LOTOS仕様の実装支援システムの開発

6P−7

小野 良司† 程 子学†† 白鳥 則郎†

†東北大学工学部 ††東北大学応用情報学研究センター

1 はじめに

筆者らは, LOTOSを用いて記述されたシステムを効率良くインプリメントすることを目的として, LOTOS仕様の自動実装法[1]に基づき, LOTOS-to-Cトランスレータを開発・実装している。このトランスレータによって出力されたプログラムを, 実際に分散ネットワーク環境上でマルチランデブの分散アルゴリズム(MRDA)[2]に基づいて動作させる場合には, さまざまな問題が生じる。

本稿では, LOTOSマルチランデブ実装における, プロセスの数や配置の動的変化に対応した, 論理的, 物理的ネットワークの構成法および同期木の形成法およびその実装について述べる。

2 実装上の問題点

LOTOS-to-Cトランスレータ では LOTOSのプロセスを OSのユーザプロセスに変換する。以下, 単にプロセスといえばこのLOTOSプロセスから変換されたユーザプロセスを指すものとする。

2.1 プロセスが持つべき情報

これらの各プロセスは, 実際に環境上で動作するときには以下のような情報を持たなければならない。

1. ネットワーク上に存在する物理的ノード(マシン)に関する情報(ネットワーク情報)
2. 自らとメッセージ交換するプロセスに関する情報(コネクション情報)

1は主に新しいプロセスの生成に必要であり, 2はMRDAを適用したときに必要である。これらの情報を直接プロセス自身に持たせるためには, LOTOSからC言語への変換が繁雑になる。

2.2 実装上の問題

前述ようなの情報を各プロセスが持ったとき, プロセスの数や配置が動的に変化するときには, 以下のような問題を解決しなければならない。

1. 各プロセスをどのマシン上に配置するか。
2. プロセス間にどのように通信路を設定するか。
3. プロセスの生成・消滅があったとき, 通信路をどのように接続し直すか。
4. 形成された論理的ネットワーク(プロセスによるネットワーク)上でどのようにして同期木を形成するか。

こうした問題を解決するために, 3章に述べる実装法を提案する。

3 実装法

3.1 概要

LOTOSプロセスは動的に変化するが, 動的な環境上で以上のような問題を解決するのは容易ではない。これらの問題を静的環境上の問題に置き換えるため, コントロールプロセスを導入する。

さらに, 2.1で述べた情報の分離のために, 環境を2階層から構成する。

最後にプロセスの配置法を述べる。

3.2 コントロールプロセスの導入

各マシン上に一つずつコントロールプロセス(以下CPと略称)を置く。仕様の実装者は, あらかじめ各マシン上にCPを置き, これにネットワーク情報を付与し, CP間に通信路を設定しておく。

各プロセスは自らの配置されたマシンのCPに接続する。各CPは他のCPとメッセージを交換することによって, 接続されたプロセスに対し通信媒体として機能する。

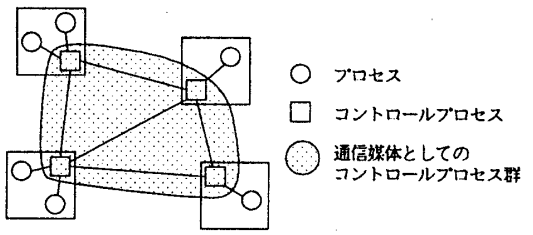


図1: コントロールプロセス

CPを用いることにより, 各プロセスはCPへの接続によって直ちに他のプロセスとの通信路が確保されるため, 問題2.2-2が解決される。また, プロセスの生成・消滅時にはCPとの接続を設定・解消することによって論理的ネットワークは自動的に変更されることになり, 問題2.2-3が解決される。

3.3 実装環境

コントロールプロセスを導入した環境下で, さらに実装環境を以下のように階層分けする(図1参照)。

1. CP群を利用して通信するプロセス群
2. CP群によるネットワーク

---

*A support system for implementation of LOTOS specification

**Ryoji ONO†, Zixue CHENG††, Norio SHIRATORI†

†Faculty of Engineering, Tohoku University

††Reserch Center for Applied Information Sciences, Tohoku University

1のプロセス群はCP群とのみ通信し，自分以外のプロセスの存在を意識することはないものとする。また，プロセスの生成はプロセスからの要請によってCP群によって処理する。これによって各プロセスが，ネットワーク情報およびコネクション情報(2-1)を意識することなく通信することが可能になる。

