P2P 技術を利用した分散システム上の実時間操作共有システム

山之上 卓†

P2P 技術を利用して，多数の端末コンピュータのアプリケーション操作を，その利用者間で，実時間で共有するシステムについて述べる．P2P 技術とネットワークスイッチを利用することによって，端末数が N の場合，$O(\log N)$ の遅延時間で，1 つの端末で行われる操作をすべての端末で表示することができる．端末間で操作を共有するためには，同時に複数の端末で異なる操作が行われることがないようにしなければならない．これを実現するために，最大で $O(\log N)$ の時間で critical section に入ることができる排他制御アルゴリズムを組み込んでいる．40 台の端末を使って，本システムと同じアプリケーションを使用するクライアント–サーバ型のシステムと性能を比較したところ，マウス操作を行った場合は P2P 技術を利用したほうが遅延が短かった．遠隔地間で操作を共有し，ゲームを行うこともできた．

A System Which Shares the Common Operation on a Distributed System in Realtime Using P2P Technology

Takashi Yamanoue†

A sytem, which shares the common operation of applications on many terminals of a distributed system in realtime using P2P technology, is shown. This system can show an operation on a terminal to every terminal in the latency of at most $O(\log N)$ time complexity, where N is the number of terminals, using a P2P technology and a switching network. In order to share a common operation on computer terminals, at most one operation must be executed on the terminals at a time. In order to realize this, a mutual exclusion algorithm is embedded in this system. The time complexity of entering the critical section is $O(\log N)$. We have compared the performance of this system with the performance of a client-server system which has the same applications using 40 terminals. The latency of our system was shorter than the client-server system when a mouse was moving. A game could be played by remote users using this system.

1. はじめに

近年，LAN やインターネットに複数のコンピュータを接続して構成された分散システム上で，業務や教育を行うことが一般的になっている．職場では，1 人 1 台のパソコン端末の上で，グループウェアを使った情報の共有や，電子会議が日常的に行われている．教育現場では，多数のパソコン端末を持った端末教室で様々な科目の授業が行われており，社会人教育などを行うために，インターネットを使った遠隔教育もさかんに行われるようになっている．このような分散システム上で，実時間で共同作業を行うことができれば，業務や教育をより効果的に行うことが期待できる．このため従来から，分散システム上で，実時間で共同作業を行うためのシステムが数多く研究されてきた．

実時間で共同作業を行うためには，使用する複数の端末の操作を共有する必要が生じる．端末の操作を共有するためには，同時に複数の端末で異なる操作が行われることがないよう，排他制御が必要となる．テキスト編集を共有するのであれば，1 台のサーバに複数の端末を接続し，このサーバで排他制御を行うことによって，かなり多くの台数の端末間で操作を共有できる．しかしながら，描画の操作を共有する場合，マウスの動作などを実時間で送信する必要が生じるため，単位時間あたりの通信量はテキストのみを共有する場合と比較してはるかに大きくなる．しかも，このときに通信される操作の情報が途中で少しでも欠けると，送信側と受信側の操作が異なってしまうので，信頼性のないマルチキャストは利用できない．

大量のデータを大量の端末に配信するために，P2P 技術を利用する研究が行われており，製品も市販され

† 鹿児島大学
Kagoshima University

ている．P2P 技術をネットワークスイッチなどと組み合わせて利用することにより，大量のデータを大量の端末に短時間で信頼性を持って送ることが可能になる．

我々は，P2P 技術と排他制御を組み合わせることにより，大量の端末で描画操作などを共有するシステムを開発している．本論文では，この操作共有システムの要件，P2P 技術を利用した信頼性のあるマルチキャスト，ここで使用する排他制御アルゴリズムについて説明し，クライアント–サーバ型のシステムとの比較や，遠隔地間で実施したゲームを示すことによって，その性能を評価する．

2. 操作共有システムの要件

我々が開発している操作共有システムは，このシステムが備えているテキストエディタ，描画プログラム，簡単なプログラミング環境，Web ブラウザ，英作文支援システム[15] などのアプリケーションの操作を，多数の端末（ノード）の利用者間で実時間で共有するものである．本システムは Java で開発されており，様々な端末で構成された分散システムで利用できる．多くの X ウィンドウ端末が接続された UNIX ワークステーションや，多くの WBT 端末が接続された Windows サーバマシンのような環境でも利用することができる[17]．

