論　文

分散移動システムのための前後関係保存放送プロトコル

大堀　力[†*]　井上美智子[†]　増澤　利光[†]　藤原　秀雄[†]

A Causal Broadcast Protocol for Distributed Mobile Systems

Chikara OHORI[†*], Michiko INOUE[†], Toshimitsu MASUZAWA[†], and Hideo FUJIWARA[†]

あらまし　本論文では，分散移動システムのための前後関係保存放送プロトコルを提案する．移動計算機は一般に不特定多数であり，計算能力，通信能力が固定計算機に比べて著しく劣っているため，複雑度が移動計算機数にできるだけ依存せず，移動計算機の実行する計算量と通信量の小さいアルゴリズムが求められる．また，移動計算機の移動に伴うハンドオフへ対処するために行われる処理のより小さいアルゴリズムが求められる．本論文は移動計算機と移動支援局の階層構造に着目した，効率の良い前後関係保存放送アルゴリズムを提案する．提案手法は，メッセージオーバヘッド（各メッセージに追加する情報量）が移動計算機数に依存していない．更に，既知の手法に比べて，メッセージオーバヘッドが小さく，移動端末移動時のハンドオフに対処するための遅延時間とメッセージ数が小さい点で優れている．

キーワード　分散アルゴリズム，移動通信，放送，前後関係

1. まえがき

小型計算機の高性能化と無線通信技術の進歩によって，移動計算機が無線通信を利用してネットワークに接続することが可能になった．分散移動システムは，固定ネットワークに移動計算機（mobile host, MH）を付加したシステムである．MH は計算の実行中に移動することができ，無線機能をもつ固定計算機と無線通信が可能である．無線機能をもつ固定計算機を特に移動支援局（mobile support station, MSS）と呼び，各 MSS の地理的，あるいは，論理的な無線通信可能領域をその MSS のセルと呼ぶ．ある MSS のセル内に存在する MH が別の MSS のセルに移動すると，移動元の MSS と MH との間の無線通信チャネルが切断され，移動先の MSS と MH との間に無線通信チャネルが開かれる．この移動した MH の無線通信チャネルの切換動作を MH のハンドオフと呼ぶ．また，移動元の MSS から移動先の MSS へ MH がハンドオフされるともいう．

MH を含まない分散システムで問題を解くプロトコルについては，これまで数多くの研究がなされている [2]〜[4], [6]．しかし，これらはハンドオフに伴うネットワークの形状の動的な変化に対応できず，分散移動システムに適用できないことが多い．そのため，分散移動システムのためのプロトコルの設計が必要であり，さまざまな研究が行われている [7]〜[9]．更に，分散移動システム上のプロトコルの設計には，次の三つの目標を考慮しなければならない．

1. MSS に比べて MH は性能（記憶の容量と信頼性，処理能力，使用可能電力量など）が低い．更に，MH-MSS 間の無線通信チャネルは，MSS 間の有線通信チャネルに比べて通信帯域が著しく小さい．よって，MH で記憶する情報量，MH の計算量，MH の通信量をできるだけ小さくする必要がある．

2. システム内に存在する MH の数は MSS の数に比べはるかに大きいと考えられる．また，MH の数がプロトコル開始時に既知でないことや，実行中に変化することもある．よって，プロトコル全体のコストが MH の数にできるだけ依存しないことが望まれる．

3. ハンドオフ発生時に，プロトコルの実行を正しく継続するため特別の処理が必要となる場合がある．ハンドオフによって引き起こされる特別な情報交換や，プロトコル実行の遅延操作などをできるだけ少なくする必要がある．

本論文では，上記の三つの目標を達成する分散移動

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan
[*] 現在，(株) NTT データ

システム上の前後関係保存放送プロトコルを提案する．任意の計算機から送信されたメッセージ間に何らかの因果関係がある場合，その順序に従って受信するようなメッセージ交換を，前後関係保存メッセージ交換と呼ぶ．前後関係保存メッセージ交換は，システムの監視，資源割当て，電子ニュース，電子会議など，分散システムのさまざまな場面において利用される．MH 間前後関係保存放送とは，放送メッセージのあて先が常にすべての計算機である前後関係保存メッセージ交換のことを指す．

しかし，非同期式分散システムでは，計算機の処理の非同期性と通信の非同期性から，異なる計算機におけるイベントの生起した順序（実時間に基づいた前後関係）を決めることは不可能である．そこで，Lamport はイベント間の因果関係に基づいた前後関係（causal relation (→)）を提案している [1]．ここで，任意のイベント a，b について，$a \rightarrow b$ が成り立つのは以下のいずれかの条件が成り立つ場合である．(1) 同一計算機内で a，b の順にイベントが生起した．(2) あるメッセージの送信と受信に対応するイベントがそれぞれ a，b である．(3) $a \rightarrow c$ かつ $c \rightarrow b$ が成り立つような，あるイベント c が存在する．

この Lamport の前後関係を利用して，さまざまな前後関係保存メッセージ交換プロトコルが提案されている．まず，MH を含まない分散システムにおいて前後関係保存マルチキャスト（任意の複数の計算機をあて先とするメッセージ交換）が提案された [3]．文献 [3] のプロトコルは各メッセージにそれ以前に送信されたメッセージを添付する方法を用いている．この，各メッセージに何らかの情報をメッセージヘッダとして添付する方法は，その後のさまざまな前後関係保存メッセージ交換プロトコル [4], [6], [7], [9] でも利用されており，ヘッダの大きさ（メッセージオーバヘッド）がプロトコルの重要な評価尺度となる．文献 [3] のプロトコルは，過去のメッセージをすべてヘッダに含むので非常にメッセージオーバヘッドが大きい．そこで，RST プロトコル [6] では，ベクトル時計 [2] のアイデアを利用しメッセージオーバヘッドを削減した．システム内の総計算機数を N とすると，RST プロトコルで使用するメッセージヘッダは N 個の N 次ベクトルだけでよい．つまり，RST プロトコルのメッセージオーバヘッドは $\Theta(N^2)$ である．また，文献 [4] では，RST プロトコルを放送に特化することで，使用するメッセージヘッダは 1 個の N 次ベクトルとし，メッセージオーバヘッドを $\Theta(N)$ としている．

近年，分散移動システムにおいても，ベクトル時計のアイデアを用いた MH 間の前後関係保存メッセージ交換プロトコルが多数提案されている．文献 [7] では，メッセージオーバヘッドが $O(N^2)$ である前後関係保存マルチキャストプロトコルが提案されている．このプロトコルは，メッセージのあて先 MH の数が多いほど使用するメッセージヘッダは小さくなり，特に放送に適用した場合，メッセージオーバヘッドは $\Theta(N)$ となる．また，MH の処理をその MH と接続している MSS が代理することで，MH で必要な記憶量，計算量，通信量を小さく抑えており，目標 1 を満たしている．しかし，このプロトコルのメッセージオーバヘッドはすべての計算機数 N に依存しているため，目標 2 を満たさない．文献 [9] ではメッセージオーバヘッドが MH の数に依存しない目標 2 を満たす前後関係保存マルチキャストプロトコルを提案している．システム内の MSS の数を N_{mss} とすると，このプロトコルのメッセージオーバヘッドは，$\Theta({N_{mss}}^2)$ である．また，文献 [7] と同様に MH の処理を MSS が代理することで，目標 1 も満たしている．しかし，このプロトコルでは，前後関係を保証するために，MH のハンドオフが生じたときに，その MH をあて先とする伝送中のメッセージがあるかどうかをすべての MSS に問い合わせる．このため，ハンドオフ発生時の処理に必要なメッセージ数が多く，その遅延時間も大きい．このため，目標 3 を満たしているとはいえない．

