マルチメディア通信と分散処理 74－38
グループウェア 15－38
（1996. 1. 26）

分散システムにおける因果関係を保存するメッセージ配送プロトコル

真鍋 典行 多田 知正 樋口 昌宏 藤井 護

大阪大学基礎工学部 e-mail: {manabe, tada, higuchi, fujii }@ics.es.osaka-u.ac.jp

分散システムにおいて送信イベント間の因果関係を保存してメッセージを配送する機能を実現することは有用と思われる．本稿ではベクトルクロックとs-recordとよばれるタグをメッセージに付加することによりこの機能を実現するプロトコルを提案した．s-recordは送信側と受信側プロセスの識別子と，送信プロセスのローカルクロックによる送信時刻の3字組からなる．各メッセージには先行するすべてのメッセージのs-recordを付加し，受信側で付加されたベクトルクロックとs-recordと自身の保持しているベクトルクロックとs-recordを比較することにより上記機能が実現できることを示した．また，メッセージの受信確認などにより不要となったタグを消去できる条件についても議論した．

A Message Delivery Protocol Preserving Causal Order for Distributed Systems

Noriyuki Manabe Harumasa Tada Masahiro Higuchi Mamoru Fujii

Faculty of Engineering Science,
Osaka University

e-mail: {manabe, tada, higuchi, fujii }@ics.es.osaka-u.ac.jp

A message delivery system which preserves the causal order between sending events seems to be useful in distributed systems. In this paper, we propose a protocol which achieves such a message delivery system using vector time and message tags called "s-record". A s-record is a 3-tuple, sender's and receiver's identifiers, and sending time in sender's local time. In the protocol, every message contains vector time and s-records of all preceding messages, and the receiver compares those of receiving message and those stored in itself to achieve message delivery preserving causal order between sending events. We also discusses conditions to remove useless s-records.

1 まえがき

メッセージの交換によって互いに通信を行なう複数のプロセスからなる系を分散システムという．各プロセスはイベントの逐次実行を行ない，各プロセスが実行するイベントは(i)他のプロセスにメッセージを送信する送信イベント，(ii)他のプロセスが送信したメッセージを受信する受信イベント，に大別される．個々のプロセス間はそれぞれ双方向の通信路で接続されており，一方のプロセスの送信したメッセージは有限時間内に，相手プロセスに到着するものとする．

分散システムにおいて，global clockおよび同期local clockは一般には利用できない．異なるプロセスで発生したイベント間の時間的前後関係を知ることは非常に困難である．そのための一手法として，論理時間を用い，イベント間の因果関係(半順序)を決定する方法が知られているが，そのコストは大変大きなものとなっている．本稿では，より小さいコストにより，イベント間の因果関係(半順序)を決定する分散システムにおける因果関係を保存するメッセージ配送プロトコルを実現する方法を提案する．分散システムにおける因果関係を保存するメッセージ配送とは，2つの送信イベントを s, s' とし，それらのメッセージの受信イベントをそれぞれ r, r' として，下記の条件(1)の成立を保証することをいう．

$$\{(s \rightarrow s') \Rightarrow \neg(r' \rightarrow r)\} \qquad (1)$$

K.Birmanらがマルチキャスト通信の枠組の中で，すでに因果関係を保存したメッセージ配送を実現する方法を提案している．マルチキャスト通信とは，1度の送信

により，メッセージを複数のプロセスに送信できる通信のことをいう．この方法においては，プロセスはシステム内においてあるグループに属し，そのグループに属するすべてのプロセスに，メッセージをブロードキャストすることによりメッセージ交換を行なう．また，プロセスのイベント発生数をカウントするグループ毎の論理時間を個々のプロセスが持ち，メッセージを送信するとき，プロセスが保持している論理時間をメッセージに付加し，メッセージを受信するとき，メッセージに付加されている論理時間がある条件を満足するまで，そのメッセージの受信を遅らせることにより因果関係を保証する．そのメッセージには，システム内のすべてのグループの個数を n とすると $O(n)$ の論理時間が付加されている．因果関係を保存するメッセージ配送プロトコルは，この方法において，それぞれのプロセスが1対1にメッセージ交換を行なうようにすることで可能となる．1対1通信は，送信者1，受信者1からなるグループが N^2 個存在するマルチキャスト通信の特別な場合と考えられる．このため，この方法では，メッセージに N^2 のタグを付加しなければならず，大きな負荷となる．そこで，最悪の場合 N^2 のタグをメッセージに負荷する必要があるが，実際には N^2 よりかなり小さいタグをメッセージに付加することで (1) を保証する方法を提案する．まず，(1) を保証するのみの補助手続きを紹介し，その後，タグの個数を少なくする方法を提案する．