本システムは以下の要件を満たすものとする．ここで，操作共有を行う利用者の集合を単に「利用者」，このとき利用者が使用しているノードの集合を「ノードのグループ」または単に「グループ」と呼ぶことにする．なお，グループ内の各ノードは，ネットワークスイッチ（スイッチ）で構成されたネットワークに接続されるものとする．

- 1 台から 40 台までのノードのグループで利用できること．これによって，1 人の利用者によるアプリケーションの利用や共同作業の準備，数人の利用者による共同作業，小中学校の 1 クラス程度の規模の利用者による共同作業などが行える．
- 端末ノードより高い性能のサーバノードなどを必要としないこと．これによって，端末ノードで本システムのプログラムを動作させるだけでシステムを利用することが可能となり，高性能のサーバを用意したり，すでに存在するサーバの管理者の手を煩わせたりする必要がなくなる．
- IP リーチャブルな異なるサブネットに接続されたノードも同一のノードのグループに参加できること．これによって，サブネットの異なる遠隔地間に分散配置されたノードの上で操作を共有できる．
- 1 つのノードでアプリケーション上のマウスの操作やテキスト入力が行われた場合，その過程についても，グループ内のすべてのノードのアプリケーションで反映されること．これによって，操作を行っていないユーザの目の前で，行われている操作がその過程も含めて細部にわたって表示される．
- 40 台のノードのグループで利用者の任意の 1 人が，自分のノードのアプリケーション上で GUI 部品を操作したとき，その操作結果がグループ内の全ノードのアプリケーションに反映されまでの遅延時間は，テキスト入力（タイピング）で 1 秒以内，マウス操作で 2 秒以内であること．なお，このときノードを接続しているネットワークは，100 Mbps のスイッチで構成されており，ノードの仕様は，CPU Intel Pentium III 450 MHz，メモリ 128 MB 程度であるとする．
- アプリケーション内の表示内容だけでなく，アプリケーションの操作そのものもグループ内で共有できること，および，1 つのグループ内の複数の利用者が同時にアプリケーションの操作を行うことができないこと．これらによって，グループ内の全ノードのアプリケーションの状態を同様に保つことができる．また，アプリケーション自身の操作方法の説明などを行うことが可能となる．

3. P2P 技術を利用した信頼性のあるマルチキャスト

すべてのノードで操作を共有するためには，1 つのノードで行われる操作情報（操作コマンドの列）をグループ内のすべてのノードに信頼性を持って放送する必要がある．また，実時間で操作を共有するためには，この放送の遅延時間をできるだけ短くする必要がある．この章では，これを実現するために本システム（P2P 型システム）が利用している，P2P 技術を使った信頼性のあるマルチキャストについて示す．ここでマルチキャストとは，同一データをグループ全体に，同時に配信する技術であるとする．

グループ内のノード間では，ノードを操作するコマンドを含んだメッセージが交換される．メッセージには，すべてのノードに宛てた「放送メッセージ」と，1 つのノードに宛てた「宛先指定メッセージ」の 2 種類がある．宛先指定メッセージは，排他制御などで利用している．ノード間は TCP で完全 b 分木になるよう結合する（**図 1**）．各ノードで動作する本システムのプログラムは基本的にはすべて同じプログラムであり，

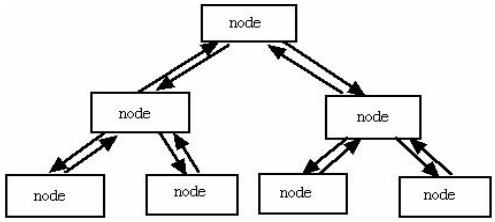

図 1 2 分木状に対等（P2P）なノード間を結合したシステム（P2P 型システム）
Fig. 1 A P2P system in which nodes are connected in binary tree shape.

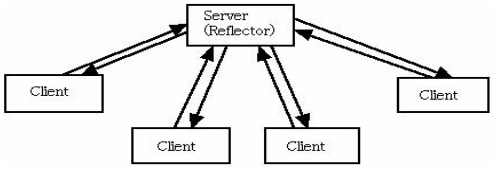

図 2 クライアント–サーバ型のシステム（CS 型システム）
Fig. 2 A client-server system.