本論文では，分散移動システムにおける MH 間の前後関係保存放送を実現するプロトコルを提案する．提案するプロトコルは，分散移動システムの MH と MSS による階層構造を利用し，放送のメッセージオーバヘッドを MH の数に依存しない $\Theta(N_{mss})$ としている．また，文献 [7], [9] と同様に，MH の処理を MSS が代理することで，MH で必要な記憶量，計算量，通信量を小さく抑えている．ハンドオフが発生した場合，提案するプロトコルでは，MH をハンドオフする二つの MSS 間で情報を伝達するのみであり，文献 [9] のプロトコルのような MSS 全体にかかわる情報交換を必要としない．更に，文献 [7], [9] のプロトコルでは，MH をハンドオフする MSS 間でその MH あてのメッセージを転送する必要がある．しかし，提案するプロトコルは，各 MSS で必要な過去のメッセージを保存することで，このような MSS 間でのメッセージ転送を不要としている．

2. では分散移動システムのモデルとこのモデルにおける前後関係保存放送を定義する．**3.** では移動計算機間における前後関係保存放送を実現するプロトコルを提案する．**4.** でプロトコルの正当性を示し，最後に **5.** でプロトコルの評価を行う．

2. 諸 定 義

2.1 分散移動システム

本論文では，分散移動システムのモデルとして文献 [8] のモデルを用いる．以下では，このモデルを簡単に示す（図 1）．分散移動システム S を 3 項組 $S = (\mathbf{MSS}, \mathbf{MH}, \mathbf{CH})$ と定義する．**MSS** は移動支援局（MSS）と呼ばれるプロセスの集合，**MH** は移動計算機（MH）と呼ばれるプロセスの集合，**CH** は MSS 間の静的通信チャネルの集合である．本論文では簡単のために，システム内に存在する固定計算機はすべて MSS であるとする．MSS の数を N_{mss}，MH の数を N_{mh} とし，$\mathbf{MSS} = \{s_1, \cdots, s_{N_{mss}}\}$，$\mathbf{MH} = \{h_1, \cdots, h_{N_{mh}}\}$ とする．各プロセスは固有の識別子をもち，簡単のために s_i，h_i の識別子を単に s_i，h_i と表す．

二つの MSS 間に静的通信チャネルが存在する場合，その両端の MSS は相互に通信可能である．**MSS** と **CH** から構成されるネットワークは連結であるとする．つまり，任意の MSS 間に静的通信チャネルによる経路が少なくとも一つ存在する．各 MH は動的通信チャネルと呼ばれる通信チャネルを用いて，一つの MSS と相互に通信可能である．ハンドオフが発生すると，MH が動的通信チャネルで接続している MSS が変更される．例えば，MH h が MSS s に接続しているときに，s から別の MSS s' へ h のハンドオフが発生したとすると，h と s との間の動的通信チャネルが消失し，h と s' との間に新たに動的通信チャネルが生成される．このように，動的通信チャネルは MH とどの MSS との間に存在するか決まっておらず動的に変化するため，分散移動システムのモデルにおいて動的通信チャネルは，その両端の MSS の状態と MH の状態の組として表されているとする．つまり，各プロセスの状態には接続状態として，動的チャネルで接続されているプロセスの識別子の集合が含まれている．MH のハンドオフ処理は有限時間内に終了し，ハンドオフ処理中を除いてすべての MH はいずれかの MSS と接続していると仮定する．

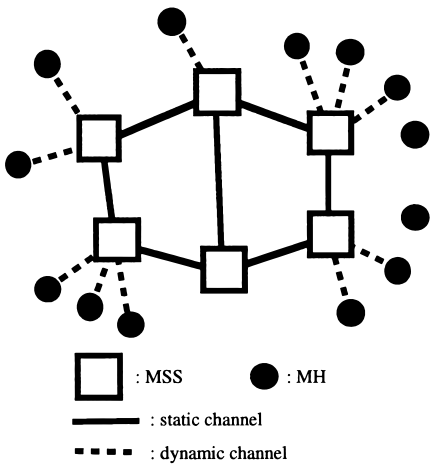

図 1　分散移動システム
Fig. 1　A distributed mobile system.

すべての静的・動的通信チャネルは FIFO キューであるとする．すなわち，通信チャネルを用いて送信したメッセージは送信された順に相手に受信される．更に，チャネルを用いて送信されたメッセージは紛失されないとする．文献 [8] のモデルでは，ハンドオフ時に動的通信チャネルが消失した場合，その FIFO キュー内のメッセージは紛失するとしている．しかし本論文では，簡単のために，静的，動的にかかわらずすべての通信チャネルに入力されたメッセージは有限時間内に出力され，メッセージの紛失は起きないと仮定する．ただし本論文のプロトコルは，動的通信チャネルにおけるメッセージの紛失に対応できるように簡単に拡張できる．

分散移動システムの状況は，システム内のすべてのプロセスの状態とすべての通信チャネルの状態から構成される．プロセスでイベントが生起することにより，分散移動システムの状況が変化する．一般性を失わずシステム全体で同時に一つのイベントしか生起しないと仮定できる．状況とイベントの交互列 $\gamma^0, e^1, \gamma^1, e^2, \gamma^2, \cdots$（$\gamma^i$，$e^i$ はそれぞれ状況，イベントを表す）を分散移動システムの実行と定義する．ただし，e^i は γ^{i-1} で生起可能なイベントであり，γ^{i-1} で e^i が生起した後の状況が γ^i である．実行は無限系列であってもよく，有限の場合は状況で終わるものとする．ここで，γ^0 は分散移動システムの初期状況であり，初期状況ではすべての MH がいずれかの MSS と接続しているとする．

プロセスで生起するイベントとして次の 7 種類のイ

ベントを定義する．まず,（MH を含まない）従来の分散システムで用いられている 3 種類のイベントを定義する．

1. *internal*：プロセスの内部計算を表すイベント．
2. *send*：メッセージの送信を表すイベント．
3. *receive*：メッセージの受信を表すイベント．

次に，分散移動システム特有のハンドオフを扱うための 4 種類のイベントを定義する．これらは，MH h_i が MSS s_j から MSS s_k へハンドオフされたときに，4. 5. 6. 7. の順にそれぞれ 1 回ずつ生起するイベントで，ハンドオフにかかわるプロセスの接続状態の変化と，移動情報の交換を表す．移動情報とは，MH のハンドオフの際に，ハンドオフにかかわるプロセス間で交換される情報で，h_i から s_j へ，s_j から s_k へ，s_k から h_i への順に交換される情報である．分散移動システム上で動作するプロトコルも，移動情報を利用して情報交換可能であるものとする．