2. では，分散システムにおける定義，定理，系について述べる．3. では，因果関係を保存するメッセージ配送を実現する方法，CMD 法について述べ，4. では，今後の課題について述べる．

2 定義

分散システムは N 個のプロセス $P_1, P_2, \cdots, P_N$ からなり，分散システムの1つの実行において，プロセス P_i で発生したイベントの集合を E_i と表す．P_i は逐次実行を行ない，E_i は発生時刻に基づく全順序集合になっており，$E_i = \{e_{i,1}, e_{i,2}, \cdots\}$ と表した場合，$e_{i,k} < e_{i,l} \Leftrightarrow k < l$ となっている．$E = E_1 \cup E_2 \cup \cdots \cup E_N$ は分散計算のすべてのイベントの集合を表し，各 E_i は互いに素になっている．また，プロセス内のイベントには，メッセージの送信を行なう送信イベント，メッセージの受信を行なう受信イベントの2つがあり，あるメッセージ m の送信イベントを $s(m)$，受信イベントを $r(m)$ と表す．プロセスは送信先を指定してメッセージを送信し，送信されたメッセージは必ず送信先のプロセスで受信されるものとする．また，同じメッセージの送信イベントと受信イベントは非同期に実行されるものとする．

定義 1 任意のイベント e, e_1, e_2, e_3，メッセージ m に対して，因果関係 "$\rightarrow$" は E_i 上の順序を含み，以下の性質を満たす．

1. $s(m) \rightarrow r(m)$
2. $e \rightarrow e$ （反射性）
3. $\{(e_1 \rightarrow e_2) \wedge (e_2 \rightarrow e_3)\} \Rightarrow (e_1 \rightarrow e_3)$ （推移性）

ただし，"$\rightarrow$" が順序関係にならない場合，因果関係に矛盾が生じている場合に相当する．実際の分散システムの実行において，そのような場合は起こり得ず，以降では "$\rightarrow$" が順序関係となる場合のみを考える． □

定義 2 任意のイベント e, e_1, e_2 に対して，並行であるという関係 "$\|$" は以下の性質を満たす．

- $(e_1 \| e_2) \Leftrightarrow \{\neg(e_1 \rightarrow e_2) \wedge \neg(e_2 \rightarrow e_1)\}$. □

定義 3 イベント $e, e' \in E$ について，$e' \rightarrow e$ となる e' を e の先行イベントと呼び，また，$e' \rightarrow e$ となる送信イベント e' を e の先行送信イベントと呼ぶ．e の先行イベント集合 $C(e)$ を以下のように定義する．

- $C(e) = \{e' \in E | e' \rightarrow e\}$.

また，$C(e)$ の E_i への射影 $C(e)[i]$ を以下のように定義する．

- $C(e)[i] = \{e' \in E_i | e' \rightarrow e\}$

このとき，$e, e' \in E, e \neq e'$ とすると，以下の関係が成立する．

1. $e \rightarrow e' \Leftrightarrow e \in C(e')$
2. $e \| e' \Leftrightarrow (e \notin C(e')) \wedge (e' \notin C(e))$ □

補題 1 任意の e, i について $k = |C(e)[i]|$ とすると，$C(e)[i] = \{e_{i,j} | 1 \leq j \leq k\}$

略証 "$\rightarrow$" が E_i 上の順序関係を含むので，$m \leq n$ に対して，$e_{i,n} \in C(e)$ ならば，$e_{i,m} \in C(e)$ となることより導かれる． □

定義 4 n 次元ベクトルの集合 $\{v_1, v_2, \cdots, v_m\}$ に対する演算 $\sup\{v_1, v_2, \cdots, v_m\}$ は，次のように定義される．$i = 1, 2, \cdots, N$ に対して，$\sup\{v_1, v_2, \cdots, v_m\}[i] = \max\{v_1[i], v_2[i], \cdots, v_m[i]\}$. ここで，$v[i]$ はベクトルの第 i 成分を表す． □

vector time の計算法 あるイベント e が保持する vector time $V(e)$ はシステム内の全プロセス数を N とすると N 次元のベクトルで表される．また，P_i で最後に発生したイベントの vector time をプロセス P_i の vector time といい，V_i と書き，以下のように計算する．

1. 計算を実行する前に，$k = 1, 2, \cdots, N$ に対して，初期化 $V_i[k] := 0$ を実行する．
2. P_i がメッセージ m を送信する時，P_i は $V_i[i] := V_i[i] + 1$ とし，V_i を m に付加する．その送信イベントの vector time を V_i とし，m に付加される vector time を $V(m)$ と表す．
3. P_i がメッセージ m を受信する時，P_i は $V_i[i] := V_i[i] + 1$，$V_i[j] := \max\{V_i[j], V(m)[j]\} (i \neq j)$ とし，その受信イベントの vector time を V_i とする．

補題 2 上記で述べた vector time の計算法により得られた vector time と先行イベント集合との間には以下の関係が成立する．

- $\forall i (1 \leq i \leq N) [V(e)[i] = |C(e)[i]|]$.

