなくなる恐れもある。

プロバイダが管理する汎用機上の Agent 同士の場合、Agent 間の通信速度に差が無くなるので、ユーザーが使用している回線のスピードの差を、ある程度 Agent が吸収できる。また、ゲームプログラムがプロバイダ上で動いていることにより、ユーザーによるプログラムの改造を防ぐことができる。

よって、Agent の運営は各プロバイダ上で行われるのが良いと考えられる。また、様々なネットワークゲームが、統一のプロトコルのもとに Agent 上で運営される場合、資源の共有が行える。ゲームに必要な処理の一部は、ゲームが違っても演算自体は似たことをしているので、PC クラスターによって共通する処理を高速に行うシステムを構築することも可能である。(図 2 参照)

3. Agent・PlayStation 2 間のプロトコル設計

3.1 ゲーム進行に必要な情報

ネットワークゲームにおいて、ゲームを進行させる上で最低限必要となる通信内容は、以下の物であると考えられる。

- 各種デバイスからのプレイヤーの入力

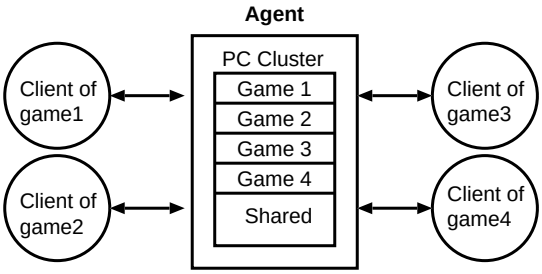

図 2 Agent における資源の共有

- オブジェクトの状態
- ゲームの状態
- file 転送

オブジェクトの状態とは、各オブジェクトが持っている座標・各座標軸に対する回転角・移動量・移動速度等のオブジェクトが持っている各パラメーターである。

ゲームの状態とは、ゲームの秩序を構成する様々な事象を決定づけているパラメーター群、例えば、時間・プレイヤーの数・勝敗・点数・重力値等である。

PlayStation 2 は、Core CPU と 2 つの Vector Unit で並列にジオメトリ処理を行うことで、高度な物理シミュレーションが実現できる。上記の情報さえ揃っていれば、シミュレーションによってゲームを進行させることができる。表現されるオブジェクトの動きは全てシミュレーションの結果であり、オブジェクトの操作とは、シミュレーションのパラメータを操作することにほかならない。

3.2 オブジェクトテーブル

Agent・PlayStation 2 間の通信プロトコルのインターフェースが、オブジェクトテーブルである。まず、ゲームで使用する全てオブジェクトを、特定の意味を持つグループに分ける。例えば、人間のプレイヤーならば、人の形を構成するパーツ (頭・体・手...etc) の構造体データを一つのまとまりとして、ツリー状にデータを構築しておく。(図 3 参照) また、手の中でも 5 本の指、といった様に各パーツ毎の更に細かい部分でのツリーも、必要ならば予め形成しておく。作成したオブジェクトのツリー毎に個別の ID をつけ、その ID を基にリストの先頭のアドレスをハッシュテーブルにしたものが、オブジェクトテーブルである。(図 4 参照) オブジェクトだけではなく、通信によって変更する必要があるパラメーターの構造体なども、登録しておくことができる。

Agent から送られてくるオブジェクトのデータに、オブジェクトテーブルやツリーを参照するための情報をヘッダーに含める事で、Agent から受け取ったデータを速やかに反映することができる。

3.3 パケットの形式

プロトコルに用いるパケットの形式は、ヘッダー+実データの形を取る。ヘッダーとして必要な情報は、

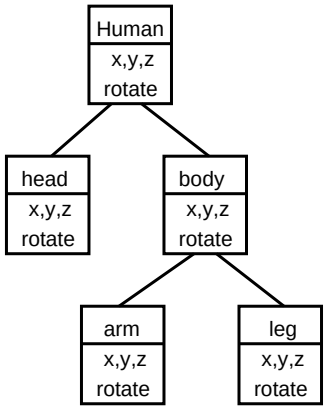

図 3 オブジェクトのツリー

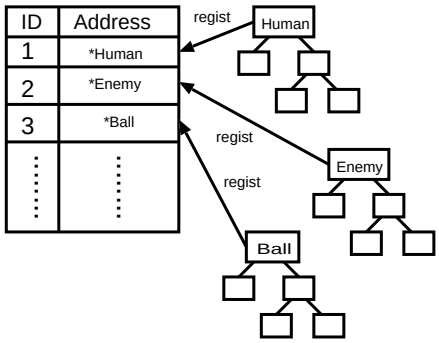

図 4 オブジェクトテーブル

以下の通である。

- パケットが持っているデータの種類
- 参照するオブジェクトテーブルの ID
- オブジェクトのツリーの offset 値
- オブジェクトデータの長さ
- オブジェクトデータ