ノード間では対等（peer-to-peer，P2P）にメッセージが交換される．

任意のノードが，1 つの TCP ソケットからメッセージを受け取ったら，そのノードは，他の TCP ソケットに受け取ったメッセージを送信した後，そのメッセージが放送メッセージか，または，自分あての宛先指定メッセージであった場合は，そのメッセージが含むコマンドを解釈実行する．宛先指定メッセージもグループ全体に放送されてしまうが，そのことによるオーバヘッドは少ないものと仮定する．

もしネットワークスイッチなどでノードが結合されていて，ノード間の通信速度とノード内の処理時間がつねに一定であり，木の根のノードから同じサイズの t 個のメッセージが連続して送信され，ノード間では 1 度に 1 つのメッセージが転送される場合，この送信が開始されてから，すべてのノードで t 個のメッセージ受信が完了する時間（最大遅延時間）T_{p2p}(sec) は，

$$T_{p2p} = C_B(tb + b(i-1)) + d(t) \qquad (1)$$

となる[6]．i は木の高さであり，木が完全 b 分木であることを仮定しているため，i はノード数の対数にほぼ比例した値となる．$d(t)$ は送信ノードにおける処理時間であるが，これ以降は無視できるものとする．C_B は 1 つのメッセージをノード間で転送するのに必要な時間であり，

$$C_B = k\frac{s}{w} + d_H \qquad (2)$$

で表すことができる．ここで s は 1 つのメッセージの大きさであり，w はネットワークのバンド幅であり，k は係数であり，d_H はネットワークスイッチの遅延時間である．式 (1) の $d(t)$ を無視してこれを変形すると

$$T_{p2p} = b((t+i-1)k\frac{s}{w} + (i-1)d_H) \qquad (3)$$

となる．メッセージが 1 個送信されるときの最大遅延時間（T_{p2p1}）は，式 (3) の t を 1 とした値であるので，

$$T_{p2p1} = b(k\frac{s}{w}i + (i-1)d_H) \qquad (4)$$

となる．スイッチの遅延時間 d_H が無視できれば，式 (1) と (3) はメッセージサイズに比例し，ノード数の対数にほぼ比例した値となる．

各ノードでは，メッセージを受け取る TCP ソケットとメッセージを送る TCP ソケットの区別はないため，グループ内の任意のノードでメッセージの送信が可能である．ここで，木の葉の部分にあるノードからメッセージが放送された場合，そのメッセージが木の根をたどって別の葉にたどり着く必要があるため，このときの最大遅延時間が最も長くなる．しかしながらこの時間は，木の根からメッセージが放送された場合の，たかだか 2 倍である．

数十台規模のノードに，ビットマップのような比較的大きなデータを放送する場合，ノード間を 2 分木状に結合し，木の根からデータを分割して送信すると，Minimal broadcast tree[3] などを使う場合とくらべて，短い時間ですべてのデータをすべてのノードに送り終わることができる[6]．このため，本システムはノード間を 2 分木状に結合している．

グループに後から新しいノードが加わったり，グループに参加していたノードが途中で離脱したりした場合にも，ノード間の結合を 2 分木になるように保つため，ノードの結合を管理するプログラムを備えている．

なお，**図 2** のような 1 台のサーバ（リフレクタと呼ばれることがある）に複数のクライアントを接続したクライアント–サーバ型のシステム（CS 型システム）の場合，送信ノードがサーバノードにメッセージを送る時間を無視すると，サーバノードを根とする高さ 2 の $(n-1)$ 分木を構成していると考えることができる．したがって CS 型システムのメッセージ列の最大遅延時間（T_{cs}）とメッセージが 1 個送信されるときの最大遅延時間（T_{cs1}）は，

$$T_{cs} = (n-1)(k\frac{s}{w}(t+1) + d_H) \qquad (5)$$

$$T_{cs1} = (n-1)(k\frac{2}{w}s + d_H) \qquad (6)$$

```
node 1
 integer p,x:0..N;
 初期化:
  p:=0;
 when この node が critical section を要求 do
  if p=0 then begin
     p:=1;
     1 を放送;
     critical section;
     p:=0;
     0 を放送; /* 解除 */
  end
 od
 when この node が x を受信 do
  if x=0 or p=0 then begin
     p:=x;
     x を放送;
  end
  /*
     それ以外は node p はすでに
     critical section に入ろうとしている
  */
 od
```

```
node i (2 ≤ i ≤ N)
 integer p,x:0..N;
 初期化:
  p:=0
 when この node が critical section を要求 do
  if p=0 then begin
     i を node 1 へ送る;
  end
 od
 when この node が x を受信 do
  p:=x;
  if p=i then begin
     critical section;
     p:=0
     0 を node 1 へ送る; /* 解除 */
  end
 od
```

図 3 本システムに組み込んでいる排他制御アルゴリズム
Fig. 3 The mutual exclusion algorithm which embedded in the computer assisted teaching system.