証明 $|E|$ に関する帰納法で証明する．基底段階として $|E|=0$ のとき，イベントが1つもなく，題意は成立．$|E|=k$ のとき成立すると仮定する．$|E'|=k+1$ とする．"$\rightarrow$" は E' 上の順序関係なので，任意の $e' \in E$ について$\neg(e \rightarrow e')$ となる $e \in E$ が存在する．そのような e に対して，$E'-\{e\}=E''$ とすると，帰納法の仮定より，$|E''|$ について題意は成立する．$e=e_{i,p}$ として，$V(e_{i,p+1})=|C(e_{i,p+1})|$ が成立することを証明する．$e_{i,p+1}$ は送信イベント，もしくは，受信イベントのいずれかになっており，それぞれの場合について証明を行なう．

- $e_{i,p+1}$ が送信イベントのとき

  $e_{i,p+1}$ が送信イベントであるので，C の定義より $C(e_{i,p+1})=C(e_{i,p})\cup\{e_{i,p+1}\}$ となる．従って，$|C(e_{i,p+1})[i]|=|C(e_{i,p})[i]|+1$ となり，$l\neq i$ のとき，$|C(e_{i,p+1})[l]|=|C(e_{i,p})[l]|$．また，$P_i$ は $e_{i,p+1}$ を発生したとき，$V_i[i]:=V_i[i]+1$ を実行するので，$V(e_{i,p+1})[i]=V(e_{i,p})[i]+1$ となる．また，$l\neq i$ のとき，$V(e_{i,p+1})[l]=V(e_{i,p})[l]$．以上と帰納法の仮定より，$V(e_{i,p+1})[l]=|C(e)_{i,p+1}|$ が得られる．
- $e_{i,p+1}$ が受信イベントのとき

  $e_{i,p+1}$ が受信したメッセージ m とすると，$s(m)\in E''$ となっているので，帰納法の仮定より，$l=1,\cdots,N$ に対して，$V(m)[l]=|C(m)[l]|$．$e_{i,p+1}$ が受信イベントであるので，C の定義より，$C(e_{i,p+1})=C(e_{i,p})\cup\{e_{i,p+1}\}\cup C(s)$ となる．"$\rightarrow$" が E_i 上の全順序を含む順序関係であることより，$C(e_{i,p+1})[i]=\{e_{i,1},\cdots,e_{i,p+1}\}$ なので，$|C(e_{i,p+1})[i]|=p+1=|C(e_{i,p})[i]|+1=V(e_{i,p})[i]+1=V(e_{i,p+1})[i]$，$j\neq i$ について，$C(e_{i,p+1})[j]=C(e_{i,p})[j]\cup C(s(m))[j]$．ここで，補題1と帰納法の仮定より，$C(e_{i,p+1})[j]=\{e_{j,1},\cdots,e_{\max\{V(e_{i,p})[j],V(m)[j]\}}\}$ なので，$|C(e_{i,p+1})[j]|=\max\{V(e_{i,p})[j],V(m)[j]\}\ V(e_{i,p+1})[j]$． □

定義 5 u,v を m 次元の vector time とする．

1. $u\leq v \Leftrightarrow \forall i(1\leq i\leq m)(u[i]\leq v[i])$.
2. $u<>v \Leftrightarrow \neg(u<v)\wedge\neg(u>v)$. □

補題 3 2つのイベント e,e' に対して次の関係が成立する．

1. $e\rightarrow e' \Leftrightarrow V(e)\leq V(e')$.
2. $e\|e' \Leftrightarrow V(e)<>V(e')$.

証明 [1. の証明] $e\rightarrow e'$ が成立すると仮定する．関係 $\rightarrow$ の定義 の3. 推移性 と C の定義より，$C(e)\subseteq C(e')$ となる．従って，$k=1,\cdots,N$ に対して，$V(e)[k]=|C(e)[k]|\leq|C(e')[k]|=V(e')[k]$ となり，$V(e)\leq V(e')$ となる．

逆に，$V(e)\leq V(e')$ と仮定すると，$C(e)\subseteq C(e')$ である．定義 3より，$e\in C(e)$ なので，$e\in C(e')$ が導かれ，$e\rightarrow e'$ となる．
[2. の証明] 1. および 関係 $\|$ の定義より導くことができる． □

補題 4 $e\in E_i, e'\in E_j (e\neq e')$ となる2つのイベント e,e' に対して，次の2つの関係が成立する．

1. $e\rightarrow e' \Leftrightarrow V(e)[i]\leq V(e')[i]$.
2. $e\|e' \Leftrightarrow (V(e)[i]>V(e')[i])\wedge(V(e')[j]>V(e)[j])$.