4. $disconnect_i$：h_i で生起するイベント．h_i の接続状態が空になることと，h_i から s_j への移動情報 f の送信を表す．

5. $remove_j$：s_j で生起するイベント．s_j の接続状態からの h_i の削除と，h_i から送信された移動情報 f の受信，s_k への移動情報 f' の送信を表す．

6. $accept_k$：s_k で生起するイベント．s_k の接続状態への h_i の追加と，s_j から送信された移動情報 f' の受信，h_i への移動情報 f'' の送信を表す．

7. $connect_i$：h_i で生起するイベント．h_i の接続状態が $\{s_k\}$ になることと，s_k から送信された移動情報 f'' の受信を表す．

2.2 前後関係保存放送

あるプロセスからすべてのプロセスへあるメッセージを伝達することを放送と呼び，放送を実現するプロトコルを放送プロトコルと呼ぶ．プロセスがメッセージの放送を行うとき，まず放送プロトコルへ放送要求を出す．放送要求を受けた放送プロトコルはすべてのプロセスで放送要求されたメッセージの配達処理を行う．

プロセス p_i における放送メッセージ m の放送要求と配達を，それぞれ p_i のイベント $cbcast_i(m)$，$deliver_i(m)$ で表す．混乱が生じない限り，i，m を省略することがある．*cbcast*，*deliver* をまとめて放送イベントと呼ぶ．以後，簡単のために，任意の実行において放送要求がなされる放送メッセージは相異なると仮定する．

［定義 1］（放送イベント間の前後関係）分散移動システムにおける任意の実行を $\mathcal{E}$，$\mathcal{E}$ に含まれるすべての放送イベントの集合を E_B とする．E_B 上の前後関係（$\xrightarrow{B}$）は以下のいずれかを満たす最小の 2 項関係である．

1. 同じプロセスの異なる二つの放送イベント e，e' で，$\mathcal{E}$ において e が e' より前に現れるとき，$e \xrightarrow{B} e'$．
2. 任意の放送メッセージ m と任意のプロセス p_i に対し，$cbcast(m) \xrightarrow{B} deliver_i(m)$．
3. 次の推移律が成り立つ．

$\forall e, e', e'' \in E_B[(e \xrightarrow{B} e') \wedge (e' \xrightarrow{B} e'') \Rightarrow e \xrightarrow{B} e'']$．

□

二つの放送要求間に前後関係がある場合，対応する同一プロセス上の二つの配達がその前後関係を保存する放送を，前後関係保存放送と呼ぶ．

［定義 2］（前後関係保存放送）分散移動システムにおける任意の実行 $\mathcal{E}$ について，以下の三つの性質を保証する放送を前後関係保存放送と定義する．

1. 各 $cbcast(m)$ に対し，任意のプロセス p_i 上で $deliver_i(m)$ が正確に 1 回生起する．
2. あるプロセスで $deliver(m)$ が生起するならば，$cbcast(m)$ が生起するあるプロセスが存在する．
3. 任意の二つの放送メッセージ m_1，m_2 について，$cbcast(m_1) \xrightarrow{B} cbcast(m_2)$ ならば，任意のプロセス p_i において $deliver_i(m_1) \xrightarrow{B} deliver_i(m_2)$ である． □

本論文では，以下に定義する MH 間前後関係保存放送プロトコルを提案する．

［定義 3］（MH 間前後関係保存放送）定義 2 の条件 1，2，3 の対象をすべてのプロセスではなく，システム上のすべての MH に限定して定義した前後関係保存放送を MH 間前後関係保存放送と定義する． □

本論文では，簡単のため，定義 3 のように，MH のみが放送要求を出し，MH のみにメッセージを配達することを考える．MSS も放送要求を出し，MSS へのメッセージ配達が必要な場合には，MSS 内に一つの（移動しない）MH を仮想的に考え，この MH の放送要求，メッセージ配達として処理することにより，本論文のプロトコルを適用できる．

3. MH 間前後関係保存放送プロトコル

3.1 基本アイデア

MH 間前後関係保存放送を実現するプロトコルを提案する．本プロトコルではシステム内に存在する MSS とその識別子が全 MSS で利用可能とする．MH の総

数や識別子は利用可能である必要はない．ただし，すべての MH はハンドオフを行う場合を除いていずれかの MSS に接続していると仮定する．また，ハンドオフは有限時間内に完了するものとする．つまり，ある MH で *disconnect* が生起した場合，有限時間内に必ず *connect* がその MH で生起すると仮定する．

Birman は MH を含まない分散システム上の前後関係保存放送プロトコルを提案している [4]．このプロトコルはベクトル時計 [2] のアイデアを次のように利用している．システム内のプロセスを $p_1, p_2, \cdots, p_N$（N は総プロセス数）とする．プロセス p_i が放送メッセージ m を放送するとき，ベクトル $(v_1, v_2, \cdots, v_N)$ をヘッダとして m に添付する．ここで，各 v_j $(j \neq i)$ は，p_i が m の放送要求を行う前に p_i で配達された p_j の放送した放送メッセージの個数であり，v_i は m を含めて p_i が放送要求を行った放送メッセージの個数を意味する．m を受信したプロセスは，p_i の放送した $v_i - 1$ 個，各プロセス p_j $(j \neq i)$ の放送した v_j 個の放送メッセージを配達するまで m の配達を延期することにより，前後関係保存放送を実現する．

Birman のプロトコルをそのまま分散移動システム上の MH へ適用すると，各メッセージに付加するヘッダは N_{mh} 次ベクトルとなり，前後関係保存放送を実現するための通信オーバヘッドが非常に大きい．また，MH のハンドオフを考慮していないためいくつかの問題点が発生する．そこで，まず MH のハンドオフが発生しない場合について，メッセージのヘッダサイズを減らす方法を考案する．その後，MH のハンドオフに伴う問題点とその対策法を示す．

分散移動システムは，MSS と MH による階層構造をなしている．すなわち，MH が他の MH とメッセージを交換するときは，必ず MSS を介してのみ行う．更に，MH と MSS の間の通信チャネルでは，メッセージは送信された順番に受信される．したがって，MH のハンドオフが発生しない場合は，MSS 間において前後関係保存放送を行うことで，MH 間の前後関係保存放送を実現することができる（図 2）．

MSS 間前後関係保存放送において MSS s_j の放送メッセージ m の放送要求イベントを $mss\_cbcast_j(m)$，配達イベントを $mss\_deliver_j(m)$ と表す．ある二つの MH h_1，h_2 がそれぞれ MSS s_1，s_2 に接続しているときに，放送メッセージ m_1，m_2 をそれぞれ送信したとき，$cbcast_1(m_1) \xrightarrow{B} cbcast_2(m_2)$ が成り立つならば，$mss\_cbcast_1(m_1) \xrightarrow{B} mss\_cbcast_2(m_2)$

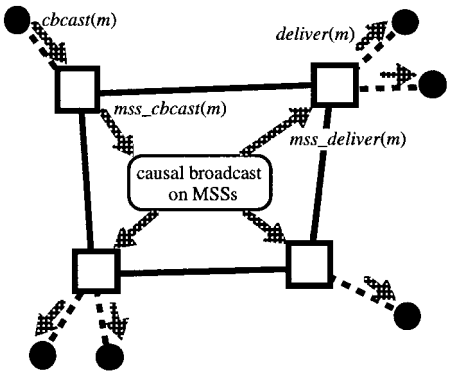

図 2　MSS 間前後関係保存放送の利用
Fig. 2　Application of causal broadcast on MSSs.

