# LOTOSマルチランデブの実装とその応用

小野 良司[†]　Zixue Cheng[††]　白鳥 則郎[†]

[†]東北大学　[††]会津大学

**Abstract**

LOTOSはマルチランデブという高度な通信プリミティブを持っているため，抽象度の高い記述が可能である。しかし，マルチランデブと，実際にネットワークで用いられている通信プリミティブとの間には大きなギャップがあるため，その実装には困難を要する。本研究では，メッセージ交換を基にした，マルチランデブのネットワーク環境上での実装法を与え，これを用いて複数プロセス間での資源競合の解決法を与えることにより，この実装法の有効性を示す。

## 1　まえがき

LOTOSはISOによって開発された形式記述技法 (FDT) の一つであり，種々のアプリケーションやプロトコルなどの仕様記述を厳密かつ曖昧なく行なうことができる。しかしLOTOSは記述レベルの非常に高い言語であるため，LOTOSを用いたアプリケーション開発においては，実装時における支援が不可欠である。この目的のため，LOTOSで記述された仕様を自動的に実装しようとする研究が多く行なわれている [5]。

この自動的な実装は，実装の対象となる環境が単一のコンピュータシステムである場合とネットワーク環境である場合の二つに大別される。単一のコンピュータシステムでの実装では，各プロセスはシステム上の共有メモリなどの通信手段を用いることができるが，実装の対象がネットワーク環境であるときには，LOTOSの各プロセスはネットワークを介して通信しなくてはならない。

しかし，LOTOSにおいてプロセス間の通信はゲートと呼ばれる通信路を介して多重同期的に行なわれる。これはマルチランデブとよばれる高度な通信プリミティブである。マルチランデブと，実際のネットワークシステムで用いられる通信プリミティブとの間には大きな隔たりがあるため，LOTOS仕様の実装においてはマルチランデブをどのようにして実現するかが重要である。

---

*An Implementation of LOTOS Multi-rendezvous and Application

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

[†]Tohoku University　[††]University of Aizu

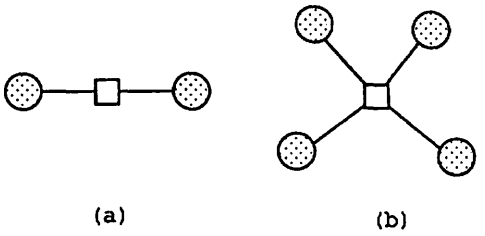


図 1: ランデブ (a) とマルチランデブ (b)

本研究では, プロセス間のメッセージ交換を基にした, LOTOS マルチランデブのネットワーク環境上における実装法を与え, 実際にこれをネットワーク環境上に実現し, 試用を行なった。またこれを用いて簡単な分散的システムの設計 / 実装を行なうことにより, この実装法の有効性を示す。

## 2 マルチランデブ

### 2.1 マルチランデブとは

二つのプロセスの間での同期的な通信をランデブ (rendezvous) と呼ぶ。また二つのプロセスによって同期が行なわれることに注目して two-way rendezvous とも呼ぶことができる。

これに対し, 二つ以上のプロセスが同時に同じ同期的通信に関与するとき, これをマルチランデブ (multi-rendezbous) と呼ぶ (図 1)。

LOTOS におけるマルチランデブは, 同期的な通信路であるゲートを用いたプロセス間通信の形で表現される。また LOTOS では, 各プロセスは同時に複数のゲートでの通信を試みることができる。このため LOTOS においては, 複数のランデブが同時に並行して, かつそれらが相互に関係しあって生じることができる (図 2)。