証明 [1. の証明]
{$\Rightarrow$ の証明} 補題3より，直ちに導かれる．
{$\Leftarrow$ の証明} $e\in E_i$ なので，$e=e_{i,V(e)[i]}$ と書ける．補題1より $V(e)[i]\leq V(e')[i]$ なので，$e_{i,V(e)[i]}\in C(e')[i]$．よって，$e\rightarrow e'$ となる．
[2. の証明] $e\|e'$ であるので，関係 $\|$ の定義より，$\neg(e\rightarrow e')\wedge\neg(e'\rightarrow e)$．従って，1. より，$\neg(e\rightarrow e')\wedge\neg(e'\rightarrow e) \Leftrightarrow (V(e)[i]>V(e')[i])\wedge(V(e')[j]>V(e)[j])$ であるので，題意は証明された． □

3 因果関係を保存するメッセージ配送を実現するシステム

本研究で提案する因果関係を保存するメッセージ配送プロトコルを実現するシステムとは，2つのメッセージ m,m' に対して，以下の条件が成立するシステムのことをいう．

$$\{(s(m)\rightarrow s(m'))\Rightarrow\neg(r(m')\rightarrow r(m))\} \quad (1)$$

メッセージ配送システムは複数のエンティティを持ち，エンティティはそれぞれ分散システムにおけるプロセスと1対1に対応しており，2つのエンティティ間は1対1に完全二重結合をしているが，その通信路は FIFO である必要はない．メッセージの送信を行なう場合，プロセスは送信イベントを発生し，対応するエンティティに対して，メッセージの配送要求を出す．エンティティはプロセスからの配送要求を受けて，メッセージの送信を行なう．メッセージの受信を行なう場合，エンティティは別のエンティティからメッセージを受信する．プロセスがそのメッセージを受信することにより，(1) の条件が破壊されなければ，エンティティは対応するプロセスにメッセージを配送する．プロセスがそのメッセージを受信することにより，(1) の条件が破壊されるときは，メッセージを Queue に入れて，(1) の条件が破壊されなくなるまで，プロセスへのメッセージの配送を遅らせる．プロセスは受信イベントを発生し，対応するエンティティにより配送されたメッセージを受けとる．

3.1 補助手続き

定義 6 s-record とは3字組 $\langle i,j,t\rangle$ であり，$V(e)[i]=t$ となるプロセス P_j への送信イベント $e\in E_i$ が存在することを表す． □

[補助手続き] プロセス P_i,P_j と対応しているエンティティをそれぞれ S_i,S_j とする．また，S_i,S_j が保存している s-record の集合をそれぞれ R_i,R_j とし，メッセージ m に付加されている s-record の集合を $R(m)$ とする．

- メッセージを送信するとき

  S_i は P_i から，m を P_j へ送る配送要求を受け取ると，$V_i[i]:=V_i[i]+1$ とし，vector time および R_i の中のすべての s-record をメッセージに付加して送信する．メッセージを送信後，$\langle i,j,V_i[i]\rangle$ を R_i に加える．

- メッセージを受信する時

  S_j は別のエンティティから m を受信すると，m に，受信プロセスが j 自身である s-record $\langle i,j,t\rangle$ が付加されていれば，S_j は以下の (i),(ii) のいずれかを実行する．$\langle i,j,t\rangle$ が付加されていなければ，S_j は $V_j[j] := V_j[j]+1, V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，$R(m)$ の中のすべての s-record を R_j に加え，m を P_j に配送する．

  (i) $\forall\langle i,j,t\rangle \in R(m)[V_j[i] \geq t]$ のとき

  このとき，S_j は P_i が時刻 t に P_j に送ったメッセージ m' をすでに P_j に配送している．S_j は $V_j[j] := V_j[j]+1, V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，$R(m)$ の中のすべての s-record を R_j に加え，m を P_j に配送する．Queue の中のメッセージ m' が $\forall\langle i,j,t\rangle \in R(m')[V_j[i] \geq t]$ となると，S_j は $V_j[j] := V_j[j]+1, V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，m' を Queue から取り出し，$R(m')$ の中のすべての s-record を R_j に加え，m' を P_j に配送する．

  (ii) $\exists\langle i,j,t\rangle \in R(m)[V_j[i] < t]$ のとき

  このとき，S_j は P_i が時刻 t に P_j に送ったメッセージ m' をまだ P_j に配送していない．S_j は m を P_j に配送せず，Queue に入れる．□

補題 5 分散システムにおいて，条件 (1) が成立しないならば，$\{s(m) \rightarrow s(m')$ かつ $r(m') \rightarrow r(m)\}$ かつ $r(m), r(m') \in E_j$ となる 2 つのメッセージ m, m' が存在する．

