マルチメディア通信と分散処理 86－16
グループウェア 26－16
（1998. 1. 29）

待ち時間を考慮した Δ 因果順序配送アルゴリズムの提案

吉本 忠司 多田 知正 樋口 昌宏 藤井 護

大阪大学大学院基礎工学研究科

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

多くのリアルタイムアプリケーションではしばしばメッセージが送信されてから受信されるまでの制限時間 Δ が設定される．この制限時間 Δ を考慮した Δ 因果順序メッセージ配送アルゴリズムが考えられている．各メッセージには Δ 因果順序配送を行うために必要な情報がタグとして付加される．これまで，配送可能になったメッセージはすべてすぐに配送されるがタグが大きくなるというアルゴリズムや，タグは小さく抑えられるがどのメッセージも送信されてから一定時間以上配送されないというアルゴリズムなどが提案されている．本稿では，タグの大きさを抑えることと，多くのメッセージについて到着から配送までの時間を短縮することを共に考慮したΔ 因果順序配送アルゴリズムを提案する．

Δ-causal ordering algorithm that considered delivery time

Tadasi Yosimoto Harumasa Tada Masahiro Higuchi Mamoru Fujii

Graduate School of Engineering Science,
Osaka University

In many real time applications, messages have to be delivered to the destinations by a deadline, and a message which have been passed its deadline is discarded. The deadline of a message is defined as being Δ time units after the message is sent. Several Δ-causal ordering algorithms have been proposed in the literature. Though, some algorithms ensure that all messages will be delivered as soon as they can be delivered, they induce high message size overhead. Others impose low message size overhead, but any message won't be delivered until some fixed time passed since it is sent. This paper presents an algorithm for Δ-causal ordering, which is designed to suppress message size overhead, and to insure that many messages will be delivered as soon.

1 まえがき

分散システムにおいて，各メッセージの送信順序と受信順序の間の整合を取る事は重要である．どの 2 つのメッセージに対しても，送信イベントの因果順序 (causal order [1]) と逆の順序が受信イベントに対して成立しないことを保証する因果順序配送が考えられている．しかし，[1] ではメッセージの消失や制限時間は考慮されていない．

リアルタイムアプリケーションでは，メッセージが送信されてから受信されるまでの制限時間 Δ が設定されている場合があり，送信されてから Δ 時間以上経って受信されたメッセージは捨てられる．そこで，制限時間 Δ を考慮した因果順序配送が考えられている [2].

因果順序配送は，必要な情報を各メッセージにタグとして付加することにより実現される．プロセス数を N とすると，最悪時のタグの大きさが $O(N^2)$ となるアルゴリズムが提案されている [2]. このアルゴリズムを用いると，各メッセージは，それより前に受信されなければならないすべてのメッセージがプロセスに配送されたか，制限時間を超えた時点で即座に配送することができる．一方，タグの最悪時の大きさを $O(N)$ に抑えたアルゴリズムが提案されている [3]. しかしこのアルゴリズムでは，どのメッ

セージも送信されてから一定時間経つまでは配送されない.

本稿では，最悪時のタグの大きさを抑えることと，多くのメッセージについて到着から配送までの時間を短縮することを共に考慮した，因果順序配送アルゴリズムを提案する.

以降，2. で因果順序配送について説明し，3. では提案する因果順序配送アルゴリズムとその正当性を示し，4. で提案アルゴリズムの性能について考察する.

2 準備

2.1 因果順序

分散システムは非同期通信を行う N 個のプロセス $P_i(1 \leq i \leq N)$ からなるとする. 各プロセスで起きるイベントはメッセージの送信イベント，受信イベントおよび内部イベントの 3 種類であり，プロセスはこれらを逐次実行するものとする. また，各メッセージの送信先は 1 つ以上のプロセスであるとする. メッセージ m の送信イベント，受信イベントをそれぞれ $s(m)$, $r(m)$ と表す.

定義 1 2 つのイベント e_1, e_2 に対して，因果順序 (causality) $e_1 \rightarrow e_2$ が成り立つのは，以下の条件のいずれかが成り立つ時，かつその時のみである [1].