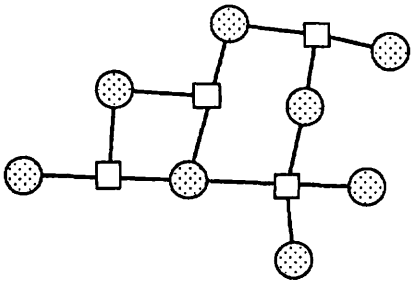

図 2: LOTOS におけるマルチランデブ

### 2.2 LOTOS マルチランデブの性質

LOTOS おけるマルチランデブには, その文法的特徴から以下のような性質がある [1]。

1. 同期性

   同期条件を満たさないグループのイベントは生起できない。ここで, 同期条件とは, 文献 [2] で定義された値の一致 (Value Matching) を指す。

2. 局所的非決定性

   局所的非決定性とは, 一つのプロセスに複数の生起可能なランデブがある場合, 以下の (1) かつ (2) のことである。

   1. 高々一つのランデブしか生起できない。
   2. 複数の生起可能なランデブから非決定的に一つを選択する。

3. 大域的非決定性

   大域的非決定性とは, 複数のグループがそれぞれ同期条件を満足し, かつ

ある共通のプロセスの参加が必要な場合, 以下の (1) かつ (2) のことである。

1. 高々一つのグループしか生起できない。
2. 複数の生起可能なグループから非決定的に一つを選択する。

さらに, LOTOS では非同期にプロセスの生成 / 消滅が起こり得る。このことと非同期性とから, 以下の性質が生じる [3]。

4. ランデブする相手のプロセスは, 直前のランデブが終るまでわからない

## 3 実装

### 3.1 実装における主問題

以上に述べたような性質を持つ LOTOS マルチランデブをネットワーク環境上に実装しようとする場合には, 以下のことを与えなければならない。

1. どのプロセスとランデブすべきかを知る方法

   あるプロセスがランデブしようとしている (複数の) プロセスは, そのプロセスと同じマシン上で動作しているとは限らない。このため, 各プロセスが自分とランデブすべきプロセスを知る方法を与えなければならない。
2. あるランデブが生起可能かを知る方法

   ランデブを行なおうとする各プロセスの間で, 同期条件が一致するかどうかを判定しなければならない。
3. どのランデブが生起するかを決定する方法

   非決定性に基づいて, 同期条件の成立する複数のランデブから実際に生起するランデブを選択しなければならない。

### 3.2 実装法

筆者らは, これらの問題に対応した LOTOS マルチランデブのネットワーク環境上での実装法として, 文献 [1] に示す実装法を提案した。以下にこの実装法の概略を示す。

#### マルチランデブの分散実装法

実装環境として, プロセスとその間をつなぐ通信路からなるネットワークを考える (図 3(a))。各プロセスは, 隣接するプロセスと非同期にメッセージ交換を行なうことができる。

ランデブの相手を特定するため, 同じゲートでランデブを行なおうとするプロセスでルートつきの木を形成する。これを同期木と呼ぶ (図 3(b))。同期木は最初のネットワークの部分木である。

この同期木上でメッセージ交換を行い, 同期条件の判定, 非決定性の解決を行なう。これによってマルチランデブが解決される。

### 3.3 実装法の問題

この実装法を実際にネットワークシステム上に実装しようとするときには, さらに以下のような点を考慮する必要がある。

1. プロセス間の通信路の設定法が与えられていない

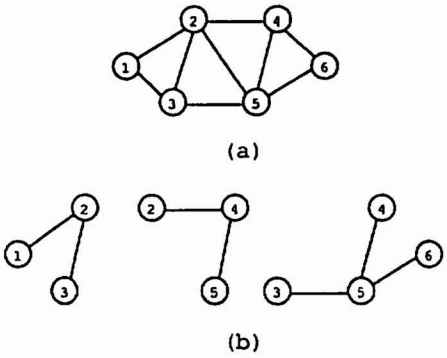


図 3: マルチランデブの分散実装法

上記の実装法は，プロセス間に通信路が設定されたところから始まる。そのため，プロセス間の通信路をどのようにして設定するかを与えなければならない。

2. 同じゲートで同期するプロセスグループの形成法が明示されていない

同期木の形成は，同じゲートで同期するプロセス同士が文献 [4] の方法によって木を形成することになっているが，このアルゴリズムを具体的にどのように適用するかは示していない。

### 3.4 実装法の追加

マルチランデブの実現にあたって，以下の二点を実装法に追加して用いた。

#### 3.4.1 プロセス間の通信路の設定法

LOTOS においては各プロセスは非同期に生成 / 消滅するため，ある時点で存在するプロセスの情報を，各プロセスが自律的に獲得するのは困難である。

そこで，マルチランデブが実装されるネットワーク環境上の各マシン上に，この環境上に存在する LOTOS プロセスの情報を管理するプロセスを設置する。これらのプロセスを管理プロセス (CP) と呼ぶことにする (図 4)。

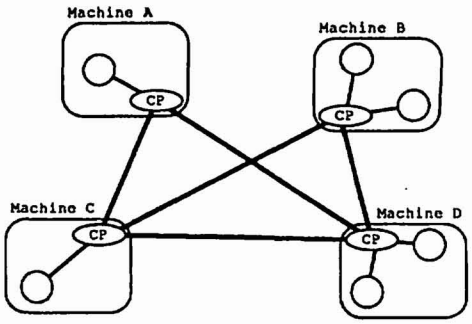


図 4: 管理プロセス

CP 間はあらかじめ設定される通信路によって接続される。また通信路の故障は考慮しない。さらに，すべての CP はあらかじめ設定される一つの木を形成し，その木の上で存在するプロセスの情報の交換を行なう。