も成り立つ．このとき，MSS 間前後関係保存放送はすべての MSS s_j について $mss\_deliver_j(m_1) \xrightarrow{B} mss\_deliver_j(m_2)$ を保証するため，すべての MH h_i について $deliver_i(m_1) \xrightarrow{B} deliver_i(m_2)$ が保証される．

MSS 間前後関係保存放送を実現するために Birman のプロトコルを用いると，各放送メッセージに付加するベクトルの次元は N_{mss} となり，ヘッダサイズが MH の数 N_{mh} に依存しないプロトコルを実現できる．

3.2　ハンドオフへの対応

MH のハンドオフに対応するために，前節の基本アイデアを以下のように拡張する．

a. 配達メッセージの重複・欠落の回避：MH h_i が MSS s_j から MSS s_k へハンドオフされたとする．MSS 間におけるメッセージ交換は非同期であるため，ある時点において MSS s_j と MSS s_k の配達済み放送メッセージは一般に異なる．このため，h_i が放送メッセージ m を s_j から受信した後，ハンドオフ先の s_k からも更に m を受信してしまうと，h_i は m を重複して配達してしまうことがある（配達メッセージの重複）．また，ある放送メッセージ m' が s_j で未配達であり，h_i のハンドオフ先の s_k で既に m' が配達済みであるとき，h_i は永久に m' を受信・配達ができない（配達メッセージの欠落）ことがある．

配達メッセージの重複，欠落を回避するために，各 MSS s_j は s_j に接続している各 MH h_i で配達済みの放送メッセージ数を表す N_{mss} 次ベクトル $\mathbf{RECV}_j[h_i]$ を管理する．ここで，$\mathbf{RECV}_j[h_i][s_k]$ $(s_k \in \mathbf{MSS})$ は MSS s_k から放送されたメッセージのうち，既に h_i へ送信された放送メッセージの数を表す．$\mathbf{RECV}_j[h_i]$ を用いることにより，h_i で配達されるメッセージの重

複，欠落は容易に回避できる．ただし，メッセージの欠落を回避するために，各 MSS s_j で配達済みの放送メッセージをキュー **DELIV_MES**$_j$ に配達順に保存する．

b. メッセージの前後関係非保存の回避：MH h_i が MSS s_j から s_k へハンドオフされたとする．また，h_i は s_j へ放送メッセージ m_1 を送信した後，s_k へハンドオフされ，更に s_k へ放送メッセージ m_2 を送信したとする（図 3）．このとき，$cbcast_i(m_1) \xrightarrow{B} cbcast_i(m_2)$ という前後関係が成り立つため，m_1 は m_2 より先に各 MSS で配達されなければならない．しかし，基本アイデアにおける放送メッセージに添付するヘッダは，MSS 間放送における前後関係のみ表しており，同一の MH が異なる MSS に対して送信した放送メッセージ間の前後関係を表していない．そのため，別の MSS では m_1 より前に m_2 が配達されることがある．

上記の問題を回避するために，次のようにプロトコルを拡張する．各 MSS s_j は次に送信する放送メッセージに先行する放送メッセージ数を N_{mss} 次のベクトルによって MSS ごとに管理している．このベクトルを SENT$_j$ とする．$\mathcal{M}_k$ を MSS s_k の送信した放送メッセージの集合とすると，s_j で $mss\_cbcast_j(m)$ が生起したときには，各 MSS s_l に対して，SENT$_j[s_l] = |\{m' | m' \in \mathcal{M}_l \wedge mss_cbcast_l(m') \xrightarrow{B} mss_cbcast_j(m)\}|$ が成り立つものとする．h_i がハンドオフの前後で送信した放送メッセージの前後関係を保存するためには，各 MSS s_l に対し，h_i のハンドオフ直後の SENT$_k[s_l]$ の値が，h_i のハンドオフ直前の SENT$_j[s_l]$ 以上となれば十分である．そこで，ハンドオフ時に移動情報を用いて SENT$_j$ の値を s_k へ伝え，s_k は，各 MSS s_l に対して，SENT$_k[s_l] := \max(\text{SENT}_k[s_l], \text{SENT}_j[s_l])$ を実行し SENT$_k$ を更新する（図 3）．

このように MSS s_k において SENT$_k$ を更新すると，例えば，MSS s_j で配達済みのメッセージが s_k で未配達の場合，SENT$_k$ が s_k で配達済みの放送メッセージ数よりも大きくなることがある．そこで，s_k で配達済みのメッセージ数を管理する別のベクトル DELIV$_k$ を用意して，配達の実行・延期の判断に使用する．

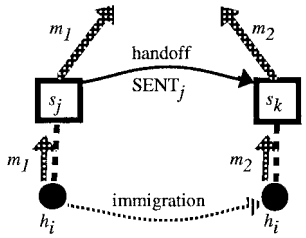

図 3　ハンドオフ時の SENT の転送
Fig. 3　Transfer of SENT in the handoff procedure.

3.3　配達済みキューからの放送メッセージの削除

配達メッセージの欠落を避けるために各 MSS s_j に **DELIV_MES**$_j$ を導入したが，すべての放送メッセージをすべての MSS で保存するためには膨大な記憶容量を必要とする．そこで，提案するプロトコルでは，各メッセージのヘッダに更に N_{mss} 次ベクトルを加えることで，すべての MH で配達済みと確認された放送メッセージをキュー **DELIV_MES** から効率良く削除する．

各 MSS s_j は新たな放送メッセージを他の MSS へ送信する際に，接続しているすべての MH で配達済みの放送メッセージを表す N_{mss} 次ベクトル *REDUCE* を放送メッセージに添付する．ここで，*REDUCE*$[s_k]$ は，s_j に接続しているすべての MH で受信済みの，MSS s_k から送信された放送メッセージ数を表す．各 MSS s_j は受信した *REDUCE* を配列 RECV_RDC$_j$[**MSS**] で管理する．RECV_RDC$_j[s_k]$ は s_j が MSS s_k から受信した *REDUCE* の最大ベクトルとする．s_j は s_k が送信した放送メッセージを配達するときに，RECV_RDC$_j[s_k]$ を更新し，すべての MH で配達済みと確認できた放送メッセージを **DELIV_MES**$_j$ から削除する．ただし，MH h_i が MSS s_j から MSS s_k へハンドオフされるとき，h_i で配達済み放送メッセージを表す RECV$_i$ が移動情報を用いて s_j から s_k へ送られるが，RECV$_i$ が s_j にも s_k にも存在しない瞬間が生じるため，h_i の配達済み放送メッセージの情報が s_j と s_k のどちらのメッセージの *REDUCE* にも反映されず，h_i で未配達の放送メッセージが削除されてしまう可能性がある．そこで，s_j は RECV$_i$ を s_k へ送った後，s_k が RECV$_i$ を受け取ったこと（s_k 上の *accept* の生起）を確認してから，s_j は RECV$_i$ を削除する．以上の手法により，すべての MSS が適当な頻度で放送メッセージを送信することが保証されていれば，すべての MH で配達済みの放送メッセージを **DELIV_MES** から削除できる．