証明 条件 (1) が成立しないので，2 つのメッセージ m_1, m_2 が存在し，$s(m_1) \rightarrow s(m_2)$ かつ $r(m_2) \rightarrow r(m_1)$ が成立する．その $r(m_1), r(m_2)$ の関係は以下の 2 つの関係のどちらかになっている．

1. $r(m_1), r(m_2) \in E_j$
2. $r(m_1) \in E_i, r(m_2) \in E_j (i \neq j)$

- 1. の場合

  題意は直ちに成立する．
- 2. の場合

  $r(m_2) \rightarrow r(m_1), r(m_2) \in E_j, r(m_1) \in E_i$ であるので，あるメッセージ m_3 が存在して，$r(m_2) \rightarrow s(m_3) \rightarrow r(m_3) \rightarrow r(m_1)$ $(r(m_3) \in E_i)$ となる．一方，仮定より，$s(m_1) \rightarrow r(m_2)$ である．従って，$s(m_1) \rightarrow s(m_3) \rightarrow r(m_3) \rightarrow r(m_1)$ $(r(m_3) \in E_i)$ となり，題意は成立する．□

補題 6 分散システムにおいて，条件 (1) が成立する必要十分条件は，すべてのメッセージ $m\big(s(m) \in E_i, r(m) \in E_j\big)$ に対して，$e \rightarrow r(m)$ となるすべての $e \in E_j$ について，次の条件 (2) が成立することである．

$$V(e)[i] < V(m)[i] \tag{2}$$

証明 [十分性の証明] 分散システムにおいて，すべてのメッセージ $m\big(s(m) \in E_i, r(m) \in E_j\big)$ に対して，$e \rightarrow r(m)$ となるすべての $e \in E_j$ について，$V(e)[i] < V(m)[i]$ が成立しているにも関わらず，条件 (1) が成立していないと仮定して，背理法により証明する．

いま，条件 (1) が成立していないので，補題 5 を満たすような 2 つのメッセージを m, m' とすると，そのメッセージについて，$\big(s(m) \rightarrow s(m')\big) \Rightarrow \big(r(m') \rightarrow r(m)\big)$ が成立する．$s(m) \in E_i$，$r(m), r(m') \in E_j$ とする．$\big(s(m) \rightarrow s(m')\big)$ と補題 4 より，$V\big(s(m)\big)[i] \leq V\big(s(m')\big)[i]$ となる．$r(m')$ は $s(m')$ に対応する受信イベントであるので，$s(m') \rightarrow r(m')$ となり，$V\big(s(m')\big)[i] \leq V\big(r(m')\big)[i]$ となる．従って，$V\big(s(m)\big)[i] \leq V\big(r(m')\big)[i]$ となる．$V\big(s(m)\big) = V(m)$ と併せて $V\big(r(m')\big)[i] \geq V(m)[i]$ となる．一方，$r(m') \rightarrow r(m)$ および，$r(m') \in E_j$ なので，条件 (2) より，$V\big(r(m')\big)[i] < V(m)[i]$ となり，矛盾が生じる．

[必要性の証明] 分散システムにおいて，条件 (1) が成立しているにも関わらず，あるメッセージ $m\big(s(m) \in E_i, r(m) \in E_j\big)$ に対して，$e \rightarrow r(m)$ となる $e \in E_j$ が存在し，$V(e)[i] \geq V(m)[i]$ となると仮定して，背理法により証明する．

P_j において，$V_j[i](j \neq i)$ の値を更新するのは，あるメッセージ m' を受信したときのみであるので，P_j は $V(m')[i] = V(e)[i]$ となるメッセージ m' を m を受信する以前に受信している．$V\big(s(m')\big) = V(m')$ より，$V\big(s(m')\big)[i] = V(e)[i]$ となる．一方，$V(e)[i] \geq V(m)[i] = V\big(s(m)\big)[i]$ なので，$V\big(s(m)\big)[i] \leq V\big(s(m')\big)[i]$ となり，補題 4 より $s(m) \rightarrow s(m')$ となる．また，$r(m') \rightarrow r(m)$ であるので，$\big(s(m) \rightarrow s(m')\big) \Rightarrow \big(r(m') \rightarrow r(m)\big)$ が成立する．これは，条件 (1) に反する．従って，仮定に矛盾が生じた．□

補題 7 補助手続きが任意の実行 D において，受信プロセスが同じであるような任意のメッセージ対 m_1, m_2 に対して，$\{s(m_1) \rightarrow s(m_2)\} \Rightarrow [\{r(m_1) \rightarrow r(m_2)\} \vee r(m_2) \notin D]$ が成立を保証する．