CP は，CP 間でネットワーク上に存在するプロセスの情報を交換し，LOTOS プロセスからの要求によってメッセージの転送およびプロセスの生成を行なう。CP の動作の概略を以下に示す。

**管理プロセスの動作**

CP は受信するメッセージにしたがって処理を行なう。受信するメッセージは LOTOS プロセスからのものと，CP からのものに分けられる。

**LOTOS プロセスからのメッセージ**

以下の4つがある。

1. ランデブ開始要求メッセージ (ready)
   ランデブのためのメッセージ交換の用意ができたことを知らせるメッセージ。メッセージに書かれたプロセスのidとその同期すべきゲートを記録する。自分のマシン上のすべてのLOTOSプロセスの用意ができたら, CPはそれらにランデブ開始メッセージ (ok) を送る。
2. プロセス生成メッセージ (instantiate)
   LOTOSプロセスの生成を要求するメッセージ。プロセスを生成するマシンをランダムに決定する。それが自分のマシンならばプロセスを生成し, readyメッセージを待つ。そうでなければそのマシンのCPにinstantiateを送る。
3. 終了メッセージ (exit)
   LOTOSプロセスの終了を知らせるメッセージ。自分につながるすべてのLOTOSプロセスが終了したら, CPは終了確認メッセージ (ending) (後述) を他のCPに出して終了確認を試みる。
4. 転送メッセージ
   他のLOTOSプロセスに送信されるべきメッセージ。自分につながるプロセス宛ならばそのプロセスに送る。そのプロセスの場所を知っている場合はそのマシンのCPへ送る。そうでなければ問い合わせメッセージ (where) (後述) を出して調べる。

CPからのメッセージ

以下の4つがある。

1. 問い合わせメッセージ (where)
   あるLOTOSプロセスがどのマシン上にあるか問い合わせるメッセージ。そのプロセスが自分の記録にあれば, それを情報メッセージ (info) (次項) として木上でつながるすべてのCPに送る。なければ木上の他のCPに同じ問い合わせメッセージを送る。
2. 情報メッセージ (info)
   あるLOTOSプロセスがどのマシン上にあるかの情報。情報を記憶して木上の他のCPに同じメッセージを送る。
3. プロセス生成メッセージ (instantiate)
   プロセスの生成を要求するメッセージ。処理はLOTOSプロセスからの場合に準ずる。
4. 転送メッセージ
   どこかのLOTOSプロセスに送られるべきメッセージ。処理はLOTOSプロセスからの場合に準ずる。

### 3.4.2 同期木の形成法のための変更

同期木の形成は文献[4]に基づいたアルゴリズムを用いるが, 本実装法では各ゲート毎に木を形成しなければならないので, オリジナルのアルゴリズムに以下の点を追加する。

1. すべてのメッセージには, そのメッセージで形成すべき同期木のゲート名をつける
2. 各プロセスはメッセージに付加されたゲート名によってメッセージを分別し, 各ゲート毎の同期木の生成アルゴリズムを並列に実行する。

### 3.5 システムの実現

以上に示した実装法にしたがって，LOTOS マルチランデブを実現するためのシステムをネットワーク環境上に実現した。システムは，イーサネットを介して接続された複数の UNIX マシン上で，C 言語を用いて実現した。

また，このシステムへのインタフェースは C 言語のライブラリとして用意されるので，文献 [5] による LOTOS-C トランスレータによって生成されるプログラムから利用可能である。

## 4 応用例

以上のように実装したマルチランデブを用いて，簡単なシステムを実現する。

### 4.1 会議室予約システム

ある一日の会議室の使用権の交渉を，LOTOS マルチランデブを用いて記述する。

前提として以下のような状況を考える。

1. 会議室は一つしかない
2. 予約は一時間を単位とする
3. 各研究室は任意の連続する時間 (時間帯) を予約できる
4. 各研究室は一日に高々一つの時間帯しか獲得できない
5. 研究室間に優先順位はない

