PlayStation 2® Linux 上のネットワークゲーム・フレームワークの提案

河野 真治
科学技術振興事業団さきがけ研究 21(機能と構成)

佐渡山 陽
琉球大学大学院理工学研究科

概要

PS2 Linux は、PlayStation 2 上で動作する Linuxである。PlayStation 2 の描画機構である Emotion Engine および専用 DSP である Vector Unit の特徴を活かしつつ、ネットワークゲームを構築するフレームワークを提案する。ネットワーク部分には、リアルタイム処理を考慮したユーザレベル・トランスポート層を持つ Suci ライブラリを用いる。

1 想定するネットワークゲーム

これまで、本研究室ではサッカーやオブジェクトを PlayStation へ転送して表示を行う、等の基本的なネットワークゲームを作成してきた。PlayStation 2 では、一万人規模の多対多通信ゲームを特定の集中管理サーバーの存在しないネットワーク構成で実現することを最終的な目標としている。

2 通信エージェント

現在のネットワークゲームでは、集中管理を行っているサーバーがネットワーク上に存在していて、プレイヤーはサーバーと通信を行うことによってゲーム参加するという形態である。ゲームを運営する側としては管理が楽になるが、サーバーがボトルネックになるのは明らかである。

また、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となると、パワー不足は否めない。また、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ため、通信に対して十分なリソースを提供することが難しい。

そこで、通信の処理を専門に行うエージェントをユーザ一人一人に一つ用意することを提案する。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。

提案するエージェントとは、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 PlayStation 2 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 (図 1 参照)。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。

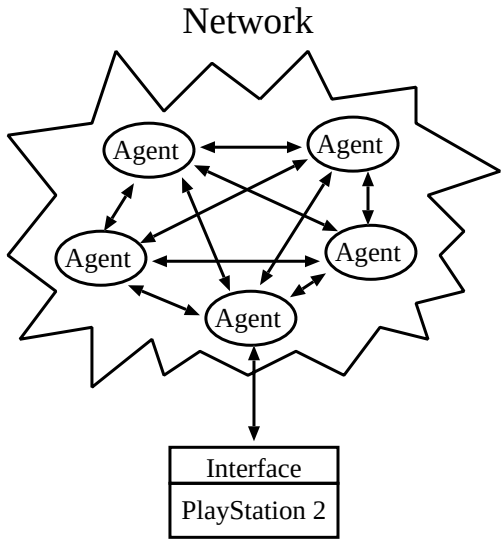

図 1: エージェントを用いた通信システム

3 Linda による通信システム

本研究室では、旧 PlayStation の通信システムとして、Linda を使用していた。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。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。Linda に対するコマンドはキューにためられていき、プログラマが指定したタイミングで (psx_sync() を実行した時)、一気に実行される。

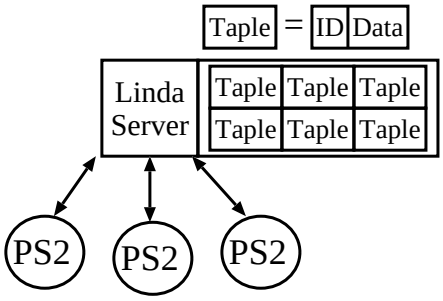

図 2: Linda の通信形態

Linda システムで使用される API の解説を、以下に示す。

psx_out()	指定した ID のタプルに データを書き込む
psx_in()	指定した ID のタプルの 読込み要求を送る 読込み後タプル削除される
psx_rd()	指定した ID のタプルの 読込み要求を送る 読込み後も、タプルは消えない
psx_reply()	サーバーからデータを受け 取り、それを確認する
psx_sync()	キューにたまったコマンドを 実行する

しかし、Linda を用いた通信システムでは、全てのデータを一旦サーバーへ蓄積する必要があるので、そこがネックになってしまう。また、ゲームはリアルタイム性が重要なので、待ち合わせを行ってデータを受け取るまでプログラムが止まる、ということがあってはならない。従って、非同期型を採用しているが、そのために通信相手がデータを受け取るために待ち合わせをしていると仮定することが出来ない。相手がデータを受け取った状態を確認するためには、送ったタプルとは別のタプルを使って Ack を取らなければならない。単純に計算すると、相互に待ち合わせている場合の 2 倍の通信が必要である。こうしたことが原因となり、Linda は通信速度の低下をまねいている。

図 3 は、あるデータを PS2_A から PS2_B へ送り、Acknowledge を取るまでの例を示した物である。PS2_A が psx_out() でデータを ldserv(Linda サーバー) へ送り、Ack が帰ってくるのを待つ。PS2_B は、psx_in() でデータの読み込み要求を行い、reply() でデータの受け取りを行う。その後、今度は PS2_B が別のタプルで Ack を ldserv へ送り、それを PS2_A が reply() で受け取っている。ここまで来て、ようやくデータの送受信が 1 つ終了する。

