マルチメディア通信と分散処理 102－7
（2001．3．21）

並行モバイルエージェント間でのマルチランデブチャネルの動的設定が記述可能な言語とその実装

寺島芳樹[1] 安本慶一[2] 中田明夫[1] 東野輝夫[1] 谷口健一[1]

[1] 大阪大学大学院基礎工学研究科情報数理系専攻 [2] 滋賀大学経済学部情報管理学科
{terasima,nakata,higashino,taniguchi}@ics.es.osaka-u.ac.jp
yasumoto@biwako.shiga-u.ac.jp,

あらまし 本論文では，無線環境における複数の移動可能エージェント間での動的なマルチランデブチャネル通信が記述可能な言語 LOTOS/M とその実装法を提案する．LOTOS/M では，システムは互いに独立なエージェントの集合として与えられ，同期募集側，参加側の２つのエージェントが互いの無線範囲に入るとチャネル生成が可能になる．複数のエージェントが同チャネルで段階的に結びつくことにより，無線範囲を越えたマルチランデブが可能となる．あるエージェントが無線範囲から離脱した場合，残りのエージェント群との同期関係を解消し，以後互いに独立に動作できる．エージェント間のマルチランデブの実行制御は，エージェントの結合，離脱に伴い変化する同期木（同期関係を表す２分木）を関連エージェント間で管理・維持し，木の構造に沿って同期要求メッセージを集め処理することで行う．実験により，幾つかの典型モバイルシステムが LOTOS/M で容易に記述でき，試作したコンパイラで生成したコードが無線 LAN 上で実用上問題ない速度で動作することを確かめた．

An Extended LOTOS Language for Wireless Mobile Agents with Dynamic Multi-way Synchronization and its Implementation

Yoshiki Terashima[1], Keiichi Yasumoto[2], Akio Nakata[1],
Teruo Higashino[1] and Kenichi Taniguchi[1]
[1] Dept. of Informatics and Mathematical Science, Osaka University
[2] Faculty of Economics, Shiga University

Abstract In this paper, we propose (1) a new language called LOTOS/M which enables dynamic multi-way synchronization among multiple mobile agents in wireless networks, and (2) its implementation method. In LOTOS/M, a system specification is given by multiple mobile agents. When an agent is seeking synchronization through some gates and another agent wants to participate in the synchronization in their common radio range, they can be combined via the offered gates. Multiple agents can be combined via the same gates to enable multi-way synchronization. When an agent moves out of the ranges, the synchronization relation is cleared and it can proceed independently. In our implementation method, we have agents keep the latest tree representing the synchronization relation among them and implement multi-way synchronization by collecting request messages from those agents along the tree. Based on the method, we have implemented a LOTOS/M compiler. Through some experiments, we have confirmed that (1) typical wireless mobile systems can be easily specified in LOTOS/M, and (2) the generated code can run on a wireless LAN at reasonable speed.

1 まえがき

近年の携帯端末および無線通信環境の発展・普及により，移動可能な端末 (モバイルエージェント) 間の無線通信を伴うモバイルアプリケーションが注目されており，ロケーションアウェアシステム (エージェントの現在の地理的位置に応じて，異なったサービスを受けることが可能なシステム [4]) や，動的に構築したアドホックネットワーク [9] 上での電子会議システムなどが考案されている．こういった無線モバイルアプリケーションでは，インターネットのような相手を特定した通信のみならず，エージェントの自由な移動に応じて，たまたま通信可能範囲に入った相手と通信を行う機能や，無線範囲にいない通信相手との間に幾つかの中間ノードを介して通信路を動的に確保する機能 [9] などが必要になるため，これらの機能が容易に取り扱える言語および開発環境が望まれている．