パケットが持つデータは、単なるオブジェクトのデータだけでなく、Agent と PlayStation 2 がお互いに対して制御・同期を行うための情報を送る場合もある。その判別をつけるために先頭にデータの種類を示す。

オブジェクトのツリーの offset 値とは、オブジェクトテーブルによって参照されたツリーの中で、葉を特定するための値である。

通信を行う際、オブジェクトの状態を表す構造体ごと送るのも、構造体のフィールド 1 つ送るのも、通信状態は変わらない。よって、オブジェクトのツリーを構成している葉の持つ構造体のデータが、1 回の通信における最小の通信量となる。

4. 評価実験に用いたゲーム

4.1 サッカーサーバー

オブジェクトテーブルを用いた描画システムの評価

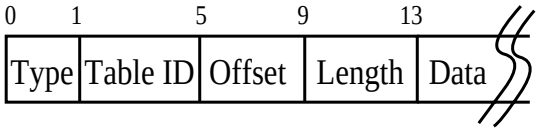

図 5 パケットの形式

を行うために、これをサッカーサーバーに対して実装した。サッカーサーバとは、ロボカップで使用される、シミュレータである。サッカーサーバー上でシミュレーション (ゲーム) が動いており、そこへ複数のクライアントが、走る・蹴る等のコマンドを送り、各プレイヤーを操作してゲームを進める。

本来、クライアントがどのようなコマンドを送るかは AI によって決定されるが、その代わりに PlayStation 2 から操作できるように実装を行った。ただし、サッカーサーバー・クライアント間の通信には、サッカーサーバー独自のものが用いられており、我々が実装したものとは異なる。ゲームの流れは、以下の通りである。

(1) サッカーサーバーから送られて来た各プレイヤーとボールの座標データをもとにして、PlayStation 2 上で 3D のフィールドがモニタに描画される。
(2) 人間がモニタを見てパッドを操作すると、その内容がクライアントへ送られる。
(3) クライアント上でパッド操作のデータがサッカーサーバー用のコマンドへ変換され、サッカーサーバーへ送られる。
(4) クライアントからのコマンドをサッカーサーバーが実行し、その結果をシミュレートする。その結果得られた座標のデータを PlayStation 2 へ送る。
(5) 1〜4 のくり返し。

4.2 オブジェクトテーブルによる実装

この実装では、サッカーサーバーのクライアントは、各 PlayStation 2 と 1 対 1 で通信を行う。そのため、PlayStation 2 の数だけ、クライアントが必要となる。オブジェクトテーブルによる実装では、サッカーサーバーとクライアントの二つを合わせて、Agent と見なすことができる。(図 6 参照)

4.3 Linda による実装

Agent を用いたシステムとの比較題材として、Linda サーバーを用いた。Linda とは、サーバーにタプルと呼ばれる ID と Data がセットになったものを、各クライアントが読み書きすることによって通信を行うシステムである。(図 7 参照) サーバーに蓄えられたデータへのアクセスは、タプルの ID を指定することによって行われる。つまり、セントラライズサーバーの例である。