1. e_1 と e_2 は同じプロセスにおけるイベントであり，e_1 の次に e_2 が起きた.
2. あるメッセージ m に対して，$e_1 = s(m)$, $e_2 = r(m)$ である.
3. $e_1 \rightarrow e$ かつ $e \rightarrow e_2$ を満たすイベント e が存在する. □

定義 2 2 つのメッセージ m_1, m_2 に対して，$s(m_1) \rightarrow s(m_2)$ であるとき，かつその時のみ，$m_1 \rightarrow m_2$ とする. □

定義 3 分散システムのある実行において，任意の 2 つのメッセージ m_1, m_2 について $m_1 \rightarrow m_2 \Rightarrow \forall r(m_1), r(m_2)\Big(\neg(r(m_2) \rightarrow r(m_1))\Big)$ であるとき，その実行がメッセージの因果順序を保存しているという. □

因果順序を保存した実行について，以下の性質が知られている.

定理 1 分散システムのある実行において，同じプロセス P_i で受信される任意の 2 つのメッセージ m_1, m_2 について，$m_1 \rightarrow m_2$ ならば P_i が m_1 を m_2 よりも前に受信しているとき，その実行はメッセージの因果順序を保存している.

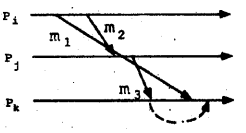

図 1: 因果順序を保存しない実行

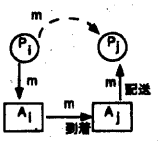

図 2: システムモデル

一般にメッセージの通信には遅延があるため，メッセージの因果順序を保存しない実行が起こり得る. 例として，図 1 を考える. この例では P_k の受信するメッセージ m_1, m_3 について $m_1 \rightarrow m_3$ であるにもかかわらず，$r(m_3) \rightarrow r(m_1)$ となっており，メッセージの因果順序を保存した実行ではない.

このような因果順序を保存しない実行を排除するためには，メッセージの受信を遅らせる機能が必要になる. 例えば，図 1 の場合 m_3 の受信を破線のように遅らせてやればよい. そこで，各プロセス P_i に対して通信エージェント A_i が存在し，プロセスは通信エージェントを介して通信するとする (図 2). ここで，各 P_i, A_i $(1 \leq i \leq N)$ 間は FIFO，A_i, A_j $(i \neq j)$ 間は非同期通信とする. これにより，受信側の A_j は送られて来たメッセージ m をバッファに入れておくことで，P_j における m の受信を遅らせることが出来る. 以降では，メッセージ m の受信プロセスを P_j とすると，A_j に m が届くことを到着するといい，A_j が P_j に m を渡すことを配送するという.

2.2 制限時間が設定されている場合

リアルタイムアプリケーションではメッセージが送信されてから受信されるまでの制限時間 Δ が設定されている場合がある. そのような場合，送信されてから Δ 時間以上経って受信されたメッセージは古くて利用価値がないとして捨てられる.

以降では，A_i が m を配送すると同時に P_i において $r(m)$ が起きるものとする. また，A_i に Δ 時間以上経って到着したメッセージ m は P_i に配送することなく A_i が捨て，P_i において $r(m)$ は起きないものとする.

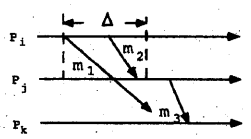

図 3: 制限時間 Δ が設定されている場合

制限時間 Δ が設定されている場合に，制限時間内に配送されたメッセージについての因果順序を保存した実行を保証することを考える．メッセージ m が送信された時刻を $SendTime(m)$ とする．$m_1 \rightarrow m_3 \wedge SendTime(m_1) < SendTime(m_3) - \Delta$ となる m_1，m_3 を考える (図 3)．m_3 が到着するのは m_1 の制限時刻より後であり，m_1 は制限時間内に到着し配送されたなら必ず m_3 より前に配送されている．したがって m_3 は，制限時間内に到着すれば，m_1 がいつ到着するかに関係なく配送することができる．

定義 4 2 つのメッセージ m_1，m_2 に対して，$m_1 \stackrel{\Delta}{\rightarrow} m_2$ が成り立つのは，以下の条件が成り立つとき，かつその時のみである [2]．