証明 補助手続きで得られる実行の集合を D として，実行 D に含まれるイベント数 n に関する帰納法により証明を行なう．

基底段階として，$n = 0$ のとき，イベントが 1 つもなく，題意は成立．D の中の p 個以下のイベントを持つ任意の実行における，受信プロセスが同じ任意のメッセージ対 m_1, m_2 に対して，$\{s(m_1) \rightarrow s(m_2)\} \Rightarrow [\{r(m_1) \rightarrow r(m_2)\} \vee r(m_2) \notin D]$ が成立していると仮定し，$p+1$ 個のイベントを持つすべての実行におけるすべてのメッセージ対について，題意が成立していることを証明する．ある p 個のイベントを持つある実行を D_1 とすると，帰納法の仮定として，D_1 について題意は成立すると仮定して，$D_2 - \{e\} = D_1$ となる D_2 についても，題意が成立することを証明する．いま，e は送信イベントもしくは受信イベントのいずれかになっており，それぞれの場合について証明を行なう．

- e が送信イベント $s(m)$ のとき

  D_1 中のすべての送信イベント $s(m')$ について，$s(m) \| s(m')$ のとき，$\neg\{s(m) \rightarrow s(m')\}$ かつ

$\neg\{s(m') \rightarrow s(m)\}$. $s(m) \rightarrow s(m')$ となる $s(m')$ は D_1 中に存在しない．$s(m') \rightarrow s(m)$ のとき，$r(m) \notin D_2$ である．以上と帰納法の仮定より，題意は成立．

- e が配送イベント $r(m)$ のとき

  $s(m') \rightarrow s(m)$ となる S_j 宛の m' は，すでに配送されていること，すなわち，$r(m') \in D_1$ を示したい．いま，$r(m') \notin D_1$ と仮定して背理法により証明する．

  m' を送信した s-record を $\langle k,j,t\rangle$ とすると，$s(m') \rightarrow s(m)$ なので，$\langle k,j,t\rangle \in R(m)$ である．m を配送する条件は，補助手続きより，$t \leq V_j[k]$ である．従って，S_j では $V(m'')[k] = V_j[k]$ となるメッセージ m'' をすでに配送している．いま，$s(m') \rightarrow s(m)$ より，$V(m')[k] \leq V(m)[k]$ ．また vector time の計算法より，$t = V(m)[k]$ なので，$V(m)[k] \leq V(m'')[k]$ となっている．さらに，$s(m') \in E_k$ なので，補題 4 より $s(m') \rightarrow s(m'')$ となり，$s(m') \rightarrow s(m'') \rightarrow r(m'')$ となる $r(m'') \in E_j$ が存在する．ところが，$r(m') \notin D_2$ であるので，帰納法の仮定に矛盾する．従って，$s(m') \rightarrow s(m)$ となる S_j 宛のすべての m' はすでに配送されており，題意は成立． □

定理 1 補助手続きが条件 (1) の成立を保証する．

証明 補題 6，補題 7 より，直ちに導かれる． □

3.2 エンティティが保存している s-record を取り除く方法

3.1のままでは，$|R_j|$ が分散計算の実行において，単調増加するため，メッセージ m に付加される $R(m)$ も単調増加し，メッセージの送受信のコストが大きくなり問題となる．ここでは，分散計算の実行において，増加する s-record を取り除き，$|R_j|$ を減少させる方法を提案する．

定義 7 プロセス P_i による s-record $\langle k,l,t\rangle$ の確認とは，あるプロセス P_j において，s-record $\langle k,l,t\rangle$ が表す送信イベントによって送られるメッセージを m とすると，m の受信イベント $r(m)$ を発生したか，もしくは，$r(m) \rightarrow e$ となるイベント e が発生したことをいう．また，イベント e を S_i において，s-record $\langle k,l,t\rangle$ の確認を行なったイベントという． □

補助手続きにおいて増大する s-record を以下のようにして，エンティティ S_j が P_j へ P_i の送信したメッセージ m を配送するとき，s-record を取り除くことにより，分散計算の実行において，エンティティが保存する s-record の個数を減らす．

1. R_j と $R(m)$ の中に，送信プロセスの識別子，受信プロセスの識別子がともに同じであるような s-record がある場合，時刻の値が小さい s-record を取り除く．
2. P_i が m を送信する以前に取り除いている s-record を P_j が R_j に保存していれば，その s-record を取り除く．
3. P_j がすでに取り除いている s-record が $R(m)$ の中に存在しても，P_j は，その s-record を R_j に加えない．

3.3 因果関係を保存するメッセージ配送を実現する方法 (CMD 法)

エンティティは分散計算の 1 つの実行において，先行送信イベントの s-record を保存しており，メッセージを送信するとき，保存しているすべての s-record をメッセージに付加して送信を行なう．また，メッセージを配送するとき，前節で提案した s-record を取り除く方法 1.,2.,3. を用いて，自身が保存している s-record の集合およびメッセージに付加されている s-record の集合から s-record を取り除く．その後，メッセージに付加されているすべての s-record をエンティティ自身が保存している s-record の集合に加える．