形式記述言語 LOTOS[5] は，プロセス間の選択，並列，割込などの高度な制御機構に加え，複数並行プロセス間でイベントを同期し実行するためのマルチランデブと呼ばれる機構を持つ．マルチランデブの利用により，ブロードキャストや資源への排他制御などを含む分散システムを容易に記述できることが知られており，モバイルシステムの設計開発においても，これらの機能の利用が望ましいと思われる．しかし，従来の LOTOS では，エージェント同士が互いの無線範囲に入り通信可能状態となったことの判定や，たまたま出会ったエージェント間で動的にマルチランデブによる通信を行うことを記述する方法がないため，モバイルシステムの記述に適用するのは困難であった．一方，プロセス代数に，互いに交信可能な範囲のエージェント間での動的なチャネル生成機能を追加した π 計算 [1] や，同様な機能を LOTOS に取り込んだ M-LOTOS[3] などでは，エージェント群が互いの通信可能範囲に入った時にのみ通信を行う，という動作をうまく記述できるが，LOTOS の特徴であるマルチランデブがエージェント間に指定できない．

本論文では，複数モバイルエージェント間での動

的なマルチランデブの設定・通信が記述可能な言語LOTOS/M とその実装法を提案する．LOTOS/M では，モバイルシステムは，互いに独立に動作するエージェントの集合として与えられる．任意の2つのエージェントが(無線範囲内に移動したなどで)互いに通信可能な状態になった時のみチャネル(ゲート)生成を可能とする構文を導入した．この際，あるチャネルで結合したエージェント群を新たな一つのエージェントとしてとらえ，別のエージェントを同じチャネルに段階的に結合するためのセマンティクスを定義した．これにより，無線範囲を越えたエージェント群の間のマルチランデブが可能になる．また，あるエージェントが移動などによって無線範囲から離脱した場合，結合している他のエージェント群との間の同期関係を解消し，以後独立に動作できるようなセマンティクスを与えた．

提案する実装法では，エージェントの結合，離脱に伴い変化する同期木（同期関係を表す2分木）を関連エージェント間で管理・維持し，木の構造に沿って同期要求メッセージを集め処理することで，エージェント間のマルチランデブの実行制御を行う．エージェントの離脱は，ポーリングを行うことで検出し，適切な処理を行うことで対処した．なお，エージェント内の他の動作については，我々が開発したLOTOSコンパイラの手法 [6] に基づき実装する．

提案手法に基づき JAVA コードを生成するコンパイラを作成した．提案したLOTOS/M の機能を用いて，ロケーションアウェアシステムや，無線アドホックネットワーク上の経路探索プロトコルなどが容易に記述できること，自動生成したコードについて，無線 LAN 上でのマルチランデブチャネルの生成，消滅およびデータ転送が実用的な時間で実行できたこと，などの実験結果から，提案手法がモバイルシステムの設計・開発に有用であることを確かめた．

2 LOTOS/M 言語の提案

2.1 LOTOS とモバイルシステム

LOTOS では，システムの仕様をいくつかのプロセスからなる並行プロセスとして記述する．各プロセスの動作は**動作式**と呼ばれ，プロセス外部（環境）から観測可能なアクションである**イベント**，観測不可能な**内部イベント**，あるいはプロセス呼び出しの実行系列として定義される．ここでイベントは，環境との相互作用（データの入出力）であり，ゲートと呼ばれる作用点で発生する．イベント間の実行順序を指定するため，連接 $(a;B)$，選択 $(B1[]B2)$，同期並列 $(B1|[G]|B2)$，非同期並列 $(B1|||B2)$，割込み $(B1[>\ B2)$，逐次 $(B1\ >>\ B2)$ などのオペレータが任意の部分動作式間に指定される．特に，同期並列オペレータを使用することにより，複数のプロセスが指定されたゲート上のイベントを同時に実行しデータ交換を行なう，といった動作を記述することができる（**マルチランデブ**と呼ばれる）．

分散システムの複数ノード間にマルチランデブを指定することで，データのブロードキャストや資源への排他制御など複雑な機構を含むシステムが簡潔に記述でき，さらには制約指向記述スタイル [7] による，ノード間の動作制約の段階的な追加などシステムの設計・保守上有用な機能が利用できることが知られており [10]，モバイルシステムのエージェント間においてもマルチランデブが使えることが望まれる．

図 1: 移動に伴う通信可能なエージェントの変化

LOTOS でモバイルシステムを記述する際の問題点

無線モバイルシステムでは，図 1 に示すようなエージェントの移動により，互いに通信可能なエージェントの組合せが動的に変化するため，LOTOS でモデル化する際には以下の問題が生じる．