この前提の下では，会議室の予約の交渉は図 5のように表すことができる。

このとき，3 章までに実現したシステムを用いたこのシステムの実現法は以下のようになる。

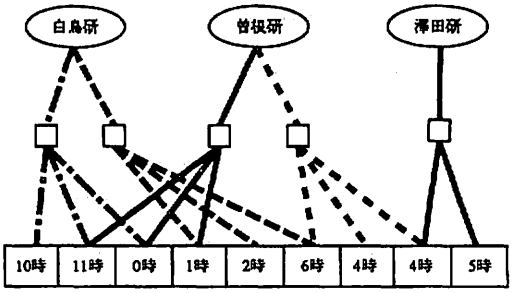


図 5: 会議室の予約

まず，時間帯の可能なすべての組合せ (10 時から 12 時，3 時から 4 時など) に対応したゲートを考える。

各時間枠 (10 時台，3 時台など) 毎に LOTOS プロセスを記述する。各時間枠に対応するプロセスは，その時間枠が関係するすべてのゲート間のチョイスとして表す。

各研究室の要求を，使用したい時間帯に対応したゲートのチョイスのみからなる LOTOS プロセスとして記述する。

トップレベルのプロセスは，これらすべてのプロセスの並列として記述する。

以上を一つに集め，LOTOS-C トランスレータを用いて C 言語に変換し，コンパイルする。このとき，作成したマルチランデブ用ライブラリをリンクする。

生成されたプログラムを実行し，結果を得る。

### 4.2 考察

以上のように 3 章までに実現したシステムをそのまま用いた方法では，各研究室の要求をいったん一箇所に集め，その後に変換 / 実行を行なっている (図 6(a))。しかしこの

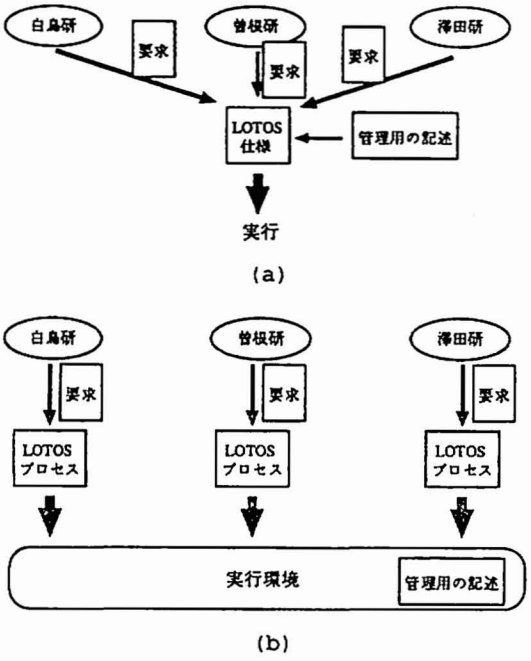


図 6: 予約システムの構成

ままでは, ネットワーク環境上でマルチランデブを用いている意味があまりない。各研究室のマシンはネットワーク上に分散しているので, システムもこれを用いた分散的なものであるのが望ましい (図 6(b))。

このような場合にネットワーク上でのマルチランデブを有効に使う方法として, 次のような方法が考えられる。

### 分散的な実現法

要求を記述したプロセスは, 各研究室でそれぞれ変換 / 実行する。また, 各時間枠に対応したプロセスはあらかじめネットワーク上で動作させておく。

これらに加えて, これらのプロセスの間の関係を定義する LOTOS のトップレベルのプロセスにあたるものを用意しておくことによって, 前述のシステムと同じことを分散的に行なうことができる。

このような実現法に対応するため, 以下のものを実現した。

1. ネットワーク上で, あらかじめ指定されたプロセス間のマルチランデブを解決する分散環境
2. トップレベルでない LOTOS のプロセスのみでも C 言語に変換することができるトランスレータ

1. はシステムを運用するための基盤, 2. はそれにアクセスするためのインタフェースと考えることもできる。これらを用いることによって, 上記のような分散的なシステムを実現することができる。

## 5 あとがき

本研究では, LOTOS マルチランデブをネットワーク環境上で実現する方法を与え, これを実際にネットワークシステム上に実現した。またこれが種々の分散サービスに用いられる可能性をその一例を実現することで示し, さらにその場合の分散環境の実現について述べた。

今後は, LOTOS にとらわれないマルチランデブの実現法や, その実現のための分散環境の強化を考えていきたい。

## 参考文献

[1] 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).

[2] ISO, "Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour," ISO 8807, (1989-2).

[3] R. Sisto, L. Ciminiera and A. Valenzano : "A Protocol for Multirendezvous of LOTOS Processes," IEEE Trans. on Computer, **40**, 1, pp. 437-447, (1991-4).

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

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

[6] G. v. Bochmann, Q. Gao and C. Wu : "On the Distributed Implementation of LOTOS," in Proc. of FORTE'89, Vancouver, (1989-12).