マルチメディア通信と分散処理　90－19
コンピュータセキュリティ　2－19
（1998．9．18）

# Peer to Peer型 Replicationによる disconnected operation

山口実靖　　相田 仁　　齊藤 忠夫

東京大学工学系研究科電子情報工学専攻

モーバイルコンピューティング環境などユーザ同士が常時接続されていない状態で一つのドキュメントを複数の人間で編集する場合、接続時に変更を他のユーザに伝えなくてはならない。オブジェクト指向アプリケーションへのアプローチとして変更オブジェクト化システム提案する。提案方式により理論上で最も速い変更の反映を行うことができ、提案方式の評価を行ったところ最大で約７倍の反映の速度を実現が可能であった。

# Disconnected Operation with Peer to Peer type Replication system

Saneyasu Yamaguchi　　Hitoshi Aida　　Tadao Saito

Department Information And Communication Engineering, Faculty of Engineering, University of Tokyo

If we edit one document in mobile computing, we have to update changes when connected. We propose Modification Object System for object oriented applications. This system enables the fastest propagation.

## 1　はじめに

### 1.1　想定している問題

モーバイルコンピューティングなどの普及により、ディスコネクティッドオペレーションなど分散した環境(非常時接続)での協調作業が行われることが多くなった。そして、一つのドキュメントを複数の人間で編集することも多くなった。

協調作業を円滑に進めるには以下の問題の解決が重要となる。

1. 各ユーザが行った処理を他のユーザにどのようにして伝えて行くか。
   (a) ユーザはどのユーザと情報を交換して行くべきかの“接続の方針”
   (b) 接続が可能であったとき、どのように情報を交換して行くかの“情報交換の方針”
2. ドキュメントの意味をどのようにして保っていくか。
   (a) 各のユーザの処理が伝わったとして、それぞれのユーザが別々の変更を施したとき、それらの意味的な衝突をどのようにして検出して行くかの“矛盾検出の方針”
   (b) 各のユーザの処理が衝突しているとき、それらの意味的な矛盾をどのようにして解決して行くかの“矛盾解決の方針”

本稿ではオブジェクト指向アプリケーションにおける問題(1b)の“情報交換の方針”について述べていく。

### 1.2　提案する方式

1.1の問題(1b)に対するアプローチとして“変更オブジェクト化システム”による Peer to Peer型 Replicationを提案し、それを実装した。本稿で提案する方式は

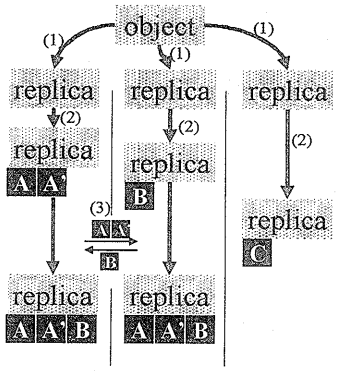


図 1: 変更オブジェクト化方式

“情報交換の方針” のみであり、“接続の方針” とは独立である。

変更オブジェクト化システムではあるオブジェクトのインスタンスを各ユーザで共有する。そして、そのインスタンスに対して行った処理を記録しておき、その処理を交換することにより、全ユーザで同じ状態を保持しする。

そのアルゴリズムを第 2 章で示す。

## 1.3 本稿の貢献

提案方式の変更オブジェクト化方式により、理論上で最も速いディスコネクティッドオペレーションの伝搬を可能にすることができる。実際にどの程度効率的な情報交換が行えるかを第 4章に示す。そして、少人数のときに得られるメリット、多人数のときに得られるメリットの違いについて述べる。

# 2 変更オブジェクト化システム

## 2.1 変更オブジェクト化方式

### 2.1.1 概要

以下の手法により、同じドキュメントの保持、共有ドキュメントの変更、その変更の通知を行う (図 1 参照)。

(1) 各ユーザは同じオブジェクトの複製 (レプリカ) を保持して切断された状況になる。以後、それぞれがローカルに保持しているレプリカオブジェクトを用いて処理を続ける。

(2) 各ユーザはドキュメントを編集するにあたって、保持しているレプリカに直接変更を施さずに、変更を表すオブジェクト ( 変更オブジェクト) を作成する[1]。“未変更のレプリカ”+“変更” で新しい状態を表現する。