3.4　プロトコルの詳細

前後関係保存放送プロトコルを，MSS と MH のそれぞれについて，イベント駆動型のプログラムで記述

する．あるイベントに対するプログラムの実行中に別のイベントが生起した場合，実行中のプログラムが終了してからそれらのイベントに対するプログラムがイベントの生起順に実行される．プログラム中の **wait**(e) という記述は，**wait**(e) が呼び出されてからイベント e が生起するまで，そのプログラムの実行が停止することを表す．プログラム中の演算 max，min が配列に対して適用された場合，配列の要素ごとに，最大値，最小値を選ぶ演算であるとする．また，プログラム中の任意の二つの N_{mss} 次配列（ベクトル）V_1，V_2 について，$\forall s \in \mathbf{MSS}[V_1[s] \leqq V_2[s]] \wedge \exists s' \in \mathbf{MSS}[V_1[s'] < V_2[s']]$ が成り立つとき，$V_1 < V_2$ であるとする．更に，プログラムでは MSS 間の放送を行うマクロ命令を利用できるとする．このマクロ命令は，メッセージの受信順には何の保証もしないが，送信されたメッセージは有限時間内にすべての MSS で正確に 1 回受信されることを保証するものとする．

分散移動システムにおける 7 種のイベントのうち，プロトコル中でプログラムの駆動に利用するイベントは *internal*，*receive*，*disconnect*，*remove*，*accept*，*connect* の 6 種類である．特に MH で生起するイベント *internal* のうち，放送メッセージ m の放送要求が出されたことを表す MH の内部イベントを $cbcast(m)$ としてプログラムの駆動に利用する．$receive(p, m)$ は，プロセス（MH 又は MSS）p から m というメッセージを受信したことを表す．$disconnect(old)$，$remove(new, mh, PUT)$，$accept(old, mh, GET)$，$connect(new)$ は MH mh が MSS old から MSS new へハンドオフされたときにこの順番で生起するイベントである．ここで，PUT，GET は old から new へ送受信される移動情報を表す．以下に，MH h_i の実行するプログラム中で使用する変数を説明する．ここで，変数の添字はその変数をもつ MH の添字に等しい．

- $TELEPOINT_i$：h_i が接続中の MSS を指す．ただし，ハンドオフ処理中で接続する MSS が存在しないときは値として *null* を使用する．

以下に，MSS s_j で実行するプログラム中で使用する変数を説明する．ここで，**MSS** はシステム内に存在するすべての MSS の識別子の集合を表す．また，変数の添字はその変数をもつ MSS の添字に等しい．

- $\mathbf{MH}_j$：MSS s_j が接続中の MH の識別子の集合．
- $DELIV_j[\mathbf{MSS}]$：$DELIV_j[s_k]$ は MSS s_k からの放送メッセージのうち s_j で配達済みの放送メッセージ数（初期値：すべての要素が 0）．
- $SENT_j[\mathbf{MSS}]$：放送メッセージの前後関係の管理に使用．MSS 間で交換するメッセージのヘッダとなる（初期値：すべての要素が 0）．
- $RECV_j[\mathbf{MH}_j][\mathbf{MSS}]$：$RECV_i[h_i][s_k]$ は MSS s_j と接続中の MH h_i で配達された，MSS s_k から送信された放送メッセージの数（初期値：すべての要素が 0）．
- $\mathbf{DELIV\_MES}_j$：MSS s_j で配達済みの放送メッセージを，配達順に保存するキュー（初期値：空キュー）．要素の順序を保存したまま，任意のキューの要素を削除可能であるとする．
- $\mathbf{WAITING}_j$：MSS s_j で受信されたが，前後関係を保存するために s_j での配達を延期している放送メッセージを保存するバッファ．
- $\mathbf{MOVING}_j$：MSS s_j から別の MSS へハンドオフされたが，ハンドオフ先の MSS との接続を確認していない MH の識別子とその MH に対する $RECV$ の 2 項組を要素とする集合（初期値：$\emptyset$）．
- $RECV\_RDC_j[\mathbf{MSS}]$ ： $RECV\_RDC_j[s_k]$ は MSS s_j が MSS s_k から受信したベクトル $REDUCE$（削除可能な放送メッセージの情報）を表す（初期値：すべての要素が零ベクトル）．

⟨⟨ MSS s_j で実行するアルゴリズム ⟩⟩

1. $receive(h_i, m)$: /* h_i から m の放送要求 */
   $\mathrm{SENT}_j[s_j] := \mathrm{SENT}_j[s_j] + 1$;
   $REDUCE := \mathbf{DELIV}_j$;
   foreach $(mh, RECV) \in \mathbf{MOVING}_j$ **do**
   $REDUCE := \min(REDUCE, RECV)$;
   $(\mathrm{SENT}_j, \mathrm{REDUCE}, m)$ をすべての MSS へ放送;
2. $receive(s_k, (S, R, m))$:
   /* Deliver Check 開始 */
   if $S[s_k] = \mathrm{DELIV}_j[s_k] + 1$ **and**
   $\forall mss \in (\mathbf{MSS} - \{s_k\})[S[mss] \leqq \mathrm{DELIV}_j[mss]]$
   then begin /* 配達処理 */
   foreach $mh \in \mathbf{MH}_j$ **do**
   if $\mathrm{RECV}_j[mh][s_k] = \mathrm{DELIV}_j[s_k]$ **then begin**
   m を mh へ送信;
   $\mathrm{RECV}_j[mh][s_k] := \mathrm{RECV}_j[mh][s_k] + 1$;
   end;
   $\mathrm{DELIV}_j[s_k] := S[s_k]$;
   $\mathrm{SENT}_j[s_k] := \max(\mathrm{SENT}_j[s_k], S[s_k])$;
   (s_k, S, m) を $\mathbf{DELIV\_MES}_j$ の最後尾に追加;
   foreach $(s', S', m')]) \in \mathbf{WAITING}_j$ **do**
   (s', S', m') を $\mathbf{WAITING}_j$ から
   取り出して Deliver Check
   （配達すれば (s', S', m') を $\mathbf{WAITING}_j$ から削除）;
   end else
   $\mathbf{WAITING}_j := \mathbf{WAITING}_j \cup \{(s_k, S, m)\}$;