(1) エージェント同士が互いの無線範囲に入り通信可能となっているどうかの判定をうまく表す手段がない．

(2) 多数のエージェントのうち，どのエージェントが互いに無線範囲に集まって通信を行うのか，また幾つのエージェントの間でマルチランデブを行うのか，その組み合わせは無数にあり，LOTOS では基本的にそれらすべての組み合わせをあらかじめ記述しておく必要がある．

文献 [8] では，従来 LOTOS により，携帯端末の基地局に対するローミングを含む移動体電話システムを記述する試みがなされているが，携帯端末 (エージェント) のいずれかと基地局のいずれかが 1 対 1 通信を行うような動作に限定し，エージェントの ID と基地局の ID をあるゲートを介して動的に交換するなどの方法で，上述の問題を一部回避している．

しかし，根本的な解決のためには，エージェントの移動による通信可能状態となったことの検知および，その際に動的にマルチランデブチャネルを生成・解放し，そのチャネルを用いて，複数エージェント間でのマルチランデブを行えるようにするための機能が必要であると考える．

2.2 LOTOS/M 言語で拡張する機能

このため，エージェントの接近，移動に伴って動的にチャネルを生成・解放し，そのチャネルを用いて，複数エージェント間でのマルチランデブを行えるようにするための機能を拡張した LOTOS/M を定義する．

LOTOS/M では，表 1 に示すように，モバイルシステム全体を，`A := A1 | A2 | ... | An` のように，互いにどのように通信するのかがあらかじめ定まっていないエージェント群の集合として与える．通信範囲内にあるかどうかの判断およびチャネルの生成は，ゲート群 G での同期の募集 (sync !G!sid in B1 endsync), 募集に対する応答 (sync ?H?sid in B2 endsync) により行うことができる．ここで G, H はゲートリスト，sid はエージェントの組が結合する際の同期関係に対する ID を表す (ID の送受信の省略時は，適当な ID が自動的に付加されると仮定)．あ

表 1: 新たに追加された構文と機能

構文	意味
A1 | A2 | ... | An	同期関係があらかじめ指定されていないエージェントの並行動作
agent A[G](E):= B endagent	エージェントの動作式の宣言
<B>	動作式 B をエージェントとして起動
sync !G!sid in B1 endsync	ゲート群 G での同期の募集
sync ?H?sid[条件] in B1 endsync	他の同期募集への参加，参加した際募集側のゲート群を G に代入
disc(sid)	ID が sid の同期関係を解消
g!h; ... |[g]| g?i:gate; ...	ゲート名の送受信
g!P; ... |[g]| g?Q:process; ...	動作式の送受信

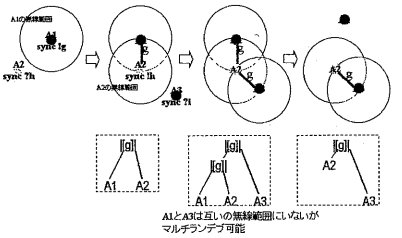

図 2: エージェント間のチャネルの生成・消滅

るエージェントの組のうち，一方が sync !G を，もう一方が sync ?H を実行可能な状態にあり，G, H に含まれるゲート数が同じであり，かつ，それらのエージェントが互いに通信可能な範囲にいる時，それらは，同期関係 G で結合することができ，以後，結合が解除されるまで，$B1|[G]|B2[H/G]$ のように振舞う (B2[H/G] は動作式 B2 に出現する H のゲート群を G の対応するゲート群で置き換えた動作式)．sync ?H[条件] のように「条件」が指定された場合，この条件が成立する相手としか結合できない．指定可能な条件は，受け取ったゲート名の比較 (H=F)，生成された同期関係に対する ID の比較 (sid= xxx)，あるいは，これらの論理結合に限る．なお，エージェント間のマルチランデブに使用するゲートは，hide オペレータにより新たに作成し，使用するものとする．