(3) ある 2 ユーザ間でコネクションが確立し、情報を交換する機会に恵まれたら、お互いに持っている変更を交換する (詳しくは第 2.1.2 章で説明)。

(4) ある変更がこれ以上保留する必要がないことが保証されたら、その変更をレプリカオブジェクトに対して実際に反映させてしまい[2]、その変更を表すオブジェクトを消滅させる (詳しくは第 2.3 章で説明)。

### 2.1.2 情報の交換

ある 2 ユーザ A,B 間のコネクションが確立し情報交換の機会に恵まれたら、ユーザ A は以下のことを行う。

1. ユーザ A はユーザ B の保持していない変更オブジェクトをユーザ B に教える。自分の行った変更の他に他人の行った変更も含む[3]。
2. 逆にユーザ A はユーザ B から教わる。
3. 変更がユーザ B に伝わったので、その事実を記録する[4](詳しくは第 2.3.1 章で説明)。

## 2.2 一意性の保証

“保持している変更オブジェクトが同じなら、同じ状態を表している” ことが保証されなくてはならない。保証できないと、変更を交換しただけでは各ユーザが同じ状態のドキュメントを保持していることにならない。

## 2.3 オブジェクトの消滅

変更を変更オブジェクトとして保持し保留したままにしておくと、変更オブジェクトが単調に増加していつかは溢れてしまう。そこで、これ以上保留する必要の無いものはレプリカに対して変更を施してしまい[5]、その変更オブジェクトを消滅させる。

### 2.3.1 変更が全員に伝搬されたことの保証

変更オブジェクトを消滅させるためにはその変更オブジェクトを他の全ユーザが知っていることが保証されなくてはならない。伝搬の保証の管理方式として次の 2 通りが考えられる (表 1 参照)。

#### 伝搬自己管理方式 および 伝搬保証情報交換方式

**伝搬自己管理方式** ユーザ $x$ の行った変更を他のユーザがどこまで知っているかの情報をユーザ $x$ のみが保持する。ユーザ $x$ がユーザ $y$ とコネクションを確立し変更オブジェクトを教えたときにのみ伝搬を保証する情報[6]を更新する。直接接続したユーザへの伝搬のみ検出される。

[1] 変更の反映は保留しておく。レプリカは変更されない

[2] レプリカは変更される

[3] ユーザ A が以前に他のユーザから教わった変更オブジェクトで、ユーザ B がまだ知らないものも教える

[4] 「ユーザ $x$ がユーザ $y$ の変更をどこまで知っているか」 の情報を更新する

[5] レプリカは変更される

[6] 「ユーザ $y$ がユーザ $x$ の変更をどこまで知っているか」 の情報

表 1: 伝搬自己管理方式 と 伝搬保証情報交換方式

| 管理方式 | 自己管理 | 情報交換 |
|---|---|---|
| 伝搬管理 | 作成者のみ | 全員 |
| 全伝搬 | 本人のみ検出 | 全員が検出 |
| 記憶領域 | $o(n)$ | $o(n^2)$ |
| 保証される伝搬 | 直接の接続のみ | 間接接続も含む |

**伝搬保証情報交換方式**　ユーザ $x$ の行った変更をユーザ $y$ がどこまで知っているかの情報を全てのユーザが保持する。ユーザ同士で変更オブジェクトの交換をしたとき、交換相手の情報も更新し、さらに伝搬を保証する情報[7]を交換する。間接的に接続したユーザへの伝搬も検出され、可能である検出は必ず行え、理論的に最も正確な検出方法。

### 2.3.2　これ以上保留しなくて良いことの保証

全員に伝わったことが保証されてもその変更オブジェクトを消滅させて良いとは限らず、一意性の保証の仕方に依存する (第3章参照)。

## 2.4　実装

提案方式の変更オブジェクト化システム (Modification Object System) を Java により実装した (Mos.java)。また、 Mos.java 用 Client を実装した (Mc.java)。