[CMD 法]

- メッセージを送信するとき

  S_i は P_i から，m を P_j へ送る配送要求を受け取ると，$V_i[i] := V_i[i] + 1$ とし，vector time および R_i の中のすべての s-record をメッセージに付加して送信する．メッセージを送信後，$\langle i,j,V_i[i]\rangle$ を R_i に加える．
- メッセージを受信するとき

  S_j は別のエンティティからメッセージ m を受信すると，$R(m)$ に $\langle i,j,t\rangle$ が存在すれば，S_j は以下の (i),(ii) のいずれかを実行する．$\langle i,j,t\rangle$ が存在しなければ，S_j は $V_j[i] := V_j[i] + 1$ とした後，$V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，$R(m)$ の中のすべての s-record を R_j に加え，m を P_j に配送する．P_j は 受信イベントを発生し，S_j から m を受けとる．

  (i) $\forall\langle i,j,t\rangle \in R(m)[V_j[i] \geq t]$ のとき

  このとき，S_j は P_i が時刻 t に P_j に送ったメッセージ m' をすでに P_j に配送している．S_j は $V_j[j] := V_j[j] + 1$ とした後，$V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，次の 1.,2.,3. を順に実行する．実行後，$R(m)$ の中のすべての s-record を R_j に加え，m を P_j に配送する．その後，Queue の中のメッセージ m' が $\forall\langle i,j,t\rangle \in R(m')[V_j[i] \geq t]$ となると，S_j は $V_j[j] := V_j[j] + 1$, $V_j[k] := \max\{V_j[k], V(m)[k]\}(k \neq j)$ を実行し，m' を Queue から取り出し，1.,2.,3. を実行する．実行後，$R(m)$ の中のすべての s-record を R_j に加え，m を P_j に配送する．

  1. $\forall\langle k,l,t\rangle \in R_j[(\langle k,l,t\rangle \in R_j) \wedge (\langle k,l,t'\rangle \in R(m))]$ のとき
     (a) $t \geq t'$ のとき
     $\langle k,l,t'\rangle$ を $R(m)$ から取り除く．
     (b) $t < t'$ のとき
     $\langle k,l,t\rangle$ を R_j から取り除く．

2. $(\langle k,l,t\rangle \in R_j) \wedge (\langle k,l,t\rangle \notin R(m)) \wedge (t \leq V(m)[k])$ のとき
   $\langle k,l,t\rangle$ を R_j から取り除く.
3. $(\langle k,l,t\rangle \notin R_j) \wedge (\langle k,l,t\rangle \in R(m)) \wedge (t \leq V_j[k])$ のとき
   $\langle k,l,t\rangle$ を $R(m)$ から取り除く.

(ii) $\exists t\langle i,j,t\rangle \in R(m)[V_j[i] < t]$ のとき
このとき, S_j は P_i が時刻 t に P_j に送ったメッセージ m' をまだ P_j に配送していない. S_j は m を P_j に配送せず, Queue に入れる. □

定理 2 CMD 法が条件 (1) の成立を保証する.

証明 定理 1 より s-record の個数を減らさない場合, 計算の過程において, 条件 (1) の成立を保証しているので, 計算の過程において s-record の個数を減らしたとしても同じ動作をすることを 1.,2.,3. の場合について証明を行なう.

[1. の証明] そのような 2 つの s-record を $\langle i,j,t\rangle$,$\langle i,j,t'\rangle$ $(t<t')$ とし, $\langle i,j,t\rangle$,$\langle i,j,t'\rangle$ により送信されるメッセージをそれぞれ m,m' とする. 補助手続きにおいて, エンティティはすべての先行送信イベントの s-record を保存しているため, P_j へ送信されるメッセージには s-record $\langle i,j,t\rangle$,$\langle i,j,t'\rangle$ がともに付加される. そのようなメッセージを m'' とすると, S_j が m'' を P_j へ配送するとき, 提案した CMD 法より $V_j[i] \geq t$ かつ $V_j[i] \geq t'$ であることを判定する必要があるが, いま, $t < t'$ であるので, $V_j[i] \geq t' \Rightarrow V_j[i] \geq t$ が成り立つ. 従って, $\langle i,j,t\rangle$ はメッセージに付加する必要はない.

[2. の証明] $\langle k,l,t\rangle$ により送信されるメッセージを m とすると, $\langle k,l,t\rangle \in R_i$ かつ $\langle k,l,t\rangle \notin R_j$ かつ $V_j[k] \geq t$ ならば $r(m) \in E$ が成立することを補助手続きで得られる実行の集合を D として実行 D に含まれるイベント数 n に関する帰納法により証明を行なう.