結合したエージェントは新たな一つのエージェントとして扱い，さらなる sync !g オペレータを実行することで，他のエージェントと段階的に結合できる．3 つのエージェント A1, A2, A3 が段階的に結合する様子を図 2 に示す．

LOTOS/M では，高階 π 計算 [2] にならい，従来 LOTOS で扱えるデータタイプ以外に，ゲート名および動作式自体もデータとして，マルチランデブにより送受信可能なよう拡張した．これらを用いたアプリケーションの記述例は 2.4 節で示す．

2.3 LOTOS/M のセマンティクス

LOTOS/M で拡張された構文 sync，disc(明示的に切断する以外に，通信可能範囲から離脱した場合も含む) について，以下のような意味定義を行い，そのための推論規則を定義した (表 2)．

表 3: ロケーションアウェアシステムの記述例

```
specification LocationAware: noexit
behavior
  A | <S1 ||| S2 ||| S3>
where
  agent A : exit:=
    sync ?{h} in h?Q:process; Q
    endsync >> A
  endagent
  process S1 :noexit:=
   hide g in
    sync !{g} in g!P1; exit
    endsync >> S1
    where
    process P1[...] :noexit:=
      ...
    endproc
  endproc
  ...
endproc
```

(ここで，S2, S3 は S1 と同様であり，エージェントに送信するサービスの内容であるプロセス P1, P2, P3 の内容は省略している)

• 各 sync 動作の実行に対して id を発行し，id に対応する disc 動作ができるようにする．
• disc 動作後のエージェントは，動作式のいかなる部分式として動作していても，独立したエージェントに戻す．

結合時　ゲート群 G での募集エージェント，ゲート群 X での参加エージェントの組が存在するとき，推論規則 Agent-Join により，これらは結合して G での同期関係を持つことができる．X はゲートが格納される変数であり，結合後は G と等価になる．この同期関係には id が付随され，次の離脱時の推論規則において，指定 id を持つ同期関係が解消される．

離脱時　離脱時には，同期関係が指定されている動作式に対し，id に対応する離脱対象の並列演算子が見つかったか否かに応じて推論規則を適用していく．

2.4 モバイルシステムの記述例

2.4.1 ロケーションアウェアシステム

ロケーションアウェアシステム [4] とは，携帯端末を通し，自分が今いる地理的位置に応じたサービスが利用可能なシステムであり，美術館で部屋ごとに異なる美術品の説明を行う，などが例として挙げられる．このようなシステムの実現のためには，位置の検知，およびその位置ごとに異なる動作を行うための機構が必要である．LOTOS/M では，現在の無線範囲内で同期を募集する機構を利用して位置の検出を行い，また，動作式の送受信の機構を用いて，同期を募集する各基地局で異なったコードを用意しておくことで，その位置に応じたコードを受信，実行することで，これらの機能を含むシステムを容易に実現することができる．

記述例を表 3 に示す．基地局と携帯端末間のチャネル用として g を用意する．ある無線範囲に入りチャネルが成立した場合，g を通してそれぞれ基地局ごとに異なるデータや，何らかのサービスを行うプロセス (P1, ...) を受信し，携帯端末にてそれが処理，実行される．

この例のように動作式の送受信を行うことにより，記憶容量の少ない小型携帯端末にオンデマンドでコードをダウンロードし実行できる．

表 2: LOTOS/M におけるエージェント結合，離脱のための推論規則

Agent-Join

$$\frac{A_i \overset{sync!G}{\longrightarrow} A_i', \quad A_j \overset{sync?X}{\longrightarrow} A_j', \quad G=\{g_1,...,g_k\}, \quad X=\{x_1,...,x_k\}}{|\{A_1,...,A_k\} \overset{sync!G}{\longrightarrow} ((A_i'|[G]|_{\{sid\}}A_j'[X/G])|||disc(sid);exit)|(|\{A_1,...,A_k\}-\{A_i,A_j\})} \quad (1)$$