1. $m_1 \rightarrow m_2$ かつ
2. $SendTime(m_1) + \Delta \geq SendTime(m_2)$ □

定理 2 制限時間 Δ が設定されている場合，同じプロセス P_i へ制限時間内に配送された任意の 2 つのメッセージ m_1，m_2 について，$m_1 \stackrel{\Delta}{\rightarrow} m_2$ ならば P_i が m_1 を m_2 よりも前に受信している時，その実行はメッセージの因果順序を保存している．

3 提案アルゴリズム

3.1 従来の手法

各メッセージ m は，$m' \stackrel{\Delta}{\rightarrow} m$ を満たすすべての m' が配送されたか制限時間を超えた時点で配送することができる．この時点の事を，m が配送可能になった時点という．また，配送可能になった時点から実際に配送されるまでの時間を不必要な待ち時間という．

因果順序に従った配送を行うためには，制御情報をメッセージにタグとして付加する．

各メッセージは配送可能になった時点ですぐに配送することができる因果順序配送アルゴリズムが提案されている [2]．しかし，プロセス数を N とすると，タグの大きさは最悪時 $O(N^2)$ にまで大きくなってしまう．

一方，制限時間 Δ についての知識を利用することによりタグを小さく抑えるアルゴリズムが提案されている [3]．このアルゴリズムでは $\Delta/3 < w < \Delta/2$ を満たす定数 w を定め，どのメッセージも送信されてから w 時間経つまでは配送しない．これにより，タグの大きさを最悪時 $O(N)$ に抑えているが，すでに配送可能になった時点を過ぎていたとしても，送信されてから w 時間経つまでは m を配送できない．

そこで本稿では，タグの大きさに上限を定め，多くのメッセージについて不必要な待ち時間がかからない，因果順序配送アルゴリズムを提案する．

3.2 データ構造

各 A_i は局所時計の値を変数 $time_i$ で保持する．ここでは，すべての局所時計は完全に同期しているとする．局所時計が大まかに同期していると仮定した場合への拡張法は後述する．

各 A_i は配列 $CB_i[1 \ldots N]$ を持つ．$CB_i[k]$ は，将来 A_i が送信するメッセージ m について $m' \stackrel{\Delta}{\rightarrow} m$ となる，P_k へ送られた m' についての，$(sender_{m'}, TS_{m'})$ を保持する．ここで，$sender_{m'}$，$TS_{m'}$ はそれぞれ，m' の送信プロセスおよび送信時刻である．A_i が送信するメッセージ m には，送信時における CB_i が CB_m として付加される．m が A_k に到着したとき，A_k は $CB_m[k]$ により，$m' \stackrel{\Delta}{\rightarrow} m$ となる P_k へ送られた m' について知る事ができる．

各 A_i は配列 $deliv_i[1 \ldots N]$ を持つ．各 $deliv_i[k]$ は，P_k が送信したメッセージのうち P_i へ配送された最も新しいメッセージ m の送信時刻を保持する．$deliv_i$ は A_i に到着したメッセージ m について $m' \stackrel{\Delta}{\rightarrow} m$ となる m' がすでに P_i へ配送されているかどうかを判定するために用いられる．

プロセス P_i が m を送信する時，m の送信先をプロセスの集合として $dest_m$ に設定した後，A_i に m と $dest_m$ を送るものとする．ここで，$|dest_m| \geq 1$ とする．$|dest_m| \geq 2$ ならば m は $dest_m$ 中のプロセスにマルチキャストされる．

また，提案アルゴリズムはタグの大きさの上限を決定する定数 MAX_CB を用いる．

3.3 アルゴリズム

各 A_i がメッセージの送信時，到着時に以下の手続きを行うことにより，因果順序を保存した実行を保証する．

- m を $dest_m$ へ送信するとき

  ($S1$) $TS_m := time_i$ とする．

  ($S2$) 各 $k(1 \leq k \leq N)$ について以下を行なう．
  $(l, x) \in CB_i[k] \wedge x + \Delta < time_i$ を満たすすべての (l, x) を $CB_i[k]$ から取り除く．