## 3　例

図 2の時刻 0 ～ 5 に例を示す。伝搬管理は自己管理方式を用いた。ユーザ A,B,C,D の 4 人が同じオブジェクトに対するレプリカを保持する。ユーザ A の左半分はユーザ A が B,C,D のどの状態まで知っているかを表す。右半分はユーザ A のどの状態までを B,C,D に伝えたかを表している。他のユーザ (B,C,D) も同様。

図 2-0 : ユーザ A,B,C,D がすべて同じ状態のレプリカを保持してディスコネクトした。

図 2-1 : 時刻 1 においてユーザ A と B の接続が確立された。これにより、ユーザ A は時刻 1 までのユーザ B を知り、ユーザ B は時刻 1 までのユーザ A を知る (左半分を更新)。そして、ユーザ A はユーザ B に対して時刻 1 までの A の状態を伝えたことが保証され、ユーザ B はユーザ A に時刻 1 までの B の状態を伝えたことが保証される (右半分を更新)。

図 2-2 : 時刻 2 においてユーザ C と D の接続が確立された。ユーザ C は時刻 2 までのユーザ D を知り (左半分を更新)、時刻 2 までの C の状態をユーザ D に伝えた (右半分を更新)。ユーザ D も同様。

図 2-3 : 時刻 3 においてユーザ B と C の接続が確立された。ユーザ B は時刻 3 の C と時刻 2 の D を教わり、ユーザ C は時刻 3 の B と時刻 1 の A を教わる。そして、ユーザ B がユーザ C に時刻 3 までの B の状態を伝えたことが保証され、ユーザ C がユーザ B に時刻 3 までの C の状態を伝えたことが保証される。

[7]「任意のユーザが任意のユーザの変更をどこまで知っているか」の情報

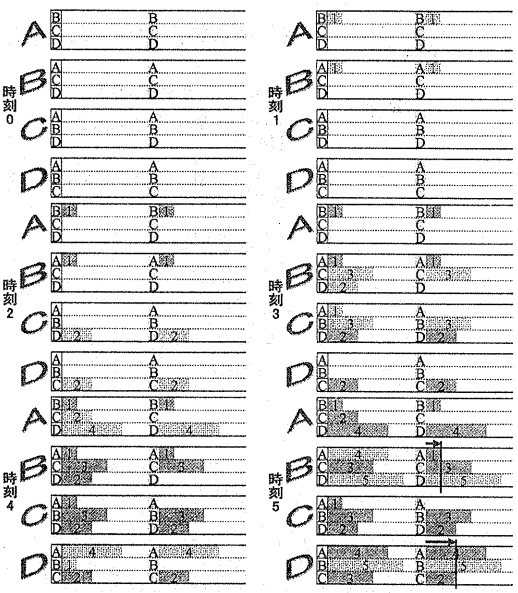


図 2: 伝搬例

図 2-4 : 時刻 4 においてユーザ A と D が接続。同様に、情報を交換し状態を更新する。

図 2-5 : 時刻 5 においてユーザ B と D が接続。同様に、情報を交換し状態を更新する。ここで、時刻 1 までの B および時刻 2 までの D はすべての人に伝わったことが保証された。よって B は次回以降は他のユーザと接続を行う度に “時刻 1 までの B はすべての人に伝わったのでこれ以上他人に伝える必要がない” ということを知らせる。しかし、残念ながらこれが “変更オブジェクトを消滅させてよい” ということを意味するとは限らない。例えば、変更オブジェクト (原因) から状態 (結果) への一意性を保証するのに「タイムスタンプ順に変更を反映させる」方法を採用した場合は、ある変更オブジェクトを消滅させるにはその変更オブジェクトよりも前のタイムスタンプの変更がすべてなされていなくてはならない。

この時点での消滅を可能にするには、変更 (原因) から状態 (結果) の一意性だけの保証でなく、変更の反映の順序に関わらず一意な結果が得られることを保証しなくてはならない。