Linda サーバーは、サッカーサーバー・クライアン

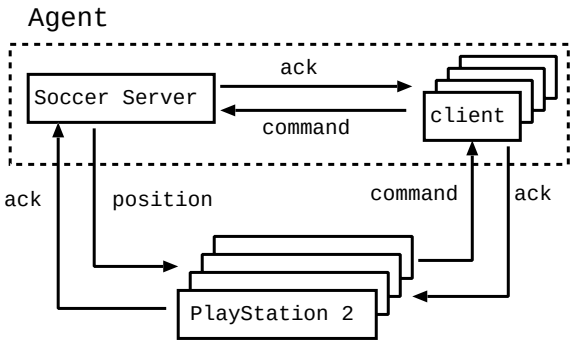

図 6 オブジェクトテーブルによる実装

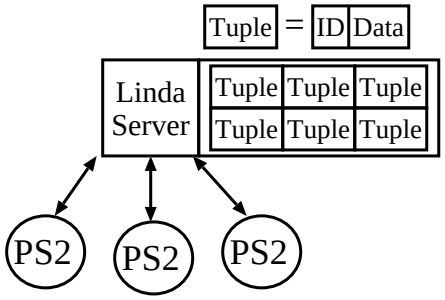

図 7 Linda サーバー

ト・PlayStation 2 の三者の間に立ち、通信の媒体となる。(図 8 参照) つまり、サッカーサーバーとクライアントは、実際のパケットのやり取りは、Linda とだけ行えば良い事になる。そのため、ここではクライアントが 1 つですんでいる。

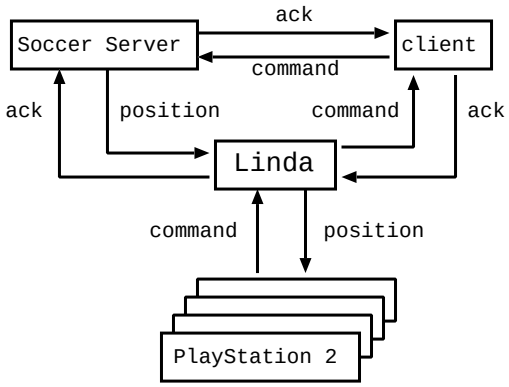

図 8 Linda サーバーによる実装

5. 計測の結果

上記にある二通りの実装において、ターンアラウンドタイムとパケット数を計測した。

5.1 ターンアラウンドタイム

ここで述べるターンアラウンドタイムとは、ゲームを行っている間に、PlayStation 2 と Agent 間、PlayStation 2 と Linda サーバー間、それぞれの通信において、一回につき、どの程度の時間がかかるのか、ということである。

図 9 に、オブジェクトテーブルによる実装における計測結果を示す。PS2 は PlayStation 2 の数、times は通信回数、sec はターンアラウンドタイム (秒) を表す。

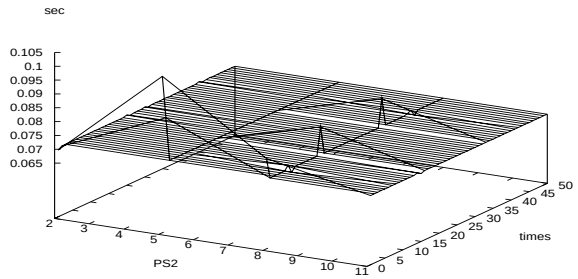

図 9 Agent・PlayStation 2 間のターンアラウンドタイム

図 10 に、Linda による実装における計測結果を示す。PS2 は PlayStation 2 の数、times は通信回数、sec はターンアラウンドタイム (秒) を表す。

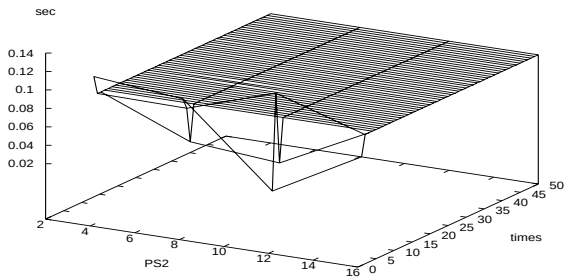

図 10 Linda・PlayStation 2 間のターンアラウンドタイム

5.2 パケット数

ここで述べるパケット数とは、上記にある二通りの実装において、ゲームを行っている間、PlayStation 2 と Linda サーバーが送受信したパケットの数をいう。

オブジェクトテーブルによる実装では、PlayStation 2 が一番多くパケットの送受信を行っているため、PlayStation 2 をパケット数計測の対象とした。図 11 と図 12 に、その結果を示す。sec は経過時間