/* Deliver Check 終了 */
$\mathrm{RECV\_RDC}_j[s_k] := \max(\mathrm{RECV\_RDC}_j[s_k], R)$;
$MIN\_R := \min\{\mathrm{RECV\_RDC}_j[s] | s \in \mathbf{MSS}\}$;
foreach $(s', S', m') \in \mathbf{DELIV\_MES}_j$ **do**
/* キューの先頭要素から順に選択 */
if $S' \leqq MIN\_R$ **then**
(s', S', m') を $\mathbf{DELIV\_MES}_j$ から削除;
else
break foreach;
3. $remove(s_k, h_i, (\mathrm{RECV}_j[h_i], \mathrm{SENT}_j))$:
$\mathbf{MH}_j := \mathbf{MH}_j - \{h_i\}$;
$\mathbf{MOVING}_j := \mathbf{MOVING}_j \cup \{(h_i, \mathrm{RECV}_j[h_i])\}$;
4. $accept(s_k, h_i, (\mathrm{RECV}_j[h_i], S))$:
$\mathbf{MH}_j := \mathbf{MH}_j \cup \{h_i\}$;
$\mathrm{SENT}_j := \max(\mathrm{SENT}_j, S)$;
foreach $(s', S', m') \in \mathbf{DELIV\_MES}_j$ **do**
/* キューの先頭要素から順に選択 */
if $\mathrm{RECV}_j[h_i][s'] < S'[s']$ **then**
m' を h_i へ送信;
$\mathrm{RECV}_j[h_i] := \max(\mathrm{RECV}_j[h_i], \mathrm{DELIV}_j)$;
$(accept, h_i)$ を s_kに送信;
5. $receive(s_k, (accept, h_i))$:
$\mathbf{MOVING}_j := \mathbf{MOVING}_j - \{(h_i, *)\}$;

⟨⟨ MH h_i で実行するアルゴリズム ⟩⟩

1. $cbcast(m)$:
m を $\mathrm{TELEPOINT}_i$ へ送信;
2. $receive(mss, m)$:
$deliver(m)$;
3. $disconnect(mss)$:
$\mathrm{TELEPOINT}_i := null$
wait$(connect)$;
4. $connect(mss)$:
$\mathrm{TELEPOINT}_i := mss$

4. 正当性の証明

証明のために，まず分散移動システムの実行に現れるすべてのイベント間の前後関係（$\xrightarrow{L}$）[8] を定義する．

MH では，放送メッセージの放送要求と同時にその放送メッセージの送信が行われ，放送メッセージの受信と同時に受信した放送メッセージの配達が行われるので，MH h_i の放送メッセージ m に関するイベントについて，$cbcast_i(m)$ と $send_i(m)$，$deliver_i(m)$ と $receive_i(m)$ はそれぞれ同一のイベントとして扱う．

MSS 間の放送に関するイベントを次のように表す．MSS s_j における放送メッセージ m の放送要求イベントを $mss\_bcast_j(m)$ とする．m の放送要求と同時に m の送信が行われるので，$mss\_bcast_j(m)$ と，m のあて先が MSS である送信イベント $send_j(m)$ を同一として扱う．ある MSS から送信されたメッセージ m の MSS s_j における受信イベント $receive_j(m)$ を特に $mss\_recv_j(m)$ と呼ぶ．また，放送メッセージ m の MSS s_j における配達イベント（s_j の内部イベント）を $mss\_deliv_j(m)$ とする．

［定義 4］（イベントの前後関係）分散移動システムにおける任意の実行を $\mathcal{E}$ とし，$\mathcal{E}$ に現れるすべてのイベントの集合を E とする．E 上の前後関係（$\xrightarrow{L}$）は以下の条件を満たす最小の 2 項関係である．

1. 同じプロセスの異なる二つのイベント e，e' で，$\mathcal{E}$ において e が e' より前に現れるとき，$e \xrightarrow{L} e'$.
2. 任意のメッセージ m に関する送受信イベントをそれぞれ $send(m)$，$receive(m)$ とすると，$send(m) \xrightarrow{L} receive(m)$.
3. 任意のハンドオフに対応する四つのイベントについて，$disconnect \xrightarrow{L} remove \xrightarrow{L} accept \xrightarrow{L} connect$.
4. $\forall e, e', e'' \in E[(e \xrightarrow{L} e') \wedge (e' \xrightarrow{L} e'') \Rightarrow e \xrightarrow{L} e'']$

□

MSS 間で交換される放送メッセージ m に添付されるベクトルを $SENT(m)$ と表す．$SENT(m)$ について，次の補題が成り立つ．

［補題 1］ 任意の実行において以下の性質が成り立つ．

（性質 1） s_j を任意の MSS，m_1，m_2 を s_j が放送した任意の放送メッセージとする．このとき，$mss\_bcast_j(m_1) \xrightarrow{L} mss\_bcast_j(m_2)$ ならば，$SENT(m_1)[s_j] < SENT(m_2)[s_j]$ が成り立つ．

（性質 2） s_j を任意の MSS，m_1 を任意の放送メッセージとする．このとき，$1 \leqq a \leqq SENT(m_1)[s_j]$ なる任意の a に対し，MSS s_j が放送したメッセージ m_2 が存在し，$SENT(m_2)[s_j] = a$ が成り立つ．

（性質 3） m_1，m_2 を相異なる放送メッセージ，s_k を m_1 を放送した MSS とする．このとき，$SENT(m_1)[s_k] \leqq SENT(m_2)[s_k]$ ならば，$SENT(m_1) < SENT(m_2)$ が成り立つ．

［補題 2］ m，m' を任意の放送メッセージとする．$cbcast(m) \xrightarrow{B} cbcast(m')$ ならば，$SENT(m) < SENT(m')$ が成り立つ．

（証明） $cbcast(m) \xrightarrow{B} cbcast(m')$ より，放送メッセージの連鎖 $m_1\ (= m), m_2, \cdots, m_q\ (= m')$ が存在し，各 p $(1 \leqq p < q)$ に対して次のいずれかを満たす MH h_i が存在している．

1. $cbcast_i(m_p) \xrightarrow{B} cbcast_i(m_{p+1})$.
2. $deliver_i(m_p) \xrightarrow{B} cbcast_i(m_{p+1})$.

いずれの場合にも，$SENT(m_p) < SENT(m_{p+1})$ が成り立つことを示せば十分である．

（1 の場合） h_i が m_p を MSS s_j へ，m_{p+1} を MSS s_k へ放送したとする．$s_j = s_k$ の場合，s_j

は m_p，m_{p+1} の順に h_i からメッセージを受信するので，$mss\_bcast_j(m_p) \xrightarrow{L} mss\_bcast_j(m_{p+1})$ であり，性質 1 より $SENT[m_p] < SENT[m_{p+1}]$ が成り立つ．$s_j \neq s_k$ の場合，s_j が m_p を放送してから s_k が m_{p+1} を放送するまでに，s_j から s_k まで，h_i のハンドオフの連鎖がある．h_i は $\sigma_1\ (=s_j), \sigma_2, \cdots, \sigma_b\ (=s_k)$ の順にハンドオフされたとする．ここで，$\sigma_a\ (1 \leqq a \leqq b)$ は MSS であり σ_a のもつ SENT を $\mathrm{SENT}_{\sigma(a)}$ と表す．各 $a\ (1 \leqq a < b)$ に対して，h_i が σ_a から σ_{a+1} へハンドオフされるとき，σ_{a+1} は移動情報として σ_a から $\mathrm{SENT}_{\sigma(a)}$ を受け取り，$\mathrm{SENT}_{\sigma(a+1)} := \max(\mathrm{SENT}_{\sigma(a)}, \mathrm{SENT}_{\sigma(a+1)})$ として $\mathrm{SENT}_{\sigma(a+1)}$ を更新する．よって，h_i が s_k へハンドオフされたとき，$SENT(m_p) \leqq \mathrm{SENT}_k$ である．この後，s_k は $\mathrm{SENT}_k[s_k]$ を 1 増やしてから m_{p+1} を放送するので，$SENT(m_p) < SENT(m_{p+1})$ が成り立つ．