$|\{A_1,...,A_k\}$ は $A_1|A_2|,...,|A_k$ を表し，$B[X/G]$ は B の仮ゲートパラメータ群 $X=\{x_1,...,x_k\}$ の全ての自由な出現を対応する実ゲートパラメータ群 $G=\{g_1,...,g_k\}$ で置き換えることを表す．また，sync オペレータに条件が設定されている場合は省略するが，推論規則の前提条件に指定された条件を付け加えることで対処できる．

Agent-Leave-1

$$\frac{< B_1 > \overset{disc(sid)}{\longrightarrow} B_1'}{[< B_1 > |[G]|_{sid} B_2 \overset{disc(sid)}{\longrightarrow} B_2, B_1', matched]} \quad (2)$$

Agent-Leave-2

$$\frac{< B_1 > \overset{disc(sid)}{\longrightarrow} B_1', \quad sid <> sid'}{[< B_1 > |[G]|_{sid'} B_2 \overset{disc(sid)}{\longrightarrow} B_2, B_1', unmatched]} \quad (3)$$

Agent-Leave-3

$$\frac{[B_1 \overset{disc(sid)}{\longrightarrow} B_1', B_3, flag], \quad sid <> sid'}{[B_1|[G]|_{sid'} B_2 \overset{disc(sid)}{\longrightarrow} B_1'|[G]|_{sid'} B_2, B_3, flag]} \quad (4)$$

Agent-Leave-4

$$\frac{[B_1 \overset{disc(sid)}{\longrightarrow} B_1', B_3, unmatched]}{[B_1|[G]|_{sid} B_2 \overset{disc(sid)}{\longrightarrow} B_1'|[G]|_{sid} B_2, B_3, matched]} \quad (5)$$

Agent-Leave-5

$$\frac{[B \overset{disc(sid)}{\longrightarrow} B', B'', matched]}{B \overset{disc(sid)}{\longrightarrow} B'|B''} \quad (6)$$

補助ターム $[B \overset{disc(sid)}{\longrightarrow} B', B'', flag]$ は，$B'', flag$ という付加情報がついている以外は遷移関係 $B \overset{disc(sid)}{\longrightarrow} B'$ と全く同じである．B" は離脱するエージェントの動作式、$flag \in \{matched, unmatched\}$ は sid に対応する離脱対象の並列演算子が見つかったか否かを表すフラグである．

2.4.2 アドホックネットワークでの経路探索

無線アドホックネットワークでは，幾つかの途中エージェントを介した無線範囲の通信を繰り返すことで，無線範囲を超えた任意のエージェント間で互いにデータ交換できる．ここでは，Dynamic Source Routing [9] に基づき，あるエージェントから別のエージェントへの経路を見つけるプロトコルを LOTOS/M で記述する．

このプロトコルでは，図3に示すように，あるエージェント (ソースエージェント) が他のエージェントへの経路を求めたいとき，その要求を無線範囲内にブロードキャストし，それを受け取ったエージェントは，その要求を再ブロードキャストする．このようなフラッディングを目的エージェントに到達するまで繰り返す．経路探索要求メッセージには，ソースから現在のエージェントに到達するまでの経路を含める（各中間エージェントは要求を転送する際に，自分の ID を変数 route_record の末尾に加える）．目的エージェントに到達した時，そのメッセージには，ソースからの経路がルートレコードに記録されているため，その情報を経路の逆順にソースエージェントに返信する．

エージェントの動作は以下のいずれかである．
(1) ソースでの，目的エージェントの ID の入力 (u?dest_id) と経路探索要求 (a!Search!dest_id !{my_id}) のブロードキャスト
(2) 中間ノードでの，経路探索要求の受け取りとその要求の転送 (再ブロードキャスト)
(3) 目的地での，経路探索要求の受け取りと発見経路の返信 (4) 発見経路返信の受け取りとその転送 (5) ソースでの発見経路返信の受け取りとユーザへの提供 (u!route_to_dest)