## 4　提案方式の評価

提案方式は、伝搬可能な全ての情報を伝搬しているので理論上で最速の伝搬を可能にしているが、他の方法と比べて実際にどの程度速くしているかを述べる。

## 4.1　伝搬の評価値

以下に定義する伝搬に対する評価値を用いて比較を行う。

単位時間に 1 人当たり n[個 / 人 · 秒] の処理が誕生するとする。

$$\text{最大総伝搬数} = \text{総発生処理数} \cdot \text{ユーザ数} \quad (1)$$

$$\text{総伝搬数} = \sum_{\text{全処理}} \text{その処理を知っている人数} \quad (2)$$

$$= \sum_{\text{全ユーザ}} \text{知っている処理数} \quad (3)$$

$$\leq \text{最大総伝搬数} \quad (4)$$

$$\text{伝搬速度} = \frac{\Delta\text{総伝搬数}}{\Delta\text{時間}} \quad (5)$$

$$\text{総未伝搬処理数} = \text{最大総伝搬数} - \text{総伝搬数} \quad (6)$$

ここで、

$$\text{伝搬速度} \rightarrow n \cdot \text{ユーザ数}\ as\ t \rightarrow \infty \quad (7)$$

$$\text{総未伝搬処理数} \rightarrow \text{定数}x\ as\ t \rightarrow \infty \quad (8)$$

となる。伝搬速度は必ず $n \cdot$ ユーザ数 に収束するので収束した状態で比較することに意味がない。収束した状態での総未伝搬処理数が最も重要な評価値になると考えられる。

本方式の様に他人から受け取った情報も間接的に伝えていく方式と、直接あったユーザの処理のみを交換する方式の以下の値を比較する。

1. 収束した状態での総未伝搬処理数 (第 4.3 章)
2. 過渡状態での総伝搬数 (第 4.4 章)

## 4.2 シミュレーション条件

ユーザ数は 3 ～ 1000 の範囲で変化させた。現実的処理数として各ユーザ数で 1000 ステップ (1000 接続) までシミュレーションを行った。接続の発生および接続相手の選択に偏りはないものとした。

## 4.3 収束した状態での総未伝搬処理数

### 4.3.1 結果

未伝搬処理数の収束値はユーザ数により図 3のように変化する[8]。図において縦軸は未伝搬処理数なので少ない程優れていることになる。

間接伝搬の利用による効果は図 4の様になる[9]。ユーザ数が 36 人の時が最大であり、総未伝搬処理で 6.98 倍の効果がある (未伝搬処理数を $\frac{1}{6.98}$ に抑えている)。ユーザ数が増えるにしたがって未伝搬処理数の減少の効果は少なくなっている。