(S3) 各 $k(1 \leq k \leq N)$ について以下を行なう．
$|CB_i[k]| > MAX\_CB$ ならば，
$(l,x) \in CB_i[k]$ のうち，x の大きい方から MAX_CB 個までを残し，それ以外を $CB_i[k]$ から取り除く．

(S4) すべての $P_j \in dest_m$ に対して，A_j へ $(m, CB_i, i, TS_m, dest_m)$ を送信する．

(S5) すべての $P_j \in dest_m$ に対して，$CB_i[j] := \{(i, TS_m)\}$ とする．

- $(m, CB_m, j, TS_m, dest_m)$ が到着したとき

(R1) $TS_m + \Delta < time_i$ ならば m を捨てて終了する．

(R2) $|CB_m[i]| < MAX\_CB$ ならば，下記の条件 (1) が真となるまで待つ．
$|CB_m[i]| = MAX\_CB$ ならば，$\exists (l,x) \in CB_m[i](x + \Delta < time_i)$ かつ条件 (1) が真となるまで待つ．

(1) $\forall (l,x) \in CB_m[i] \quad (deliv_i[l] \geq x \vee x + \Delta < time_i)$

(R3) m を P_i へ配送する．

(R4) すべての $P_k \in dest_m - \{P_i\}$ に対して，$CB_i[k] := CB_i[k] \cup_{max} \{(i, TS_m)\}$ とする．

(R5) すべての $P_k \notin dest_m$ に対して，$CB_i[k] := CB_i[k] \cup_{max} CB_m[k]$ とする．

(R6) $deliv_i[j] := TS_m$ とする．

- $T_1 \cup_{max} T_2$ は以下で定義される．

1. $T := T_1 \cup T_2$ とする．
2. 各 $i(1 \leq i \leq N)$ に対して以下を行なう．
$(i,x) \in T \wedge (i,y) \in T \wedge x > y$ を満たす (i,x), (i,y) が存在するなら，(i,y) を T から取り除く．
3. T を $T_1 \cup_{max} T_2$ とする．

3.4 正当性の証明

定義 5 m, m' について $m' \stackrel{\Delta}{\rightarrow}_d m$ が成り立つのは以下の条件がともに成り立つ時，かつその時のみである．

1. $m' \stackrel{\Delta}{\rightarrow} m$ である．
2. m, m' は同じプロセス P_j へ送信された．
3. $m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす P_j へ送信された m'' が存在しない．
4. $m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす P_j から送信された m'' が存在しない．

補題 1 P_i から P_j へ送信されたメッセージ m について $(k,x) \in CB_m[j]$ とする．このとき P_k から P_j へ送られ，$TS_{m'} = x$ かつ $m' \stackrel{\Delta}{\rightarrow}_d m$ を満たす m' が存在する．

[略証] $P_i = P_k$ の場合．(S5)(R4) より，ある (i,x) がメッセージに付加されるのは，P_i から送られ $TS_{m'} = x$ となるメッセージが存在するときのみであり，この m' は m よりも前に P_i から送信されたメッセージでなければならない．以上と (S2) より，$m' \stackrel{\Delta}{\rightarrow} m$ である m' が存在する．$m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす，P_j へもしくは P_j から送られた m'' が存在するなら，(S5)(R4) より $(i, TS_{m'}) \notin CB_m[j]$ である．したがって $m' \stackrel{\Delta}{\rightarrow}_d m$ が成り立つ．

$P_i \neq P_k$ の場合．(S5)(R4) より P_k から P_j へ送られ，$TS_{m'} = x$ なる m' が存在する．また，(S5)(R4)(R5) より $m' \rightarrow \ldots \rightarrow m$ なる系列が存在する．これと (S2) より $m' \stackrel{\Delta}{\rightarrow} m$ である．$m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす，P_j へもしくは P_j から送られた m'' が存在するならば (S5)(R4) より $(k, TS_{m'}) \notin CB_m$．したがって $m' \stackrel{\Delta}{\rightarrow}_d m$ である． □

補題 2 m は P_j へ送信されたとし，m' は P_k から P_j へ送信されたとする．$m' \stackrel{\Delta}{\rightarrow}_d m$ ならば以下のいずれかが成立する．

