オブジェクト指向に基づく通信プロトコル性能評価シミュレーションシステム

SAKAI, Katsuya / YOSHIDA, Yutaka / 吉田, 裕 / 坂井, 克也

(出版者 / Publisher)
法政大学工学部
(雑誌名 / Journal or Publication Title)
Bulletin of the Faculty of Engineering, Hosei University / 法政大学工学部研究集報
(巻 / Volume)
35
(開始ページ / Start Page)
9
(終了ページ / End Page)
13
(発行年 / Year)
1999-03
(URL)
https://doi.org/10.15002/00003793

オブジェクト指向に基づく 通信プロトコル性能評価シミュレーションシステム

坂井 克也 吉田 裕

Object Oriented Simulation System for Performance Evaluation of Communications Protocols

Katsuya SAKAI and Yutaka YOSHIDA

Abstract

It is important to evaluate performances of communications protocols before implementing or substituting them. This report describes on a newly designed system based on object oriented concepts, especially centering on system configuration, including class definitions and newly designed functions such as protocol data transmission and switching. Further, examples of extensibility of this system and execution time reduction schemes are also shown.

1 はじめに

新しいプロトコルの設計や既存プロトコルの改良を行う場合、プロトコルの性能評価が重要である。しかし、このようなプロトコルは複雑であるため、理論解析や数値解析によって性能評価を行うことは難しい。そのため、計算機によるシミュレーションによって性能評価を行うことが一般的である。

我々の研究室では以前から通信プロトコル性能評価を容易に行なえるようなシミュレーションシステムの作成を行ってきた。[1]

今回はこのシステムをベースにオブジェクト指向の観点からシステム構成を見直し、システムの機能拡張を容易に行なえるようにした。また、汎用シミュレーションシステムでは困難なフレームの構造が関係するプロトコルのシミュレーションや、シミュレーション実行速度を改善するための機能についての検討を行なった。

2 オブジェクト指向

オブジェクト指向は、実世界にある「もの」をそのまま基本要素に据え、それらの関係を記述することによってプログラミングを行っていくソフトウェア開発手法である。オブジェクト指向の基本的概念としてカプセル化や継承などがあり、これらの効果としては開発の効率化や保守性の向上などがあげられる。

シミュレーションシステムにおいては様々なシミュレーションに対応出来るよう新しい機能を随時容易に追加できるような構造が望まれる。継承の概念を用いると親クラスの性質をそのまま引き継ぎながら新たな性質を追加できたり部分的に性質を置き換えることができるので、クラスの拡張が行いやすいので、オブジェクト指向言語はシミュレーションシステムの作成に適していると言える。

このようなことから、オブジェクト指向言語である C++[2] を用いてシミュレーションシステムの作成を行うことにした。

3　通信プロトコル

通信プロトコルとは、通信を行う上での相互の取り決めのことである。

ISO と ITU-T により標準化された OSI 参照モデル (図 1) では、開放型システムを階層に区分したときの一要素をサブシステムと呼ぶ。サブシステム内にはエンティティと呼ばれるその層の機能を実行する能動的要素が存在する [3]。(以下、このエンティティのことを機能要素と呼ぶ)

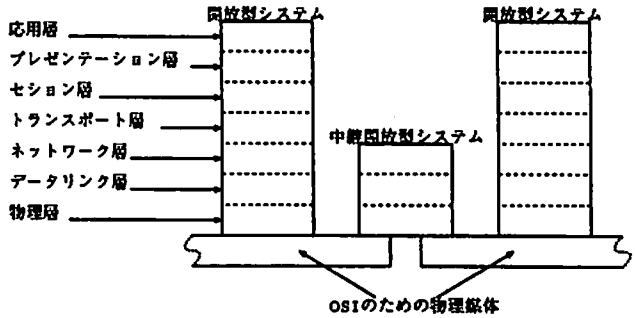

Fig 1: OSI 参照モデル

評価対象プロトコルでデータ伝送シミュレーションを行い、平均遅延やスループット等の統計値をとることによって性能評価を行うが、汎用シミュレーションシステムではフロー制御や確認応答の処理を行ったり、フレーム伝送でフレーム構造を考慮した伝送を表現することが難しく、シミュレーションプログラムが複雑になるという問題がある。

