データフロー型高速通信処理方式の一検討

1V－2

妹尾　尚一郎，坪根　宣宏，井手口　哲夫，厚井　裕司
三菱電機株式会社　情報技術総合研究所

1．はじめに

インターネット上でマルチメディア通信やエレクトロニックコマースのサービスを本格的に普及させるには，サーバにおける通信処理の高性能化と，ユーザ数やネットワーク規模の拡大に対応したスケーラビリティが求められる．このような要求に対応するため，データフロー技術を適用した高速通信処理方式について検討した．サーバにおける通信処理には多くの並列性が内在するため，並列処理を用いた高速化が有望であるが，代表的な並列処理技術であるデータフロー技術はデータの受渡しにより実行制御を行うので，メッセージないしパケットの受渡しによって処理が進む通信処理と親和性がある．

本高速通信処理方式では，データフローの処理単位を大きく取る粗粒データフロー技術[1]を採用し，通信処理をトップダウン的に並列実行可能な部分へと分割する．筆者らはこのアプローチに基づき電子メール並列処理の試作[2]を行った．本稿では，OSI参照モデルのトランスポート層をモデルとした単純なコネクション型プロトコルを対象として，並列処理方式の検討およびシミュレーションによる動作確認について報告する．

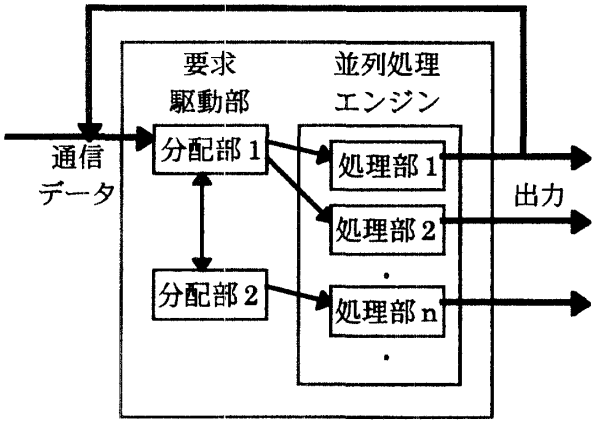

図1　並列処理の概念図

A Study on a High-Performance Communication Processing Using Dataflow Technology,
Shoichiro SENO, Nobuhiro TSUBONE, Tetsuo IDEGUCHI and Yuuji KOUI,
Information Systems R&D Center, Mitsubishi Electric Corporation, 5-1-1 Ofuna, Kamakura, 247 JAPAN.

2．データフロー型高速通信処理方式

図1に本方式における並列処理の概念図を示す．本方式はマルチプロセッサを前提として，データの処理要求が発生する毎に分配部が各プロセッサに分散配置された処理部の中から負荷の低いものを選んで，動的に処理すべき仕事を割当てる．処理部毎の処理は並列に実行されるが，処理部の中では処理が逐次的に進む．処理部への分配が動的であるためプロセッサの稼働率が高く，またプロセッサに障害が発生してもそのプロセッサを分配対象から外すことで耐障害性が確保できる．

図1のシステム上で実行されるプログラムは，従来のプログラミング言語で記述されるプログラムモジュール（各並列実行部分に相当する）と，プログラムモジュール間の実行制御をデータの流れとして記述するデータフロープログラムから成る．並列実行部分ではC言語など従来言語で記述されたソフトウェア資産の再利用が可能であるため，並列処理に伴うプログラミング負担が少ない．

通信処理を並列実行部分へ分割して並列性を導入するには，次の2つの手法が適用できる．

①水平並列性の導入：同時に存在する複数の通信相手との間の通信コネクションを並列に処理する．
②垂直並列性の導入：プロトコルのレイヤ構成に従い，各層での処理を並列に実行する．

①では並列実行部分が等価な処理となるので，通信処理の並列部分への分割は比較的容易である．①をコネクション型プロトコルへ適用した場合の並列処理の模様を図2に示すが，共用プロセスは図1の分配部に，並列プロセスは図1の処理部に相当する．

②はプロトコル処理の階層性に着目したもので，並列実行部分の等価性が成り立たず，①に比べプログラム作成が困難となる．

３．プロトコル処理のシミュレーション

トランスポート層プロトコルをモデルとした単純なコネクション型プロトコルを想定し，ワークステーション上でシミュレーションを行った．並列性の導入には上記①のみを用い，粗粒データフロー技術[1]を用いてコネクション毎のプロトコル処理を並列実行するプログラムを作成した．本プログラムにおいて状態遷移処理を制御するデータフロープログラム部分を図３に，これに対応するデータフローグラフを図４に示す．