1. $(k, TS_{m'}) \in CB_m[j]$
2. $|CB_m[j]| = MAX\_CB \wedge$
$\forall (l,x) \in CB_m[j](TS_{m'} \leq x)$

[略証] $m' \stackrel{\Delta}{\rightarrow}_d m$ であるので，$m' \stackrel{\Delta}{\rightarrow} m_1 \stackrel{\Delta}{\rightarrow} \ldots \stackrel{\Delta}{\rightarrow} m_n = m$ を満たし，かつどの $m_i (1 \leq i \leq n-1)$ も P_j へ送信されておらず，P_j から送信されてもいないメッセージ系列が存在する．$(k, TS_{m'})$ がどの m_i の送信時の (S3) においても $CB_{m_i}[j]$ から取り除かれなければ，$(k, TS_{m'}) \in CB_m[j]$ である．ある m_s の送信時の (S3) において $(k, TS_{m'})$ が $CB_{m_s}[j]$ から取り除かれたとすると，$|CB_{m_s}[j]| = MAX\_CB \wedge \forall (l,x) \in CB_{m_s}[j](TS_{m'} \leq x)$ である．$TS_{m'} + \Delta < TS_m$ であるので，すべての $s \leq t \leq n$ について $|CB_{m_t}[j]| = MAX\_CB \wedge \forall (l,x) \in CB_{m_t}(TS_{m'} \leq x)$． □

補題 3 P_j から P_i へ送られ制限時間内に配送され

た m を考える．$m' \stackrel{\Delta}{\rightarrow}_d m$ を満たすすべての m' について，m' が P_i へ配送されるならばその配送は m の配送よりも前である．

[証明] 補題1，補題2より，$m' \stackrel{\Delta}{\rightarrow}_d m$ を満たす m' が MAX_CB 個未満なら $CB_m[j]$ はこれらすべてについての $(sender_{m'}, TS_{m'})$ からなり，MAX_CB 個以上ならば $CB_m[i]$ はこれらの中で送信時刻の新しい方から MAX_CB 個についての $(sender_{m'}, TS_{m'})$ からなる．

m が制限時間内に A_i に到着した時の $(R2)$ を考える．$|CB_m[i]| < MAX\_CB$ ならば $m' \stackrel{\Delta}{\rightarrow}_d m$ を満たすすべての m' について $(sender_{m'}, TS_{m'}) \in CB_m[j]$ であるので，条件 (1) は m' が配送されるならば，その配送は必ず m の配送より前であることを保証する．$|CB_m[i]| = MAX\_CB$ ならば $m' \stackrel{\Delta}{\rightarrow}_d m$ かつ $(sender_{m'}, TS_{m'}) \notin CB_m[i]$ なる m' が存在する．しかし，このような m' はすべて $\forall (l,x) \in CB_m[i](TS_{m'} \leq x)$ であり，$\exists (l,x) \in CB_m[i](x + \Delta < time_i)$ ならば $TS_{m'} + \Delta < time_i$ を満たす．したがって $\exists (l,x) \in CB_m[j](x + \Delta < time_i)$ と条件 (1) により，$m' \stackrel{\Delta}{\rightarrow}_d m$ なるすべての m' について，m' が配送されるならば，その配送は必ず m の配送より前であることが保証される． □

定理 3 提案アルゴリズムを用いた時，その実行は因果順序を保存している．

[証明]

定義5より，同じプロセス P_j へ制限時間内に配送された，$m' \stackrel{\Delta}{\rightarrow} m$ を満たす任意の2つのメッセージ m，m' を考えると，これらの間には以下の条件のいずれかが成り立つ．

1. $m' \stackrel{\Delta}{\rightarrow}_d m$ である．
2. $m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす P_j へ送信された m'' が存在するが，$m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす P_j から送信された m'' は存在しない．
3. $m' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ を満たす P_j から送信された m'' が存在する．