今回作成したシミュレーションシステムでは、機能要素の種類、初期状態、位置関係を記述することによってシミュレーションプログラムの作成を行うようにしている。そして、機能要素として通信プロトコル用機能をいくつか用意することにより、容易に通信プロトコル性能評価シミュレーションを行えるようにした。なお、シミュレーションプログラムとは、シミュレーションシステムを用いてシミュレーションを行うための記述のことで、シミュレーションシステム自体のプログラムとは別である。

4　シミュレーションシステム

4.1　構成

シミュレーションシステムでは、シミュレーション中の時間を進行させるための時間管理機能、与えられた確率分布に従う乱数を生成する機能、統計値収集機能、シミュレーション対象システムの構成要素を記述する機能、対象システムの状態変化を表現する機能、が最低限必要となる。

オブジェクトは自分自身の情報を自分自身で保持・制御するようにモデル化すべきであるので、今回作成したシミュレーションシステムでは以下のようなクラス構成とした。

- シミュレーションクラス (シミュレーションの進行を行う)
  - 時間管理機能
  - イベント管理機能
  - 機能要素の位置管理機能

- 乱数生成クラス (様々な分布の疑似乱数を生成する)
- 統計収集クラス (統計データを収集／計算)
- データ単位クラス
- 機能要素クラス (シミュレーション対象システムの構成要素)

システム中の状態変化はその状態変化の起こる時刻と共にイベントとして扱われ、シミュレーションクラス中のイベントリストによって管理される。

シミュレーションプログラムは、機能要素の宣言、それぞれの機能要素についての位置関係と初期条件を定義を行い、最後にシミュレーションを開始させる関数を呼び出すという形で記述される。

4.2 通信プロトコル用機能

汎用シミュレーション言語で通信プロトコルに関するシミュレーションを行おうとするとプログラムが複雑になることが多い。これは汎用のシミュレーション言語には通信プロトコルを記述するのに便利な機能が少ないためである。

そこで通信プロトコルに関するシミュレーションを容易に行えるような機能要素について検討した。機能要素は基本的にはOSIモデルのエンティティに対応するが、複数の機能要素を1つの機能として扱うこともできるので、サブシステムや複数のサブシステムの集合にもなり得る。

汎用シミュレーションシステムではデータ単位 (パケット) は機能要素から機能要素へ瞬時に移動する。このため、同一計算機内部でのデータのやり取りを表現するのは容易であるが、パケットの構造を考慮したデータ伝送を表現したい場合などは、ユーザがパケット先頭／データ部先頭／最後尾などのデータ構造を意識してプログラムを作成する必要がある。

本システムでは、そのような事柄をあまり意識せずにプログラムを作成できるように、フレーム伝送機能を作成し容易にデータ伝送を表現できるようにした。

その他、プロトコル記述を容易にするため、ヘッダ付加／削除、ウインドウ制御、順序制御、カットスルー[4]、割り込み伝送などの機能も作成した。

4.3 実行速度の改善

シミュレーション用の言語を用いたシミュレーションは、一般に、汎用言語を用いた場合よりも計算時間が大きくなる。この要因としては、汎用言語を用いたシミュレーションでは実行したいシミュレーションにあわせて最適な処理を行えるという点があげられる。

シミュレーションシステムにおいても行いたいシミュレーションに合わせて処理を最適化することができれば高速化につながると考えられる。

今回は、データ収集を場合により行わないようにすることができるようにして高速化を計った。また、別のアプローチとして、イベント管理のためのイベントリストのデータ構造による処理の効率化も検討した。

4.3.1 収集データ選択による高速化

一般のシミュレーション言語では、いろいろな性能評価を行えるようにするため、シミュレーション中に内部で様々なデータを収集して結果を出力する。しかし、特定のデータだけが必要な場合には、かえって無駄な処理をしていると言える。そこで収集するデータを選択できるようにして、必要の無いデータは収集しないようにする機能を考えた。

この機能により実行速度は表1のようになった。(OS:FreeBSD2.2.6/実行時間の計測には tcsh 組み込みコマンドの time を使用)

Table 1: 実行時間

収集データ数	1	3	25
実行時間	25.43	25.75	30.73

この結果から、収集するデータを制限することで高速化が図れているということが確認できた。

4.3.2　イベントリストの構造

シミュレーション処理中で最も頻繁に実行される処理はイベントリストへの操作であり、この操作を効率が実行速度にかなり影響する。よってこの操作を効率良く行うことのできるデータ構造を検討した。