基底段階として, $n=0$ のとき, イベントが 1 つもなく, 題意は成立. D の中の p 個以下のイベントを持つ任意の実行において題意が成立していると仮定し, $p+1$ 個のイベントを持つすべての実行における題意が成立していることを証明する. ある p 個のイベントを持つある実行を D_1 とすると, 帰納法の仮定として D_1 について題意は成立すると仮定して, $D_2 - \{e\} = D_1$ となる D_2 についても, 題意が成立することを証明する. D_1 から D_2 へ変化したときに, vector time の値および R 中の s-record 変更されるのは 1 つのプロセスにおいてのみであるので, そのプロセスに着目する. e は, 送信イベントもしくは受信イベントのいずれかになっており, それぞれの場合について証明を行なう.

- e が送信イベントのとき

  $e \in E_p$ がメッセージ m を送信するとする. m を送信するとき, e が V_q $(p \neq q)$ を更新することはなく, $V_p[p] := V_p[p]+1$ とするのみであり, また, R_q 中の s-record を変更することはできず, R_p に $s(m)$ の s-record を加えるのみであるので, $s(m)$ 以外の s-record については題意は成立. $s(m)$ の s-record について考える. いま, $s(m) \rightarrow e'$ となるイベント e' が存在しないので, 補題 3 より, $V(s(m)) \leq V(e')$ となる e' は存在しない. よって, $V(s(m))[k] \leq V(e')[k]$ となる e' は存在しないので, $V(s(m))[k] = t$ より, $V_j[k] \leq t$ となることはない. 従って, 題意は成立.

- e が受信イベントのとき

  $r(m) = e \in E_p$ $(s(m) \in E_q)$ とする. $s(m)$ の s-record については $r(m)$ が存在するので題意は成立. m 以外のメッセージを m' とし, $s(m')$ の s-record を $\langle k,l,t\rangle$ とする. D_1 において, $\langle k,l,t\rangle$ について先件が成立するとき, すでに, $r(m')$ が存在するので題意は成立. 先件のうち, このとき, $\langle k,l,t\rangle \in R_i$ のみ成立し, どのプロセス P_j についても $\langle k,l,t\rangle \in R_j \vee V_j[k] < t$ であるとする. D_1 において, $\langle k,l,t\rangle \in R_p$ のとき, m を受信して D_2 において, $\langle k,l,t\rangle \notin R_p$ と仮定する. このとき, 送信プロセス側において m の送信時に, $\langle k,l,t\rangle \in R_q$ かつ $V_q[k] \geq t$ が成立する必要がある. 一方, あるメッセージを受けて, 一旦取り除いた s-record をあらたに保存することはなく, また, $V_q[k]$ の値は単調増加することにより, D_1 においても, $\langle k,l,t\rangle \in R_q$ かつ $V_q[k] \geq t$ が成立し, 仮定に反するので, m を受信して $\langle k,l,t\rangle \notin R_p$ となることはない. D_1 において, $V_p[k] < t$ のとき, m を受信して D_2 において, $V_p[k] \geq t \vee \langle k,l,t\rangle \notin R_p$ と仮定する. このとき, 送信プロセス側において m の送信時に, $\langle k,l,t\rangle \notin R_q$ かつ $V_q[k] \geq t$ が成立する必要がある. これは, 先に見たようにどのプロセス P_j についても, $\langle k,l,t\rangle \in R_j \vee V_j[k] < t$ であるという仮定に反する. 先件のうち, $\langle k,l,t\rangle \notin R_p$, $V_j[k] < t$ が成立するとき, $\langle k,l,t\rangle$ が存在しないので, あらたに $\langle k,l,t\rangle \in R_p$ となることはない. すなわち, 2. により $\langle k,l,t\rangle$ が P_i から取り除かれるのは $\langle k,l,t\rangle$ に対応する受信イベントがすでに発生している場合である.

[3. の証明] 2. の証明と同様. □

4 まとめ

本稿では 1 対 1 通信における因果関係を保存するメッセージ配送を実現する方法を提案した. また, マルチキャスト通信においても 1 対 1 通信の場合と同様の方法を用いることにより可能である. 今後の課題として, 提案した因果関係を保存するメッセージ配送を実現するシステムのシミュレーションによる定量的評価の実現を目指す.

参考文献

[1] Kenneth Birman , Andre Schiper and Pat Stephenson: "Lightweight Causal and Atomic Group Multicast". Distributed computing, vol.7, No.3, pp. 149 – 174,(1994).

[2] Reinhard Schwarz and Rriedmann Mattern: "Detecting causal relationships in distributed computations: in search of the holy grail". ACM Transactions on Comuter Systems, Vol.9, No.3, pp 272 – 314,(August 1991).