1. の場合，補題3より m' は m より前に配送される．

2. の場合，$m' \stackrel{\Delta}{\rightarrow}_d m_1 \stackrel{\Delta}{\rightarrow}_d \ldots \stackrel{\Delta}{\rightarrow}_d m_n \stackrel{\Delta}{\rightarrow}_d m$ $(n \geq 1)$ を満たすメッセージ系列が存在する．すべての m_k $(1 \leq k \leq n)$ が制限時間内に到着したならば，補題3より m' は m より前に配送される．制限時間内に到着しない m_s が存在するならば，$TS_{m'} + \Delta < TS_{m_s} + \Delta$ であり，m が配送されるのは $TS_{m_s} + \Delta$ 以降であるため，m' は m より前に配送される．

3. の場合，$m' \stackrel{\Delta}{\rightarrow} m''' \stackrel{\Delta}{\rightarrow} m'' \stackrel{\Delta}{\rightarrow} m$ となる P_j から送信された m''' が存在しないような，P_j から送信された m'' が存在する．ここで，m' の配送が m'' の送信の後であったと仮定すると，$m' \stackrel{\Delta}{\rightarrow}_d m_1 \stackrel{\Delta}{\rightarrow}_d \ldots \stackrel{\Delta}{\rightarrow}_d m_n \stackrel{\Delta}{\rightarrow} m''$ $(n \geq 1)$ となるメッセージ系列が存在し，m_n は m' より前に配送されなければならない．しかしこれは 2. の場合の証明より起こり得ない．したがって，m' の配送は m'' の送信の後である．これと，$m'' \stackrel{\Delta}{\rightarrow} m$ であることから，m' は m より前に配送される．

以上により，提案アルゴリズムを用いた時，その実行は因果順序を保存されることが証明された． □

定理 4 提案アルゴリズムを用いたとき，制限時間内に到着したメッセージはすべて配送される．

[証明] $m' \stackrel{\Delta}{\rightarrow}_d m$ ならば $TS_{m'} + \Delta < TS_m + \Delta$ が成り立つことより明らか． □

3.5 大まかに同期した局所時計

分散システムにおいて，完全に同期した局所時計は一般には利用できない．任意の局所時計の時刻差が定数 ϵ 以下である，大まかに同期した局所時計を実現するアルゴリズムがすでに提案されている [4]．そこで，大まかに同期した局所時計を用いる場合に，提案アルゴリズムを拡張する．

大まかに同期した局所時計を用いた場合，次のような問題が生じる．ある時点において $time_i > time_j$ であったとする．この時点で P_i が m を P_j，P_k へ送信すると，m が P_j に配送された時点で $time_j < TS_m$ となる可能性がある．さらに P_j が m の受信後すぐに P_k へ m' を送信したとすると，$TS_{m'} < TS_m$ かつ $m \stackrel{\Delta}{\rightarrow} m'$ となる．ここで，m' の制限時刻において，m' は A_k へ到着しており m は A_k に到着していないとすると，A_k は以下のどちらか選択しなければならない．(i) m' を配送し，m が制限時間内に到着しても捨ててしまう．(ii) m' は制限時間内に到着しているが捨ててしまい，m が制限時間内に到着したなら m を配送する．A_k がどちらを選択したとしても，制限時間内に到着したにもかかわらず捨てられなければならないメッセージが存在してしまう．

しかし，この問題は通信エージェントの局所時計の現時刻より大きな時刻印を持つメッセージが到着したとき，自身の局所時計がメッセージの時刻印に達するまで，そのメッセージを配送しないことで解決することができる．そこで大まかに同期した局所時計を用いる場合への提案アルゴリズムの拡張は以下のように行えばよい．

- 局所時計の時刻差 ϵ を制限時間 Δ に取り込むため，$\Delta := \Delta + \epsilon$ とする．

- $(m, CB_m, j, TS_m, dest_m)$ が A_i に到着したとき，まず $(R0)$ を実行した後，前述の $(R1)-(R6)$ を行う．

  $(R0)$ $time_i < TS_m$ ならば $time_i = TS_m$ となるまで待つ．

拡張を行った提案アルゴリズムの正当性については，3.4節と同じあるので省略する．

4 提案アルゴリズムの性能

4.1 考察

提案アルゴリズムにおける，最悪時のタグの大きさは $O(MAX\_CB \times N)$ であり，定数 MAX_CB を小さく取ることによりタグの大きさを抑えることができる．