変数 p は critical section に入ろうとするノードの識別子を表し，変数 x はノードが受信した識別子を一時的に保管するものである．共有変数は持たない．

となる．スイッチの遅延時間が無視できれば，式 (5) と (6) は，メッセージサイズに比例し，ノード数にほぼ比例した値となる．

4. Fischer のアルゴリズムを分散システム向けに拡張した排他制御

CS 型システムにおいて，リフレクタのデータ配信メソッドに Java の synchronized method を使うことによって，2 つ以上のノードで同時に操作が行われた場合にも，グループ内のすべてのクライアントノードで同じ表示を行わせることが簡単に実現できる．これは，任意のクライアントノードで行われたキー操作やマウス操作などの操作に対応したコマンド列を，そのクライアントで直接解釈実行せずにリフレクタに送り，リフレクタが synchronized 宣言を行ったデータ配信メソッドを使ってそのコマンド列をすべてのクライアントノードに送信し，これを受け取ったクライアントノードで，そのコマンド列の解釈実行を行わせるものである．

この方法では，各クライアントノードの表示は同じでも，それぞれのクライアントでまったく別の操作を同時に行った場合，その操作が順番に並べられて表示されることになり，ユーザの意思と異なった操作が行われる場合がある．またクライアントノードで操作の制御が行われないため，操作を表すコマンド列がリフレクタに集中し，動作が重くなる場合もある．このとき，自分の操作がなかなか反映されないユーザが，何度も同じ操作を行うことによって輻輳状態になる場合もある．この問題を解決するためには，各ノードで行われる操作そのものを排他制御する必要が生じる．

図 3 に，この問題を解決するために本システムで利用している P2P 型システム向き排他制御アルゴリズムを示す．ここで，各ノードにおいて，マウスやキーボードの操作が可能な状態を critical section とする．この排他制御アルゴリズムは，Lamport の論文[8] で取り上げられている Fischer のアルゴリズム（**図 4**）

```
repeat await <p=0>;
 <p:=i>;
 <delay>
until <p=i>;
critical section;
p:=0
```

図 4 Fischer の排他制御アルゴリズム
Fig. 4 Fischer's mutual exclusion algorithm.
< 文 >は，アトミックな操作を表し，*await b* は，*while not b do skip*; を表す．i はこのノードの識別子を表し，識別子が 0 のノードはないものとする．p は共有変数である．<delay>は，競合が発生した場合に対処できるよう，ノード数に比例した十分な時間を待つことを表す．

を分散システム向きに拡張したものである．

図 3 のアルゴリズムは，ノードが node 1 から node N まであるとき，critical section に入ろうとするノードの識別子を node 1 を通じてすべてのノードに放送し，受信した番号とノードの識別子が一致するものだけに critical section を実行させる．1 つのクライアントが critical section に入っているとき，他のクライアントは，balking pattern [9] により，critical section を要求することなく，critical section に入っているクライアントから放送されるイベントの受信に専念する．この機構により，critical section を要求する信号の競合（contention）を抑制し，ユーザは，自分の意思により近い形で，操作の共有を行うことができる．critical section に入る要求の処理がグループ内で唯一のノード（node 1）を通じて行われることによって，同時に 2 つのノードが critical section に入ることを防いでいる．なお，1 つのノードが同時に 2 つのメッセージを受信することはできないものとする．