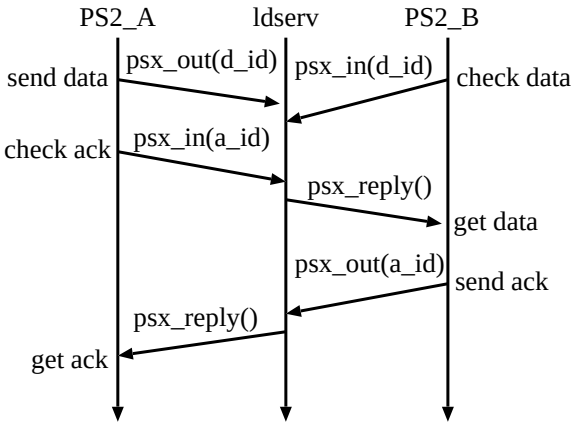

図 3: Linda における Acknowledge 通信

まとめると Linda を使用した通信システムでは、

サーバへのアクセス集中
Ack によるトラフィックの増加

があり、多数の参加者を仮定したネットワークゲームには向かないことがわかった。

4 通信ライブラリの設計方針

Linda システムの欠点をふまえ、新しい通信ライブラリでは、

- 全ての PlayStation 2 が同じ相手と通信をする必要がない

- Ack の返し方を工夫してトラフィックを減らす
- データは必要とされている所だけへ送る

という設計方針で考察する。

5 Suci ライブラリ

新しい通信システムには、Suci ライブラリを採用する。Suci ライブラリとは、本研究室で作成したユーザーレベルでフロー制御を行うための、UDP ベースの通信ライブラリである。複数のパケット毎に Ack まとめて返す等、Ack をコントロールしてスループットを上げることができる。

6 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
- 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。

7 メモリアドレステーブル

Suci の通信は非同期で行われるため、どんなデータがいつ到着するのか分からない。図 4 は、Suci による通信の例を示したものである。各 PlayStation 2 から送られる任意の Data と、それに対して起こる任意の Reaction の送受信は、必ずしも送られた Data の順番通りにやり取りが行われるとは限らない。

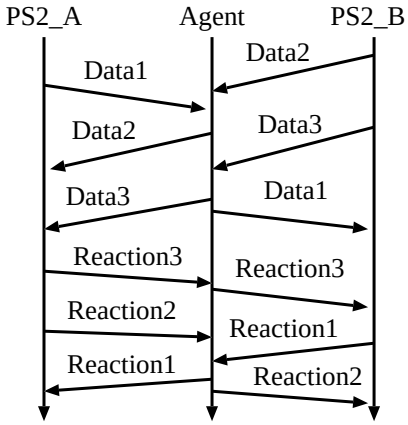

図 4: Suci の非同期通信

7.1 PlayStation 2 側

そこで、それぞれのパラメーターを構造体ごとに個別に ID をつけ、それらを格納しているメモリのアドレスとのテーブルを作る。エージェントからデータが送信されてきた際、テーブルを参照することで速やかにデータを適したアドレスへ格納できる。

PlayStation 2 には、DMA を用いたデータ転送が実装されており、CPU のリソースを浪費せずに、メインメモリや周辺プロセッサ等の間でデータ転送が行える。ただし、転送アドレスは物理アドレスである必要があり、TLB によるアドレス変換を受け付けない。よって、前もってメモリアドレスをテーブルを用意しておくことは、DMA 転送を使用してデータを転送する時にも有利となる。

ただし、このテーブルに必要なのは、他の PlayStation 2 と共有されるデータの情報だけであって、ローカルに閉じているデータは、テーブルに登録しておく必要は無い。

7.2　エージェント側

エージェントにもテーブルを送っておいて、必要なデータだけをエージェントに判断させて送ってもらうようにする。しかし、エージェントは単純に PlayStation 2 のテーブルを持つのではなく、エージェント側では、PlayStation 2 のメモリアドレステーブルの ID とその ID が何の情報を示しているのかというテーブルを作る必要がある。何故なら、各 PlayStation 2 が持つメモリアドレステーブルは固有の物であるから、エージェントのテーブルでその違いを吸収しなければならない。

7.3　テーブルの更新

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、PlayStation 2 からテーブルをエージェントへ送って更新作業を行う必要がある。更新を行うタイミングは、他の通信と同様に非同期で、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。

8　インターフェース

6 と 7 より、エージェント・PlayStation 2 間のインターフェースとして、メモリアド優玲 [傅を看却覧が影　腓き