イベントリストへの操作は以下の3つである。

- イベントを登録する
- 発生時刻の最も小さいイベントを取り出す
- 任意のイベントを削除する

ここではデータ構造として線形リスト、ヒープ条件を満たす2分木の2つの構造の計算量の比較を行った。ここでヒープ条件とは「木に含まれる全てのノード n について、n のデータ値(イベント発生時刻)が、n のどの子のデータ値よりも常に小さい」というものである。

それぞれの構造における各操作の計算量は表2のようになる。(n はイベント数)

Table 2: 計算量の比較

	線形リスト	ヒープ木
イベント登録	$O(n)$	$O(\log_2 n)$
最小値取り出し	$O(1)$	$O(\log_2 n)$
任意のイベント削除	$O(n)$	$O(n+\log_2 1) \sim O(1+\log_2 n)$

なお、線形リスト構造ではイベント登録時に発生時刻順になるように並べかえを行い、ヒープ構造では各操作後に木の平衡が乱れないように処理を行っている。

個々の操作を比較すると線形リストの方が良い場合もあるが、シミュレーション中で各操作が呼び出される回数は、"登録 $\geq$ 取り出し + 削除" であるので、総合的にはヒープ構造のほうが効率的であると言える。

5　機能要素の拡張

本システムでは、オブジェクト指向の基本概念の一つである継承を用い機能要素クラス(図2)から派生させて各機能を作成するようにしている。これによってシミュレーションクラスからは各機能を同じ「機能要素」として扱うことが可能となっている。

新しい機能を作る場合、機能要素クラスを継承させ、その機能の初期条件を設定するメソッド、イベントを実行するメソッドを定義しなおすだけでよい。必要があればシミュレーション実行前の前処理、実行後の後処理の関数も定義でき、ユーザが自由にメソッドを追加してもよい。

```
class Entity {
  public:
    virtual int execute(Event *event); // イベントを実行
    virtual Message  reply(Place from, Message  msg);
    virtual dMessage reply(Place from, dMessage msg);
                              // 他の要素からのメッセージを処理
    virtual void pre_proc();  // 前処理
    virtual void post_proc(); // 後処理
    void set(Flag flag);    // フラグ設定
    void unset(Flag flag); // フラグ設定
  protected:
    Place Number;       // 要素番号
    Flag  CollectFlag; // フラグ
    Entity *prev;       // 隣接要素のポインタ
    Entity *next;       // 隣接要素のポインタ
    ....
};
```

Fig 2: 機能要素の抽象クラス

また、既存の機能のクラスを継承させて一部の処理を置き換えることによっても、新機能を作成することもできる。例えば、バーチャルカットスルー機能を作成するには、既存の純カットスルー機能を継承させて、「出線がビジーなら呼損とする処理」を「出線がビジーなら一旦全蓄積して送信する処理」に置き換えればよい。

このようにシステム作成にオブジェクト指向言語C++を採用したことによりシステムの拡張が容易になった。

6 おわりに

今回、通信プロトコル性能評価用シミュレーションシステムの作成を行い、容易に通信プロトコルのシミュレーションが行えるような機能を作成した。そして、実行時間を短縮するために収集データ選択機能の作成、イベントリストの構造の検討を行った。また、オブジェクト指向言語である C++ を使用したことにより、新しい機能の追加などが比較的容易に行えるようになった。

今後は、より通信プロトコルの性能評価を行いやすくするための機能拡張を行いながら、作成したシミュレーションシステムを用いてカットスルーなどの性能評価を行っていく予定である。

参考文献

[1] 小菅美帆, 吉田 裕, "通信プロトコル性能評価用シミュレーションシステム", 法政大学計算センター研究報告 9, pp61-66, 1996

[2] Bjarne Stroustrup, "プログラミング言語 C++", 株式会社トッパン, 1993

[3] "JIS ハンドブック 情報処理 OSI 編", 日本規格協会, 1993

[4] 深山 修, 吉田 裕, "遅延の少ないフレーム伝送・交換方式", 法政大学研究集報 (第 34 号), 1998

[5] 坂井克也, 吉田 裕, "プロトコル性能評価シミュレーションの実行速度の高速化", 信学総大, B7-71, 1998-3

[6] 坂井克也, 吉田 裕, "離散イベントシミュレーションシステムへのフレーム伝送機能の付与に関する検討", 信学ソ大, B7-53, 1998-9