この環境に合わせ，プロセスおよびCPの動作を以下の三つのフェーズに分ける。

1. プロセス生成・消滅フェーズ
2. 同期木形成フェーズ
3. ランデブ判定フェーズ

1ではプロセスの生成・消滅とそれに伴うネットワーク情報の変更が，2では同期木の形成が，3ではランデブの生起の判定が，それぞれメッセージ交換によって実現される。フェーズ2によって問題2.2-4が解決される。

以下に各層の環境の概要を示す。

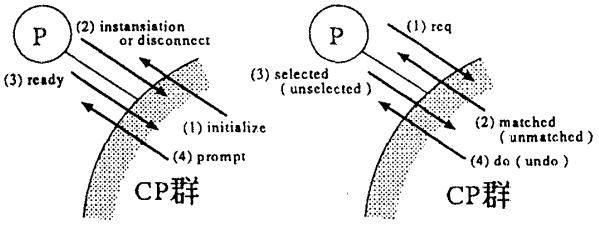
3.3.1 プロセス群

プロセスとCP群とのメッセージ交換の様子は図3のようになる。

フェーズ1では，CPからのinitializeメッセージに対し，プロセスを生成するプロセスや消滅するプロセスは，それをCPに通知する。readyメッセージを送信して次のフェーズに移行する。

フェーズ2では，CP群からpromptメッセージを受信したら次のフェーズへ移行する。

フェーズ3では，各プロセスはCP群にreqを送信し，matched (unmatched)を受信する。同様にselected (unselected)を送信してdo (undo)を受信することで，ランデブの判定を終る。



(a) フェーズ1 (1, 2) およびフェーズ2 (3, 4)　(b) フェーズ3

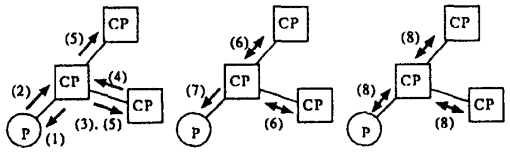
図2: CP群と通信するプロセス

3.3.2 CP群

フェーズ1では，各CPは自分につながるプロセスにinitializeを送信し，プロセス生成メッセージまたはプロセス消滅メッセージを受信したらその処理と情報の変更を行なう。すべてのプロセスからreadyを受信したら次のフェーズへ移行する。

フェーズ2では，文献[3]のアルゴリズムを適用してルート付き最小生成木を形成する。さらにゲート情報を交換して同期木を形成する。プロセスにpromptメッセージを送信して次のフェーズへ移行する。

フェーズ3では，各CPが論理的ノードとしてMRDAをほぼそのまま適用する。



(1) initialize (2) instansiate or disconnect or ready
(3) instansiate (4) complete (5) ready (7) prompt
(6) MST[3] のメッセージ (8) MRDA[2]のメッセージ

図3: CP同士の通信

3.4 プロセス配置

プロセスの配置はCP群によって行なう。トランスレータの仕様として，LOTOSのプロセス生成において配置ノードを指定する機能を与え，生成されたプロセスはこれによって指定されたノードに配置する。指定のないプロセスについては，配置するノードをランダムに決定する。これで問題2.2-1が解決する。

4 議論

ここでは(1)本稿の方法と，(2)CPを用いず各プロセスにすべての情報を持たせ，プロセス同士の通信のみで2.2の各問題を解決する場合を比較する。

(2)では，生成されたプロセスは，親プロセスから情報を受け継ぎ，また自分が接続すべきプロセスを判断して，そのプロセスに接続要求を出さなければならない。そのため各プロセスは，常に接続要求を受ける用意をする必要がある。(1)ならばこれらはすべて必要ない。

このように本稿の方法では，プロセスの動的変化に効率良く対応し，それに伴う時間と資源を節約できる。

5 おわりに

今後の課題として，効率の良いプロセスの配置法を提供することが挙げられる。

参考文献

[1] Cheng Z., Takahashi K., Shiratori N. and Noguchi S. : "An Automatic Implementation Method of Protocol Specifications in LOTOS", IEICE TRANS. INF. & SYST., **E75-D**, 4, pp. 543-556 (1992-7).

[2] 程 子学, 白鳥 則郎, 野口 正一 : "ネットワーク環境におけるLOTOSマルチランデブ実装のための分散アルゴリズム", 信学論 (D-I), **J72-D-I**, 8, pp. 545-554 (1992-8).

[3] Gallager R. G., Humblet P. A. and Spira P. M. : "A Distributed Algorithm for Minimum-Weight Spanning Trees", ACM Trans. Prog. Lang. Syst., 5, 1, pp. 66-77 (1983).