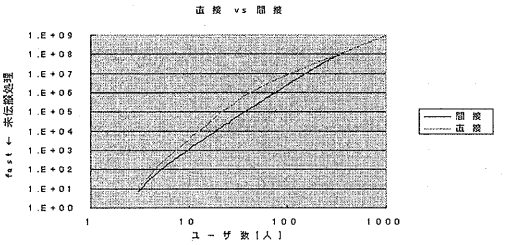


図 3: 直接伝搬のみ方式 と 間接伝搬も含む方式 (提案方式) の未伝搬処理数の収束値

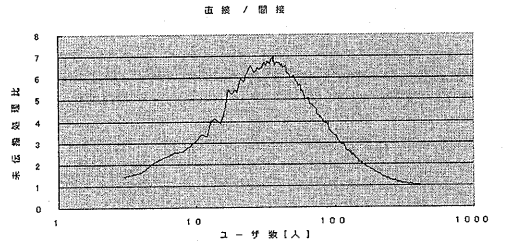


図 4: 直接伝搬のみ方式 と 間接伝搬も含む方式 (提案方式) の未伝搬処理数の収束値の比

[8] ただし、ユーザ数が多くなると完全に収束しいっていない。第 4.5章参照

[9] 縦軸は、収束状態での未伝搬処理数の比。値が大きい程未伝搬処理数の減少が激しいことを示す

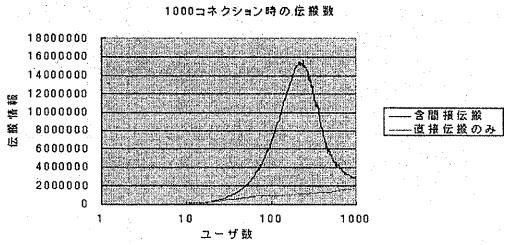


図 5: 過渡状態における 直接伝搬のみ方式 と 間接伝搬も含む方式 (提案方式) の伝搬処理数

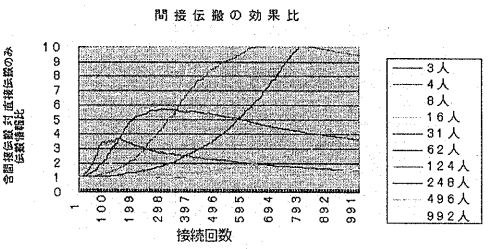


図 7: 伝搬情報比 ($\frac{\text{含間接}}{\text{直接のみ}}$) の推移

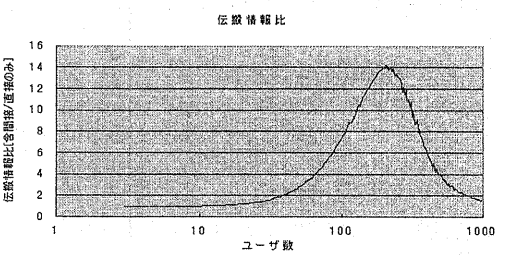


図 6: 過渡状態における 直接伝搬のみ方式 と 間接伝搬も含む方式 (提案方式) の伝搬処理数の比

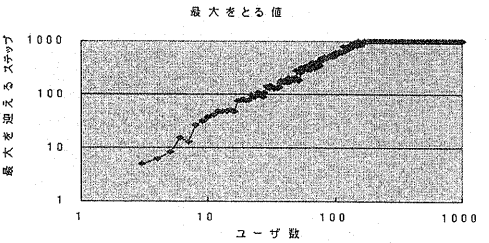


図 8: 最大値を迎えるステップ数

#### 4.3.2 評価

ユーザ数が 30 人前後で非常に効果があることが示された。ユーザ数が増えるにしたがって未伝搬処理数の減少の効果は少なくなっていることについては第 4.4 章で述べる。

### 4.4 過渡状態での総伝搬数

#### 4.4.1 比較の目的

収束する前の過渡状態ではユーザの行った処理がまだ全員に行き渡っていない。過渡状態ではほとんどの情報が未伝搬となり、未伝搬処理数を比較することはあまり意味がない。図 3, 図 4 ではユーザ数の多い箇所は (シミュレーションを行った範囲では) 過渡状態にあり、

$$\text{最大総伝搬数} \simeq \text{総未伝搬処理数} \gg \text{総伝搬数}$$

のため、共に総未伝搬処理数が 100% 近くなり差がでない。そこで過渡状態における総伝搬数を比較する。

#### 4.4.2 結果

伝搬処理数は図 5 , 図 6 の様になっており、最大でユーザ数が 208 人の時の総伝搬数が 14.22 倍の効果が得られる (総伝搬数を 14.22 倍に増やしている)。

しかし、ユーザ数がこれ以上増えると、あまり効果がでなくなってしまっている。

#### 4.4.3 評価

ユーザ数が増えると効果がでなくなっている原因として、ユーザが多すぎると他人と処理情報交換をした時にまだ相手が間接的情報をあまり持っていない可能性が高くなり、間接情報を扱う価値が下がっていると思われる。間接情報を扱う効果がでるまでにはどの程度の伝搬がおきている必要があるかは第 4.5 章で述べる。

### 4.5 過渡状態での伝搬の情報の推移

#### 4.5.1 結果

ステップ数 (コネクション回数) に対する総伝搬情報の利得の推移は図 7の様になっている。図 7よりユーザ数が多い程、効果が現れる (利得が増える) のが遅く、最大効果が大きいことがわかる。いつ最大値を迎えるかは図 8に示す。

#### 4.5.2 評価

第 4.4.3 章で述べたようにユーザ数が多すぎると効果がでるのに非常に時間がかかる。図 7, 図 8の様にユーザ数が 180 ～ 196 を境に利得の最大値がシミュレーションの範囲を超えている。それ以上の人数での効果を確認するには今回 (1000 ステップ) 以上のシミュレーションを行う必要がある。しかし、現実にそれらの人数で作業をする時は過渡状態のまま作業を行うのが現実的で収束を迎えることは少ないので過渡状態での評価が重要となる。