提案アルゴリズムを用いたときの，不必要な待ち時間を考える．P_i へ送られたメッセージ m について，$|CB_m[i]| < MAX\_CB$ であるなら条件 (1) が真になったとき，すなわち $m' \stackrel{\Delta}{\rightarrow}_d m$ なるすべての m' が配送されたか制限時刻を超えた時点で即座に配送され，不必要な待ち時間はかからない．$|CB_m[i]| = MAX\_CB$ であるなら，$m' \stackrel{\Delta}{\rightarrow}_d m$ であるすべての m' が配送されたか制限時刻を超えていたとしても，$\exists (l, x) \in CB_m[i](x + \Delta < time_i)$ が真になるまで配送できない．したがって MAX_CB を大きく取れば，不必要な待ち時間のかかるメッセージ数は少なくなる．

4.2 シミュレーション

提案アルゴリズムの性能は定数 MAX_CB の値により変動する．そこでシミュレーションにより，定数 MAX_CB のアルゴリズムへの影響について調べた．

シミュレーション条件は以下のように定めた．まず，メッセージはすべてシングルキャストであり，送信先は全プロセスの中からランダムに選択される．メッセージが送信されてから到着するまでの遅延時間は正規分布に従うものとし，Δ を遅延時間の平均の 5 倍に，分散は捨てられるメッセージの割合が 0.01% となるように定めた．

送信されたメッセージのうち，$CB_m[sender_m] = MAX\_CB$ となったメッセージ，不必要な待ち時間がかかったメッセージの割合，および，不必要な待ち時間の平均の Δ に対する割合を求めた．それぞれ，$rate_{MAX}$，$rate_{wait}$，$rate_{w\_time}$ として，シミュレーション結果を図 4. に示した．

提案アルゴリズムを用いたとき，最悪時のタグの大きさを抑えることと，不必要な待ち時間がかかるメッセージ数を減らすことは，トレードオフになっていることがわかる．しかし，$CB_m[sender_m] = MAX\_CB$ であるメッセージであっても，必ずしも不必要な待ち時間はかかっていない．これは，m は到着時に $m' \stackrel{\Delta}{\rightarrow} m$ となる m' を待たなければならないが，m の到着時刻 T について $TS_{m'} + \Delta < T$ となる m' が多くあるなら，それらすべてについての情報を付加しても無駄が多く，提案アルゴリズムではこの無駄が軽減されていることを示している．したがって，タグの大きさをおさえ，なおかつ多くのメッセージについて不必要な待ち時間がかからない MAX_CB の値を取ることが可能である．

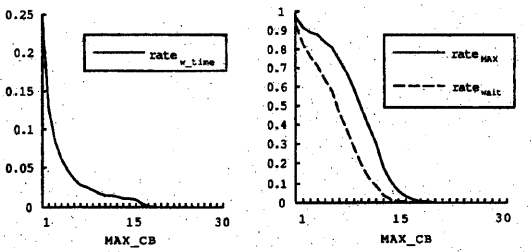

図 4: シミュレーション結果

5 まとめ

タグの大きさに上限を定め，多くのメッセージについて不必要な待ち時間がかからない，因果順序配送アルゴリズムを提案した．また，シミュレーションを行い，定数 MAX_CB のアルゴリズムへの影響について調べた．

今後は，具体的なアプリケーションにおける通信をモデル化し，そのモデルのもとでの MAX_CB の影響について調べることで，提案アルゴリズムの有効性について考察する予定である．

参考文献

[1] L. Lamport, "Time,Clocks,and the Ordering of Events in a Distributed System", Comm. ACM, Vol.21, No.7, pp.558-565, 1978.

[2] R. Yavatkar, "MCP: A Protocol For Coordination and Temporal Synchronization in Multimedia Collaborative Applications", Proc. of IEEE ICDCS-12, pp.606-613, 1992.

[3] F. Adelstein and M. Singhal, "Real-Time Causal Message Ordering in Multimedia Systems", Proc. of IEEE ICDCS-15, pp.36-43, 1995.

[4] R. Gusella and S. Zatti, "The Accuracy of the Clock Syschronization Achieved by TEMPO in Berkley UNIX 4.3BSD", IEEE Trans. Software Eng., Vol.15, No.7, pp.847-853, 1989.