このアルゴリズムにおいて，同時に 2 つ以上の node が critical section に入ることができないことは次のように示される．まず初期値として，すべての node の p の値は 0 であり，どのノードも critical section には入っていない．node i が critical section に入るときは，node 1 の p の値が 0 から i に変化し，その後，i がすべてのノードに放送される．メッセージの受信は同時に 1 つしか行われないため，この p の値が i に変わった後は node 1 が別の node から 0 以外の識別子を受け取っても node 1 の p の値は変化せず，したがって同時に複数の node が critical section に入ることは不可能である．もし，2 つの node が同時に critical section に入っていたと仮定すると，後から critical section に入った node が critical section に入る前に，node 1 の p の値が 0 になっており，なおかつもう片方が critical section に入っていなければならない．しかしながら，どれかの node が critical section に入った後，node 1 の p の値が 0 になるのは，そのノードが critical section から離脱するときだけである．したがって，critical section に入っているノードの数はたかだか 1 つである．

本システムでは，node 1 は，2 分木状に結合されたノードの中で，根にあたるノードに割り当てている．ここで，このアルゴリズムを使用した場合，ノード間の通信速度とノード内の処理速度がつねに一定であれば，グループ内のノードが critical section に入ったとき，そのノードが critical section を要求してから自分のノード識別子を受け取って critical section に入るまでの時間は最大 $O(\log N)$ となる．これは，critical section を要求してから，自分のノード識別子を受け取るまでに最も時間がかかる葉ノードが critical section に入るとき，木の高さに比例した時間がかかるからである．

critical section の要求は複数のノードから同時に行われる場合もある．このとき，node 1 がこの要求のどれか 1 つを安全に受け取る必要がある．すべてのノードでは synchronized メソッドを使用してメッセージを中継しており，このような場合でも，要求メッセージは破壊されず，どれか 1 つの要求が最初に node 1 に届く．その後，そのノードの識別子が $O(\log(N))$ 時間以内にすべてのノードで受信され，それ以降は，critical section に入ったノードがそれを抜けるまで，他のノードが critical section を要求することはできない．

このアルゴリズムは，木の根に近いノードほど有利なので非対称であり，Dijkstra の排他制御[2] の定義からはずれている．また，critical section を要求しても永遠に critical section に入れないノードが存在する可能性があるので，starvation free [7] ではない．しかしながら，特定の利用者の指示などによる運用が可能であり，starvation free でないことは，本システムでは大きな問題ではない．

なお，本システムでは，ノードで操作が行われようとしたとき，自動的に critical section の要求を試みるように本アルゴリズムを実装している．このため，操作要求ボタンのようなインタフェースは必要ない．また，特定のノードが critical section を要求した場合は，強制的に他のノードを critical section の外に出し，その特定のノードが critical section に入るような実装が行われている．

表 1 遅延時間の比較に用いた操作の内容
Table 1 Operations which are used to compare the latencies.

操作項目	操作の内容
タイピング	テキストエディタ上で，103 文字を 18.7 秒の間にタイプする．1 文字の入力に対して 1 つのコマンドが生成され送出される．送信データ量は全体で 4.2 Kbyte であり，1 秒あたりの平均送信量は 223.3 byte/sec である．
マウス操作 1（へのへのもへじ描画）	描画プログラム上で，「へのへのもへじ」を，30.5 秒間に描画する．484 のコマンドが生成され送出される．送信データ量は全体で 17.1 Kbyte であり，1 秒あたりの平均送信量は 561.1 byte/sec である．
マウス操作 2（図形の回転拡大縮小）	描画プログラム上で，2.5 秒の間に連続して星型の図形を回転拡大縮小する．109 のコマンドが生成され送出される．送信データ量は全体で 3.8 Kbyte であり，1 秒あたりの平均送信量は 1,489.4 byte/sec である．

5. 評価実験

この章では，操作の遅延時間，同時に複数のノードで操作を行ったときの振舞い，遠隔地間での利用例を示すことによって，本システムを評価する．

5.1 操作の遅延時間

本システムでは，操作を共有するために，タイピング，マウス操作など，様々な種類のデータを，1 つのノードからグループ内の全ノードに放送する必要がある．また，実時間で操作を共有するためには，放送の遅延時間はできるだけ短い方が良い．本システムが組み込んでいる排他制御は，そのアルゴリズムの中で放送を用いている．したがって，放送の遅延時間は，critical section に入る時間にも影響する．

ビットマップデータを P2P 技術を使って放送したときの最大遅延時間については文献 6) で示されている．しかしながらこの文献では，タイピング，マウス操作などを扱っていないため，キー入力とマウス操作を行った場合の本システム（P2P 型システム）の最大遅延時間を計測した．また，1 台のサーバ（リフレクタ）にすべてのノードを接続したシステム（CS 型システム）の最大遅延時間も計測し，P2P 型システムとの比較を行った．