(秒)、PS2 は PlayStation 2 の数、packet はパケットの数を表す。

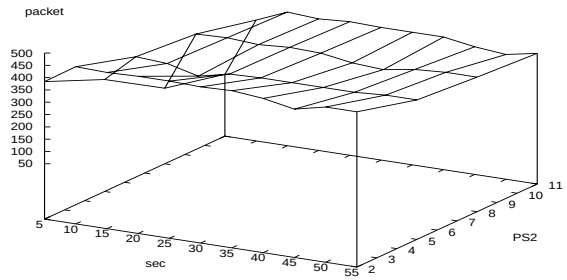

図 11 Agent に対する PlayStaion 2 の送信パケット数

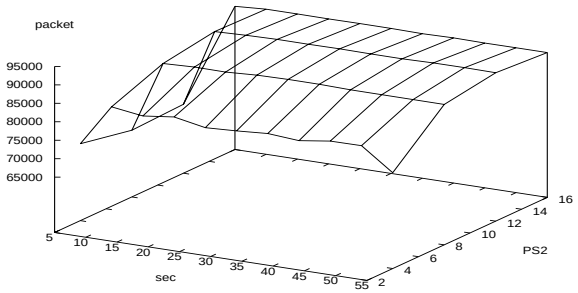

図 14 Linda サーバーの受信パケット数

tion 2 が送受信したパケットの数をグラフで示す。sec は経過時間 (秒)、PS2 は PlayStation 2 の数、packet はパケットの数を表す。

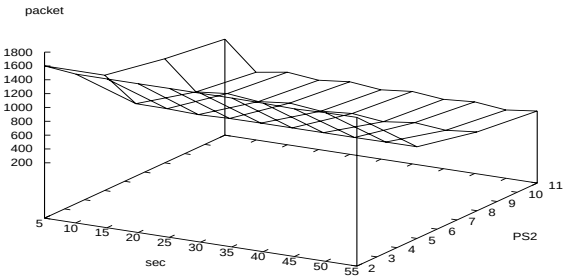

図 12 Agent に対する PlayStation 2 の受信パケット数

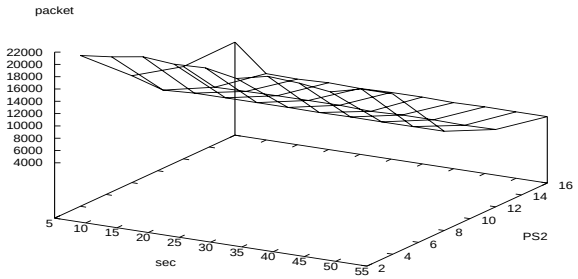

図 15 Linda に対する PlayStation 2 の送信パケット数

図 13 と図 14 に、Linda サーバーが送受信したパケットの数をグラフで示す。sec は経過時間 (秒)、PS2 は PlayStation 2 の数、packet はパケットの数を表す。

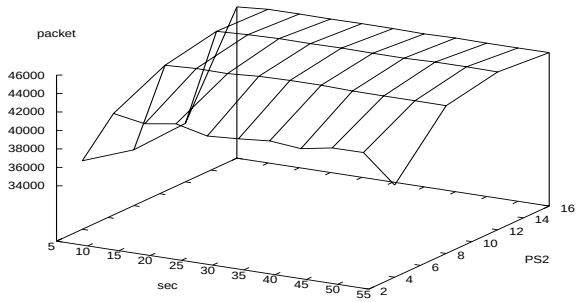

図 13 Linda サーバーの送信パケット数

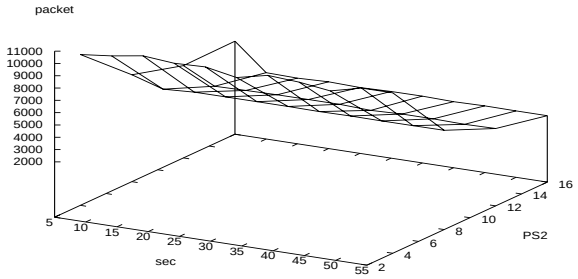

図 16 Linda に対する PlayStation 2 の受信パケット数

図 15 と図 16 に、Linda サーバーに対して PlaySta-

6. 計測結果の評価

6.1 ターンアラウンドタイム

両図とも、初期の数回は乱れているが、それ以降は PlayStation 2 の台数に関わりなく、殆んど同じ値で安定している。(図 9・図 10) その値を比べてみると、Agent・PlayStation 2 間で約 0.07 秒、Linda・PlayStation 2 間で約 0.09 秒となっている。

6.2 パケット数

Linda による実装とオブジェクトテーブルによる実装、それぞれの場合におけるグラフを比較してみる。(図 11・図 12・図 15・図 16 参照) PlayStation 2 の総数が増えるにしたがい、一台ずつのパケット送受信数が減っていくという、同じ傾向が見て取れる。しかし、PlayStation 2 の送受信パケット数を比較してみると、その差は非常に大きく明らかである。