（2 の場合） 1 の場合と同様に，$SENT(m_p) < SENT(m_{p+1})$ が成り立つ． □

$\mathrm{RECV}_j[h]$ は，MH h が接続中の MSS s_j がもつ変数であるが，h が別の MSS s_k にハンドオフされると $\mathrm{RECV}_j[h]$ の値はそのまま $\mathrm{RECV}_k[h]$ として引き継がれ，$\mathrm{RECV}_j[h]$ は破棄される．つまり，h の接続中の MSS s_j のみが $\mathrm{RECV}_j[h]$ をもつため，この変数を単に $\mathrm{RECV}[h]$ と表す．

［補題 3］ m を任意の放送メッセージ，h を任意の MH とする．このとき，h は m をたかだか 1 回受信する．

［補題 4］ m を任意の放送メッセージ，h を任意の MH とする．このとき，h は m を少なくとも 1 回受信する．

補題 4 を証明するために次の補題 5 を示す．

［補題 5］ m を任意の放送メッセージ，s を任意の MSS とすると，s はいずれ m を配達する．

（補題 5 の証明） ある MSS s_j で配達されない放送メッセージ m が存在すると仮定し，矛盾を導く．一般性を失うことなく，s_j が配達しないメッセージの中で，m が極小のヘッダ $SENT(m)$ をもつとする．m を放送した MSS を s_k としたとき，永久に $SENT(m)[s_k] > \mathrm{DELIV}_j[s_k] + 1$，あるいは，$\exists s \in (\mathbf{MSS} - \{s_k\})[SENT(m)[s] > \mathrm{DELIV}_j[s]]$ が成り立つ．いずれの場合も性質 2，3 より，$SENT(m') < SENT(m)$ なる放送メッセージ m' が存在する．m の極小性より $SENT(m') < SENT(m)$ なる s_j が配達しない放送メッセージ m' が存在することが示せ，m のヘッダの極小性に矛盾する． □

（補題 4 の証明） MH h は永久に放送メッセージ m を受信していないと仮定して矛盾を導く．MSS s_j で m の配達が行われたときに h が s_j と接続している場合，h は s_j から m を受信する．よって，h が MSS s_j と接続しているときに s_j で m の配達は行われていない．ここで，補題 5 よりすべての MSS はいずれ m を配達するので，h はまだ m を配達していない MSS s_k から，既に m を配達済みの MSS s_l へハンドオフされる．m が h で配達されないことから，このハンドオフの完了時点で，s_l の保持するキュー $\mathbf{DELIV\_MES}_l$ から m は削除されている．つまり，m を放送した MSS を s_p とするとすべての MSS s_j から $SENT(m)[s_p] \leqq REDUCE_j[s_p]$ なる $REDUCE_j$ を受信している．s_k で $SENT(m)[s_p] \leqq REDUCE_j[s_p]$ が成り立つためには s_k で既に m が配達されていなければならない．h のハンドオフ直前の時点で，s_k は m をまだ配達していないので，h のハンドオフ発生後に s_k は m を配達したことになる．しかし，h のハンドオフが開始してから，ハンドオフが完了したことを伝えるメッセージを s_l から受信するまでは，s_k の保持する $\mathbf{MOVING}_k$ には $\mathbf{RECV}[h]$ が存在する．h は m を受信していないことから $\mathbf{RECV}[h][s_p] < SENT(m)[s_p]$ が成り立つため，s_k が h のハンドオフ中に s_l へ送信した $REDUCE_k$ について，$REDUCE_k[s_p] < SENT(m)[s_p]$ であることがいえる．これは，s_l が h のハンドオフ完了時点で m をキュー $\mathbf{DELIV\_MES}_l$ から削除している条件に反している． □

［定理 1］ m，m' を任意の放送メッセージ，h_i を任意の MH とする．$cbcast(m) \xrightarrow{B} cbcast(m')$ が成り立つならば，$deliver_i(m) \xrightarrow{B} deliver_i(m')$ が成り立つ．

（証明）まず，任意の MSS s_j に対し，$mss\_deliver_j(m) \xrightarrow{L} mss\_deliver_j(m')$ が成り立つことを示す．補題 2 より $SENT(m) < SENT(m')$ が成り立つ．補題 5 より，s_j は m，m' をいずれ配達する．このとき，$mss\_deliver_j(m) \xrightarrow{L} mss\_deliver_j(m')$ となるのはプロトコルより明らか．補題 3，4 より，各 MH はすべての放送メッセージを正確に 1 度配達する．ある MH h でより前に m' が配達されたと仮定して，矛盾を導く．すべての MSS s_j は m' の配達前に m を配達するため，これらのメッセージはキュー $\mathbf{DELIV\_MES}_j$ において，m，m' の順に 1 回ずつ現れる．h がある MSS s_j

へハンドオフされた時点で，h は m，m' の両方を配達しておらず，s_j が m，m' の両方を $\mathbf{DELIV\_MES}_j$ から h へ送信したとすると，必ず m，m' の順に h へ送信される．これは，仮定に反するので，以下このような場合を除いて考える．

補題 2 より，$SENT(m) < SENT(m')$ が成り立つ．また，m' が配達された後，$SENT(m') \leqq \mathrm{RECV}[h]$ が成り立ち，$\mathrm{RECV}[h]$ の要素はそれぞれ単調非減少である．今 h が MSS s_j に接続しているとして，m を放送した MSS を s_k とすると，s_j が m を h へ送信するのは，$\mathrm{RECV}[h][s_k] < SENT(m)[s_k]$ が成り立つときである．しかし，$SENT(m) < SENT(m') \leqq \mathrm{RECV}[h]$ が成り立っているので，いったん m' が h へ送信されると，その後 h へ m が送信されることはない．これは，補題 4 に反している． □

5. プロトコルの評価

システムの MSS の総数を N_{mss}，MH の総数を N_{mh} とし，$N = N_{mss} + N_{mh}$ とする．提案するプロトコルにおいて，各放送メッセージへ付加するヘッダは，二つの N_{mss} 次ベクトルである．したがって前後関係保存放送を実現するためのメッセージオーバヘッドは $\Theta(N_{mss})$ で，MH の数に依存しない．