この実験を行うため，あらかじめ **表 1** で示す操作を 1 つのノードで行って，その操作を表すコマンド列を時刻とともに記録した．このコマンド列を，P2P 型システムと CS 型システムを使った場合のそれぞれについて，1 つのノードからグループ内の全ノードに放送した．このとき，各コマンドは記録されたときと同じ時間間隔で送信される．

操作を表すコマンド列が放送されたとき，送信ノード以外の各ノードにおいて受け取ったコマンドを時刻とともにすべて記録し，そのコマンドが送信された時刻との差を求めることによって，遅延を計測した．この平均を平均遅延時間とし，平均遅延時間が最も長くなるノードの平均遅延時間を，そのグループの最大平均遅延時間とした．ノード数は，送信ノードも含めて，5，10，19，40 とした．

P2P 型システムでは，根のノードからコマンド列が送信された場合と葉のノードからコマンド列が送信された場合の 2 種類を計測した．CS 型システムでは，送信ノードのプログラムとサーバノードのプログラムは同じ端末上で動作させた．

この計測は，鹿児島大学学術情報基盤センター教育システムを使って行った．ノードプログラムは，CPU：Intel Pentium III 450 MHz，メモリ：128 MB，NIC：100 Mbp，OS：Windows NT，JDK 1.3.1 のパソコンで動作させた．ノード数が 5，10，19 の場合は 1 台のスイッチにノードを接続した．ノード数が 40 の場合，ノードは 3 台のスイッチにまたがって接続し，これらのスイッチは 100 Mbps で 1 台のスイッチに接続した環境で計測を行った．

コマンドを受信したときの遅延時間を求めるためには，ノード間で時計を合わせる必要がある．これは，ノード間で定期的に時間の問合せをすることによって実現している．誤差も生じるが，ビデオカメラで実際の遅延時間の概略を計測し，この遅延時間と，実験によって得られた操作の遅延時間に大きな差がないことを確認した．

図 5 にそれぞれの操作におけるノード数と遅延時間の関係を示し，**図 6** にノード数が 40 のときの，単位時間あたりの送信データ量と，遅延時間の関係を示す．この図で P2P-root は P2P 型システムにおいて木の根のノードから放送を行った場合，P2P-leaf は P2P 型システムにおいて木の葉のノードから放送を行った場合，CS は CS 型システムの場合の最大平均遅延時間をそれぞれ表す．

式 (4) によれば，P2P 型システムの場合，最大遅延時間は各メッセージサイズ（コマンドのサイズ）に比例し，かつ，ノード数の対数に比例するはずである．

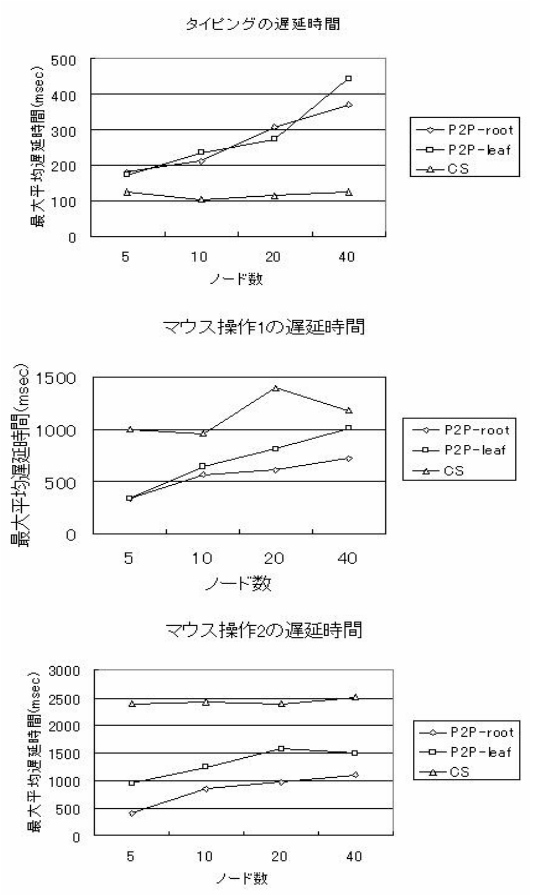

図 5 それぞれの操作におけるノード数と遅延時間の関係
Fig. 5 Relationships between the node number and the latency in each operation.