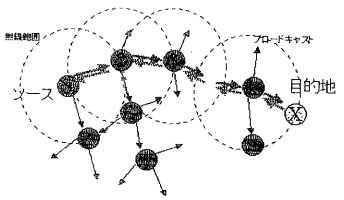

図 3: アドホックネットワークでの経路探索

エージェントの無線範囲へのブロードキャストは，ゲート b での同期募集を一定時間繰り返すことによって，無線範囲の複数エージェントの同期への参加を募った後，マルチランデブにより一度に送信する．ブロードキャストを行うサブプロセス Broadcast の記述例を以下に示す．パラメタとして，目的地の ID(dest_id)，現ノードまでに通る中間ノードの ID のリスト (route_record) を使用している (各パラメタの型は省略)．

```
process Broadcast(dest_id, route_record): exit:=
hide b in
  sync !{b}?sid in
     Broadcast(dest_id, route_record, my_id)
  [] i (* 募集開始から一定時間経過 *);
     b!Search!dest_id!route_record; exit
  endsync
endproc
```

エージェントの動作のうち，例えば，(2) は，現在の経路情報に自分が含まれない時のみ (not(included (route_record,my_id)))，Search メッセージを転送するように記述する．また，(3), (4), (5) について

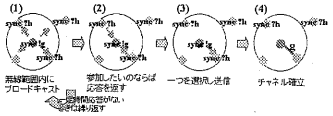

図 4: 同期募集からチャネル確立まで

は，ソースエージェントへの返送パス上の次のノード (エージェント) に対してのみ転送を行いたいため，sync オペレータの ID として，そのノードの ID を指定する (`sync !b!last(route_record)`, `sync ?{a}?sid[sid=my_id]`).

文献 [11] に，エージェント DSR の記述例の全体を与えている．

3 LOTOS/M 言語の実装法

2 章で述べたセマンティクスに基づき，エージェントは段階的に結び付くため，その同期の関係はエージェントを葉，同期オペレータを節とする 2 分木の形をとる同期木で表すことができる．これは LOTOS の動作式に相当するため，通常の LOTOS の同期判定と同じく，葉から根に向かう各節において，同期条件を判定することにより，実行可能であるかを計算できる．この操作を根まで繰り返すことにより，最終的にどのエージェント間でどのイベントの組合せが同期実行可能であるかが判定できる．

3.1 エージェントの結合とチャネル設定

同期を募集するエージェント (募集側) と，参加要求を行うエージェント (参加側) が互いの無線範囲内にいるときその間にチャネルを設定する．

範囲内かどうかの判定は，図 4 に示すように，次の (1)-(4) で行う．(1) 募集側はその無線範囲に同期募集メッセージをブロードキャストし，(2) 参加側の応答があるまでそれを繰り返す．(3) 複数応答を受けた場合は一つを選択し，承諾メッセージを返すことで，(4) チャネルを確立し，それぞれ同期木の設定を行う．

チャネルを設定したエージェント間の同期実行判定のため，ここでは募集側を責任ノードと決め，どのエージェントとどのゲートで結び付いたかといった同期関係を保持させる．参加側は責任ノードである募集側に同期実行要求を送るよう設定する．図 5 に示すように，複数エージェントが階層的に結合する場合には，結合によって生じる各節の責任ノードを決めると共に，各責任ノードの子ノードと親ノードの関係 (送信先等) を設定することにより，エージェント間の同期関係を維持する．

3.2 マルチランデブ判定の例

A1～A4 のエージェントが結合して図 5 のように責任ノードが設定された状況での，g のマルチランデブ実行判定の様子を述べる．まず A2 では，実行したいイベント g?x について，その責任ノードである A1 に実行要求を送信する．A1 では，自身の実行要求 g?x と，A2 からの実行要求 g?x を調べ，同期可能かを判定し，その結果 g?x を得て，これを親ノードである A2 に転送する．A2 では，A1 からの結果と A3 からの要求 g!1 を受け，判定結果 g!1 を

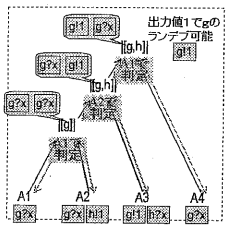

図 5: 実行可能性の判定

得てさらに親ノード A1 に転送する．同様に A1 で A2，A4 からの要求を判定した結果 g!1 を得る．この段階での親ノードは存在しないため，出力値 1 で g のランデブが可能と判定でき，この結果が各エージェントに伝えられる．