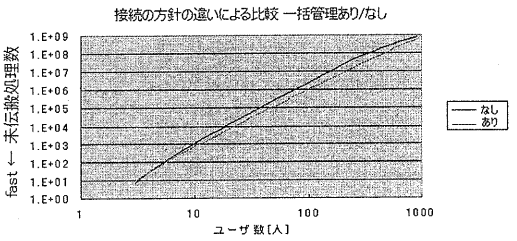


図 9: 接続の方針 一括管理あり / なしの未伝搬処理数

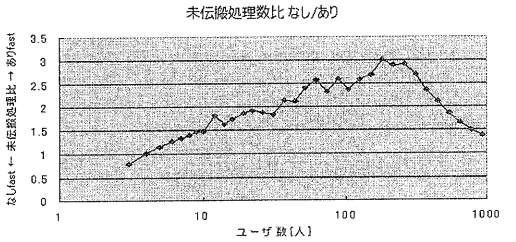


図 10: 接続方式一括管理ありとなしの未伝搬処理数の比

### 4.6 情報交換の方針の効果と接続の方針の効果

与えられた接続での "情報交換の方針" での効果について考察をしたが、"接続の方針" による効果との比較を行う。

中央に変更の一括管理を行うデータベースを設けて情報交換を行う場合と、"無作為に出会った人" 同士が情報交換を行う場合の比較を行う。情報交換の方針は間接伝搬も行う提案方式を用いた。

図 9, 図 10より中央管理データベースを用いた場合の方が良い結果となっている。これは中央の一括管理データベースと接続した方が多くの情報が得られることが原因と思われる。

得られた効果は最大で 3.00 倍 (ユーザが 181 人時) となった。

このことから、接続の方針だけが支配的でないことがわかる。接続の方針と比較しても、情報交換の方式は十分重要となる。

### 4.7 評価のまとめ

以下の事実が得られた。

1. 間接伝搬を用いる提案手法ではユーザ 36 人前後で、未伝搬処理数の収束値を約 $\frac{1}{7}$ に減らすことが可能であることが分かった。
2. 収束を迎えない程多い人数であっても 200 人前後では過渡状態での総伝搬数を 14 倍程度に増やせることが分かった (ただし、未伝搬処理数はほとんど減らせない)。
3. それ以上多いユーザ数 (1000 人以上) では間接伝搬情報を利用することのメリットが少なくコストに見合った効果が得られることはあまり期待できない。

## 5 おわりに

### 5.1 まとめ

本稿では "変更オブジェクト化システム" を提案し、説明した。本方式を用いることにより変更情報を理論上で最も速く伝搬させることが可能となる。

### 5.2 今後の課題

1. 実際にディスコネクティッドオペレーションを行うと偏ったユーザ間でしか情報の交換が行われないことが予想される。その弊害の大きさおよび解決策の考察をする。
2. 伝搬中で未確定の処理を取消するアンチ命令をサポートする。

## 参考文献

[1] 白鳥 則郎, 滝沢 誠: "分散処理", 丸善株式会社, 1996.

[2] Anthony D.Joseph, Joshua A. Tauber and M. Frans Kaashoek, "Mobile Computing with the Rover Toolkit", *IEEE Transactions on Computing*, pages 337-352, March 1997.

[3] David A. Nichols, Pavel Curtis, Michael Dixon and John Lamping, "High-Latency, Low-Bandwidth Windowing in the Jupiter Collaboration System", *in Proceedings of ACM Symposium on UIST '95 Pitt. PA.*, November 1995.

[4] Douglas B. Terry 他, "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Strorage System", December 1995. availableat `http://www.parc.xerox.com/csl/projects/bayou/pubs/sosp-95/BayouConflictsSOSPPreprint.ps`

[5] Alois Fercha. Parallel and Distributed Simulation of Discrete Event System. In Albert Y.Zomaya, editor, *Parallel and Distibuted Computing Handbook*, chapter 35, pp 1003-1041. McGraw-Hill, 1996.

[6] Louis Thomas, Sean Suchter, Adam Rifkin "Developing Peer-to-Peer Application on the Internet : the Distributed Editor, SimulEdit"California Institute of Technology

[7] 西田 豊明, "ソフトウェアエージェント", 人工知能学会誌 vol.10 No.5 pp 704-711, Sept. 1995.