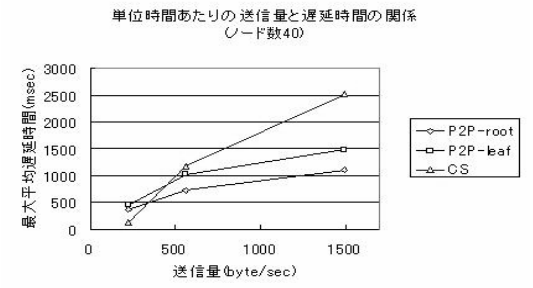

図 6 ノード数が 40 のときの単位時間あたりの送信データ量と遅延時間の関係
Fig. 6 Relationships between the size of the send data in a unit time and latencies using 40 nodes.

これに対し，式 (6) によれば，CS 型システムの場合，最大遅延時間は各メッセージサイズに比例し，かつ，ノード数に比例するはずである．サイズの大きなコマンドが連続して送られると，前のコマンドの処理を待つ時間が大きくなり，このときの最大遅延時間は，式 (3)，(5) の t を大きくしたものに近くなるはずである．このとき，サーバに負担が集中する CS 型システムよりも，木の節で負荷を分散する P2P 型システムの方が遅延時間が短くなるはずである．P2P 型システムにおいて葉からメッセージを送信した場合の最大遅延時間は，根からメッセージを送信した場合の 2 倍程度になるはずである．

実験の結果は，

- P2P 型システムの場合，遅延時間はノード数の対数にほぼ比例している．
- 単位時間あたりの送信データ量が大きくなると，P2P 型システムの方が，根から送信した場合と葉から送信した場合のどちらと比べても，CS 型システムより遅延時間が短くなる

となっており，上の予測と合致している．しかしながら，

- CS 型システムの場合，ノード数と遅延時間の関係は比例していない．

この点については，予測とは異なっている．これは，CS 型システムの場合，式 (5) における d_H などハードウェア構成や OS の状態に依存する部分の比重が大きくなり，これが無視できなくなるなどの原因が考えられる．

また

- P2P 型システムで葉から送信した場合，根から送信した場合の 2 倍も遅延時間が増えていない．

この点については，式 (3) を導出する途中で省略した，送信ノードにおける処理時間などが影響しているものと思われる．タイピング操作のときは CS 型システムの方が遅延が短かった．これは，単位時間あたりの送信データ量が少ない場合，P2P 型システムではメッセージを中継するために必要な時間が相対的に大きくなるためと考えられる．しかしながら，P2P 型システムの遅延も 40 台のノードで 0.4 秒程度であり，利用方法によっては十分利用できる．

以上のように，P2P 型システムの遅延時間は，本システムの要件である「1 台から 40 台までのノードのグループで利用できること」と「40 台のノードでタイピングを行った場合の遅延時間が 1 秒以内で，同じ台数のノードでマウス操作を行った場合の遅延時間が 2 秒以内であること」を満たしていた．これに対して，CS 型システムは「1 台から 40 台までのノードのグループで利用できること」は満たしていたが，「40 台のノードでマウス操作を行った場合の遅延時間が 2 秒以内であること」については満たしていない場合があった．なお「1 つのノードでアプリケーション上のマウスの操作やテキスト入力が行われた場合，その過程につい

ても，グループ内のすべてのノードのアプリケーションで反映されること」についてはP2P型，CS型のどちらとも，これを満たすように実装を行っている．

CS型システムにおいてノード数と遅延時間の関係は比例していない原因について，今後CPU負荷の測定を行うなどして詳しく調査する必要があるが，このCS型システムの実験ではサーバもクライアントも同じ性能のコンピュータを使用している．一般的なCS型システムで多くのクライアントにサービスを行う場合，サーバは大きな負荷に耐えられるよう，クライアントと比べて高い性能のコンピュータが必要になる場合が多いが，この実験の結果もその必要性を示していると解釈できる．したがってCS型システムは，クライアント数が多くなったとき，本システムの要件である「端末ノードより高い性能のサーバノードなどを必要としないこと」を満たすことが難しい．