図 5 では，責任ノードは葉から見て A1，A2，A1 となっているが，これを A1，A1，A1 などと極力連続するように責任ノードの入れ替えを行うことができれば，責任ノード間の通信回数を削減できる．

3.3 エージェントの動的な離脱への対処

離脱の状態を次のように定義する．

- 直接同期関係で結合しているエージェント (一対一) 同士が通信できなくなったとき
- 同期木のある同期オペレータについて，左側部分木のエージェントと右側部分木のエージェントのどの組み合わせでも通信できないとき

この際，親責任ノードと通信できない側のエージェント群を離脱したものとみなす．

(1) ランデブ判定中の離脱への対処　各エージェントの実行要求は異なる時刻に発生するため，その判定結果を受け取るまでに間がある．そのため，(i) 根ノードであるランデブの実行可能性の判定後，その判定結果を送信する前に，そのランデブの参加メンバであるエージェントが無線範囲内にいるかどうかを調べるためのポーリング信号を送信する．(ii) 各エージェントがそれに応答し，(iii) 根ノードで全関連エージェントからの応答を確認したときのみ，判定結果を通知する．一つでも返事がない場合，返事がなかったエージェントからの実行要求を削除し，ランデブの判定をやりなおす．

ここで，全てのエージェントがポーリング信号を送信してから結果を受信するまでの間，互いの無線範囲から離脱しないと仮定している．

(2) 離脱に伴う同期木の変更

責任ノード以外のエージェントの離脱　図 5 で A3 が離脱した場合を考える．A3 の責任ノードである A2 は，ポーリングによりその事実を知る．この時，(1) A2 における，A3 とのための責任ノード (図 5 の中間の I[g,h]I) の削除，および，(2) この責任ノードの左側の部分木 (I[g]I を根とする木) の親責任ノード (図 5 の最上位の I[g,h]I) への連結が必要となる．

エージェント A2 は (1) の処理を行った後，(2) のために，I[g]I の責任ノード (A1) に，親責任ノードを最上位の I[g,h]I (A1) に変更するよう通知する．

この場合，変更先が同じノード (A1) なので，変

	募集側	参加側
1000ms 待つ場合	1100	1061
待たない場合	2.61	2.72

表 4: チャネル生成時間 (ms)

エージェント数	A1(責任ノード)	A2	A3	A4
2	5.2	4.9		
3	7.0	6.6	6.6	
4	7.6	7.7	7.7	7.6

表 5: マルチランデブの実行時間 (ms)

エージェント数	転送容量
2	550Kbps
3	548Kbps
4	547Kbps

表 6: データ転送速度

更後の責任ノード間の通信は問題ない．もし最上位の|[g,h]|の責任ノードがA4であり，|[g]|の責任ノードA1との物理的な通信が不可能な場合，|[g]|以下の部分木は離脱したものとみなされる．

責任ノードの離脱　図5において，A2が離脱したとき，最上位の|[g,h]|の責任ノードA1は，ポーリングにより子責任ノードA2が離脱した事実を知る．この際，|[g]|を根とする部分木を最上位の|[g,h]|に連結する必要がある．

このため，提案手法では，各責任ノードに，右部分木，左部分木ごとに現在含まれるノードの集合を記憶させ，その情報をもとに，新たに親責任ノードに連結するための子ノードを無線範囲のブロードキャストにより募ることとした．この例では，最上位の|[g,h]|の責任ノードであるA1は，新たにA1へ連結するための責任ノードを募集するためのメッセージ（A2が離脱したという事実も含む）を応答すべき子ノードの集合 (A1, A2) を指定してブロードキャストする．子ノードのうち一つがそれに応答し，部分木の責任ノードになる．どのノードからも応答がない場合は，部分木全体が離脱したものとみなす．

4　性能評価

3 章の方法に基づき，LOTOS/M 仕様を JAVA コードに変換するコンパイラを作成した．目的コードについて，エージェント間の (1) チャネル生成，(2) マルチランデブによるイベントの同期実行，(3) マルチランデブによるデータ転送，の時間を測定し実行効率を調べた (無線 LAN 環境，11Mbps)．