提案するプロトコルでは，MH への配達メッセージの欠落を避けるため，配達済みメッセージを各 MSS でキュー **DELIV_MES** に保存する．これらのメッセージは，すべての MH で配達されたことが確認されるとキューから削除される．以下では，このキューの大きさを評価する．ここで，各ハンドオフはある時間 ϵ 内に終了し，各 MSS は少なくとも時間 γ 内に一つの放送メッセージを送信し，MSS が送信した各放送メッセージは時間 δ 内にすべての MSS で受信されると仮定する．放送メッセージ m がある MSS で送信されてから削除されるまでの最悪時の時間を考える．m は送信されてから時間 δ 内にすべての MSS で配達され，接続中の MH へ m が送信される．このときハンドオフ中の MH は時間 ϵ 内にハンドオフ先の MSS と接続し，この MH へ m が送信される．つまり，m の送信後時間 $\delta + \epsilon$ 内に，すべての MH へ m が送信される．これ以降の放送メッセージ（m が削除可能であるという情報 *REDUCE* を含む）をすべての MSS から受信した MSS は m を削除する．したがって，各 MSS は各放送メッセージをたかだか $2\delta + \epsilon + \gamma$ 時間だけ記憶すればよい．提案するプロトコルでは，すべての放送メッセージに *REDUCE* を付加したが，**DELIV_MES** のための記憶領域に余裕があれば，適当な放送メッセージにのみ *REDUCE* を付加することでメッセージオーバヘッドを軽減できる．

多くの分散移動システムでは，MH の非接続化（MH の電力消費を節約するために MH とネットワークとの接続を断つこと）と，MH の再接続（システムに接続していない MH が新たにネットワークと接続すること）を許している．よって，このような MH の非接続化・再接続への対応が，簡単な拡張によって実現できるプロトコルが望ましい．提案したプロトコルで使用するメッセージのヘッダと管理情報は，すべて MSS の数に依存した大きさで MH の個数に依存しない．また，各 MH に対する情報は，それぞれ MH が接続している MSS でのみ管理すればよく，すべての MSS ですべての MH に対する情報をもつ必要はない．このように，提案したプロトコルは MH の非接続化・再接続に容易に対処できる．

6. むすび

本論文では，移動計算機と移動支援局を含む分散移動システム上で前後関係保存放送を実現するプロトコルを提案した．本論文で提案したプロトコルでは，前後関係保存放送を実現するための通信オーバヘッド（各メッセージへ添付する情報量）は移動支援局の数にのみ比例し移動計算機の数に依存しない．分散移動システムにおいて，移動計算機の数は一般的に移動支援局に比べ非常に多いため，プロトコル実行時の通信オーバヘッドが移動計算機の数に依存しないことは非常に望ましい性質である．また，提案したプロトコルでは，ハンドオフ時の処理に必要なメッセージ数や遅延時間が小さい．更に，移動計算機で必要な計算量，通信量も非常に小さく，処理能力の低い移動計算機が存在するシステムでの実行にも適している．

前後関係保存放送を実現するには，受信したメッセージの配達を遅らせなければならないことがある．前後関係を保証しない単純な放送に比べ，本論文のプロトコルでの配達がどれくらい遅くなるかは，興味深い問題であり，そのシミュレーションによる評価は今後の課題である．また，本論文のプロトコルでは，配達済みのメッセージを削除可能になるまで保存しておく必要がある．**5.** では放送メッセージを保存しておかなければならない時間の評価を行ったが，保存に必要なバッファのサイズなどについても，シミュレーショ

ンによる評価は今後の課題である．

謝辞　本研究に関し，多くの貴重な御意見をいただいた井上智生助手をはじめとする奈良先端科学技術大学院大学情報論理学講座の皆様，及び同大学情報科学センター片山喜章助手に感謝致します．本研究の一部は，文部省科学研究費補助金（奨励研究 A 09780279, 09780281）による．

文　献

[1] L. Lamport, “Time, clocks and the ordering of events in a distributed system,” CACM, vol.21, no.7, pp.558–565, 1978.

[2] F. Mattern, “Virtual time and global states of distributed systems,” Proc. of the Workshop on Parallel and Distributed Algorithm, pp.215–226, 1989.

[3] K. Birman and T. Joseph, “Reliable communication in the presence of failures,” ACM Trans. on Computer Systems, vol.5, no.1, pp.47–76, 1987.

[4] K. Birman, A. Schiper, and P. Stephenson, “Lightweight causal and atomic group multicast,” ACM Trans. on Computer Systems, vol.9, no.3, pp.272–314, 1991.

[5] B. Charron-Bost, “Concerning the size of logical clocks in distributed systems,” Information Processing Letters, vol.39, no.1, pp.11–16, 1991.

[6] M. Raynal, A. Schiper, and S. Toueg, “The causal ordering abstraction and a simple way to implement it,” Information Processing Letters, vol.39, no.6, pp.343–350, 1991.

[7] R. Prakash, M. Raynal, and M. Singhal, “An efficient causal ordering algorithm for mobile computing environments,” Proc. ICDCS, pp.744–751, 1996.

[8] Y. Sato, M. Inoue, T. Masuzawa, and H. Fujiwara, “A snapshot algorithm for distributed mobile systems,” Proc. ICDCS, pp.734–743, 1996.

[9] S. Alagar and S. Venkatesan, “Causal ordering in distributed mobile systems,” IEEE Trans. on Computers, vol.46, no.3, pp.353–361, 1997.

（平成 10 年 4 月 1 日受付，8 月 3 日再受付）

大堀　力　（学生員）

平 8 阪大・基礎工・情報卒．平 10 奈良先端技科大博士前期課程了．同年（株）NTT データ入社．

井上美智子　（正員）

昭 62 阪大・基礎工・情報卒．平 1 同大大学院博士前期課程了．同年富士通研究所（株）入社．平 7 阪大大学院博士後期課程了．奈良先端科学技術大学院大学情報科学研究科助手，現在に至る．分散アルゴリズム，グラフ理論，テスト容易化設計，高位合成の研究に従事．工博．IEEE，情報処理学会，人工知能学会各会員．

増澤　利光　（正員）

昭 57 阪大・基礎工・情報卒．昭 62 同大大学院博士後期課程了．同年同大情報処理教育センター助手．同大基礎工助教授を経て，平 6 奈良先端科学技術大学院大学情報科学研究科助教授，現在に至る．平 5 コーネル大客員準教授（文部省在外研究員）．分散アルゴリズム，並列アルゴリズム，テスト容易化設計，テスト容易化高位合成に関する研究に従事．工博．ACM，IEEE，EATCS，情報処理学会各会員．

藤原　秀雄　（正員）

昭 44 阪大・工・電子卒．昭 46 同大大学院博士後期課程了．阪大工学部助手，明大理工学部教授を経て，現在奈良先端科学技術大学院大学情報科学研究科教授．昭 56 ウォータールー大客員助教授．昭 59 マッギル大客員準教授．論理設計，高信頼設計，設計自動化，テスト容易化設計，テスト生成，並列処理，計算複雑度に関する研究に従事．著書 “Logic Testing and Design for Testability”（The MIT Press）など．大川出版賞．工博．IEEE，情報処理学会各会員．IEEE Fellow，IEEE Golden Core Member．