上記プログラムをワークステーション上に実装された粗粒データフローシミュレータ上で実行した結果，コネクションの確立・解放を制御しつつコネクション毎の状態遷移処理の並列実行が可能であることが確認された．

４．まとめ

粗粒データフロー技術による高速通信処理方式①，②について検討した．インターネット上のサーバでは多数ユーザへの同時サービスが課題となっているので，①のみの適用でも有効と考えられる．今後は実際のアプリケーションへ本方式を適用し，その有効性を検証する予定である．

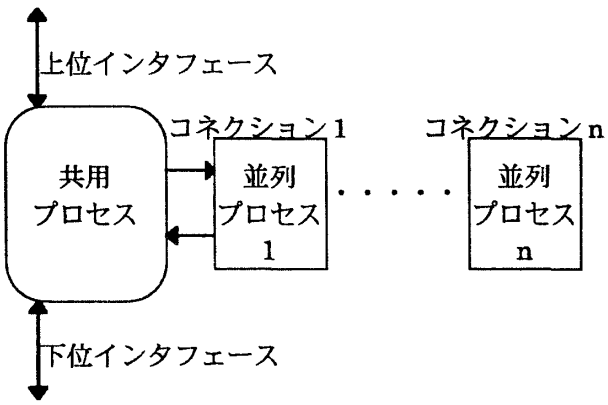

図２　コネクション毎の並列処理

```
report where
  report = Report(fsm_state, event) @.connection connid
    where
      index connection;

      fsm_state = if IsClosed (connection_a)
        then ReleaseConnection(connection_a)
        else Extract_state(connection_a) fi;
      connection_a = Action(connection_b, event);
      connection_b = if IsNew(event) then Initial(connid)
        else GetConnectionBlock(event) fi;
    end;
  connid = ConnMgnt(event);
  event = GetEvent();
end
```

図３　データフロープログラム

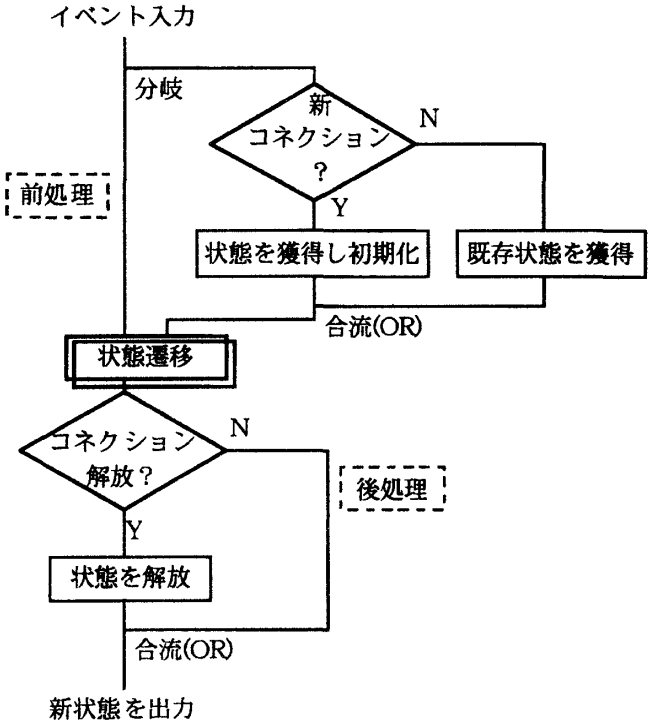

図４　状態遷移処理を制御するフローグラフ

参考文献

[1] R. Jagannathan and A. Faustini, "GLU: A system for scalable and resilient large-grain parallel processing", Hawaii International Conference in System Sciences, HICSS-24, Kauai, Hawaii, Jan. 1991.

[2] Y. Koui, S. Seno, T. Yamauchi, M. Ishizaka and K. Kotaka, "Implementation and Evaluation of MHS Parallel Processing", IEICE Trans. Commun., Vol.E77-B, No.11, pp.1388-1397, Nov. 1994.

[3] 石坂，土田，井手口，"OSI プロトコルにおける受信データの並列処理方式"，信学論(B-I), Vol.J74-B-I, No.2, pp.116-128, Feb. 1991.