実験 1　まず，1 回あたりのエージェント間のチャネル生成時間を測定した．ここでは互いに無線範囲におり，sync 実行直後に相手側から応答を受けられる状態とする．また，募集側がブロードキャスト後一定時間 (ここでは 1 秒) 応答を待つ場合と，待たない場合とを測定した．結果を表 4 に示す．

実験 2　次に，エージェント数増加時のマルチランデブ実行時間として，イベントの同期実行要求後，その判定結果を受けて実行するまでの時間を測定した．ここでは責任ノードの設定による差がでないようにするため，最初に同期を募集したエージェント A1 を 2 分木の全ての節での責任ノードとした．結果を表 5 に示す．エージェント数の増加に伴い実行時間が長くなっているが，これは，A1 に同期要求が集まるまでの時間差によるものと思われる．

実験 3　最後に，エージェント間のマルチランデブによるデータ転送速度の測定のため，実験 1 と同じ条件で，1000 バイトのパケットを送受信させた場合の結果を表 6 に示す．

5　あとがき

本論文では，無線モバイルアプリケーションの記述・実装に適した言語とその実装法を提案した．

提案した LOTOS/M では，たまたま出会ったエージェント間で動的にチャネルを生成し，マルチランデブ通信を行ったり，無線範囲からの離脱に対して動的にマルチランデブを構成メンバを変更するような動作が記述できる．幾つかのアプリケーションの記述，実装実験から，提案手法が無線モバイルアプリケーションの記述・実装に十分適用可能であることを確認した．

携帯端末へ動画などの実時間データを転送する際には，LOTOS/M 仕様の各動作に時間制約が指定できることが必要である．そのための言語の拡張および与えられた仕様を時間制約どおりに動作させるための実装法を考案することが今後の課題の一つである．

参考文献

[1] Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes: Parts I & II, *Information and Computation 100*, pp. 1– 77 (1992).

[2] Sangiorgi, D.: From π-calculus to Higher-Order π-calculus — and back, *Proc. of Theory and Practice of Software Development Lecture Notes in Computer Science (TAPSOFT'93)*, Vol. 668, pp. 151 – 166 (1993).

[3] Najm, E., Stefani, J.B. and Fevrier, A.: Towards a Mobile LOTOS, *Proc. of 8th IFIP Intl. Conf. on Formal Description Techniques (FORTE'95)* (1995).

[4] Hodes, T.D., Katz, R.H., Schreiber, E.S. and Rowe, L.: Composable Ad-hoc Mobile Services for Universal Interaction, *Proc. of Mobile Computing and Networking(MOBICOM'97)* (1997).

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

[6] 安本慶一, 安倍広多, 後藤和裕, 東野輝夫, 松浦敏雄, 谷口健一：“マルチスレッド化目的コードを生成する LOTOS コンパイラの実現”, 情処論誌, 39 (2): 566 – 575 (1998).

[7] Vissers, C. A., Scollo, G. and Sinderen, M. v.: "Architecture and Specification Style in Formal Descriptions of Distributed Systems", *Proc. 8th Int. Conf. on Protocol Specification, Testing, and Verification (PSTV'88)*: pp. 189 – 204 (1988).

[8] Randall Tuok, Luigi Logrippo: "Formal Spacification and Use Case Generation for a Mobile Telephony System",*Computer Networks*, Vol. 30, No. 11, pp. 1045-1063 (1998).

[9] Johnson, D. B., Maltz, D. A., Hu., Y. C. and Jetcheva, J. G. : The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, *IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr04.txt* (2000).

[10] 辰本比呂記, 後藤和裕, 安本慶一, 東野輝夫, 安倍広多, 松浦敏雄, 谷口健一：分散環境での LOTOS 仕様の実現とその評価, 情報処理学会論文誌, 第 40 巻, 第 1 号, pp. 333 – 342 (1999).

[11] 寺島芳樹: モバイルエージェント間マルチランデブチャネルの動的生成を可能にする拡張 LOTOS 言語とその実装, 大阪大学大学院基礎工学研究科情報数理系専攻修士学位論文 (2001).