Linda サーバーを使用した通信の場合、単純に Acknowledge が 2 倍となるため、パケット数の増加が顕著であると考えられる。上記の通り、ターンアラウンドタイムに差が出ているのは、パケットの数が関係していると予想される。

パケットの数が通信スピードに影響を及ぼすならば、このまま PlayStation 2 の数を増加させ続けると仮定すると、当然ターンアラウンドタイムは落ちることになる。Linda サーバーの場合、パケット数の増加が顕著である分 (図 13・図 14 参照)、オブジェクトテーブルによる実装に比べて、PlayStation 2(ユーザー) の数の増加による影響を受け易いと考えられる。

6.3 総合評価

計測結果から、Agent によるオブジェクトテーブルを利用した実装の方が、

- パケット数の抑制
- ターンアラウンドタイム (通信速度) の高速化

の面から Linda サーバーよりも成功していると言える。よって、今回実装したフレームワークは、従来使用していた Linda サーバーよりも、よりネットワークゲームに有効だと評価できる。

7. Agent に必要な実装

7.1 パーサー

今回の実装では、パケットが持つデータはバイナリデータであった。しかし、データの汎用性・便宜性を考慮すると、バイナリデータよりもアスキーコードに変換して通信を行う方が良いと考えられる。アスキーコードへの変換に伴い、情報量は増えることとなるが、通信に対して過剰な負荷が起こる程ではない。送られて来たアスキーコードのデータを、解析してローカルホスト用のデータに変換するのが、パーサーである。

7.2 Agent から PlayStation 2 に対する操作

- オブジェクトテーブルの同期
  ゲームが進行するに従って、エージェントと PlayStation 2 のテーブルに違いが出てくる。従って、Agent から PlayStation 2 へ更新作業を行う必要がある。テーブル内の情報が変化した時点で、差異分をエージェントへ伝える。
- ゲーム参加の認証
  Agent にアクセスして来たプレイヤーの認証を、オブジェクトテーブルのプロトコルを利用して行えるようにする。
- オブジェクトデータの展開
  DVD メディアもしくは HDD からオブジェクトデータをメモリに展開させる。
- 動画再生
  DVD メディアもしくは HDD から特定の MPEG2 動画を読み込み、GS へ送って再生させる。
- データの保存
  メモリ上のデータを任意の場所 (メモリーカードや HDD) へ書き込ませる。もしくは、Agent から転送したデータを任意の場所 (メモリーカードや HDD) へ書き込ませる。
- PS2 の接続切断
  ゲームを終了するための処理を行う。

7.3 PlayStation 2 から Agent に対する操作

- 各種の要求
  ゲーム参加 (Agent との接続) の要求やゲーム (Agent) との接続切断要求等。
- 特定の操作に対する Acknowledge
  ムービーの再生や HDD への書き込みの開始・終了等、Agent 側では予測しにくい処理に対する状況を Agent へ知らせる。

8. Suci

今回、オブジェクトテーブルの実装は、TCP を使用して行った。しかし、一つのオブジェクトが持つ描画に必要なデータは、座標・角度・座標系・色・テクスチャの種類等であり、そんなに多くはなく、1 パケット内に収まる可能性が高い。そのため、UDP によるメッセージ通信を採用することによって、TCP による実装よりも高いパフォーマンス得られると考えられる。そこで、本研究室によって開発された、UDP に信頼性を付加した、ユーザレベルでフロー制御を行う通信ライブラリである Suci を用いて、Agent システムを実装し検証してみる必要があると考えられる。

9. まとめと今後の課題

本研究では、Agent を用いたいたネットワークゲーム用フレームワークの実装・評価を行った。その結果、今回実装したフレームワークは、有効であることが示

された。今後の課題として、

- Agent の未完成な機能の実装
- 汎用性のある Agent の作成
- Agent 同士の通信プロトコルの作成
- 百万単位の参加者によるゲームの具体的構想

等があげられる。

参 考 文 献

1) 金内 典充, 今安 正和: UNIX ネットワークプログラミング, オーム社, ISBN 4-274-07778-0
2) ショーン・ウォルトン (著), 野村 純子 (訳): Linux ソケットプログラミング, Pearson Education, ISBN 4-89471-467-1
3) 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
4) 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
5) 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
6) 河野 真治, 仲宗根 雅臣: 同期型タプル通信を用いたマルチユーザ Playstation ゲームシステム, 卒業論文, 1998