5.2 同時に複数のノードで操作を行ったときの振舞い

CS型システムにおいて，サーバ（リフレクタ）のデータ配信メソッドにJavaのsynchronized methodを使うことによって，すべてのクライアントノードで同じ動作が行われる方式（synchronized method方式）を実装し，本システムに組み込んでいる排他制御方式（Fischer-balking方式）と比較した．この実験も5.1節と同じ環境で行った．

この比較は，グループ内の2台の描画プログラム上で同時にマウスの操作を行い，その振舞いを観察することによって行った．

synchronized method方式では，双方で行われた操作が交じり合い，マウスカーソルが不連続に動き，意図した操作を行うことができなかった．また，双方が操作を終了してから，マウスの動きが止まるまでの時間が長くなった．

Fischer-balking方式では，1度にどちらか一方しかマウスを操作することができず，操作を行っている利用者は，自分の意思に近い形でマウスを操作することができた．もう片方の利用者は，前の利用者の操作が終了した時点でマウスの操作を行うことが可能である．synchronized method方式と比べて，利用者のマウス操作が実時間に近い形で描画に反映され，扱いやすかった．このように，Fischer-balking方式は本システムの要件である，「グループ内の全ノードのアプリケーションの状態をつねに同様に保つため，同時に複数の利用者がアプリケーションの操作を行うことができないこと」を満たしていた．

Fischer-balking方式を，CS型システムに組み込んだものも実装して実験を行った．この場合もsynchronized method方式の問題は発生しなかった．しかしながら5.1節で述べたように，多数のノードを持つグループでマウス操作が行われた場合，それがグループ全体に反映されるまでの時間がP2P型システムと比べて長くなる．

Fischer-balking方式について，もっと多くのノードが同時に動作しようとした場合の挙動などを詳しく調査する必要があり，また，他の排他制御方式との比較も必要であるが，これらの点については今後の課題とする．

5.3 遠隔地間での利用例

本システムではノード間はTCPで接続されているため，互いに結合された複数のネットワーク（サブネット）に分散して配置された端末の上でも利用できる．このことを確認するため，本システムを使って，遠隔地にいるユーザ間で実時間オンラインゲームができるかどうかの実験を行った．複数のユーザで行う実時間ゲームは一種の実時間共同作業である．

このゲームは，約40 km離れた九州工業大学情報科学センターの戸畑と飯塚で，それぞれ10台のノードをグループに参加させ，戸畑の1ノードと飯塚の1ノードで五目並べの対戦を行い，残りのノードでその対戦を観戦するものである．戸畑と飯塚の端末はそれぞれ異なるサブネットに接続されている．この五目並べは，本システムの描画プログラムを使用した．最初に画面上に碁盤を描き，その上に遠隔地にいるユーザが黒と白の円盤を順番に描くことによって対戦が行われる．黒の円盤が黒石を表し，白の円盤が白石を表す．

この実験で，遠隔地にいるユーザ間でスムーズに対戦が行われ，本システムが遠隔地間の操作共有型システムとして利用できることを確認できた（**図7**）．このように，本システムは要件である「IPリーチャブルな異なるサブネットに接続されたノードも同一のノードのグループに参加できること」を満たしていた．

ここで，ユーザのマウス操作の過程が実時間で共有されているので，石をどこに置くか迷っている様子も相手に分かる．

この実験では，ノードプログラムを動かす端末は，Linux Thin Clientを利用した．CPUはIntel Celeron 400 MHz，OSはLinux 2.4.17（Turbo Linux 7），JavaはJava 1.4.0（Java2 RE, Standard Edition (build1.4.0-b92)）を用いた．各キャンパス内の端末は100 Mbpsでスイッチに接続されている．Javaを使って本システムを開発しているため，Windowsでも，Linuxでも再コンパイルなどをせずにシステムを

図 7 遠隔地間で行った五目並べ
Fig. 7 Gomoku narabe game which is played by the remote users.

利用することができる．

キャンパス間は 100 Mbps で接続されていたが，これには他のキャンパス間通信のトラフィックも流れている．この実験では，キャンパス間にファイヤウォールは入っていなかったが，実際の LAN は，ファイヤウォールで外部とは隔てられている場合が一般的である．本システムは，このような状況でも，LAN どうしを接続する仕組みも持っている[17]．

6. 関 連 研 究

P2P 技術を使って多くの端末間で同じ画面を共有するシステムとして，平原らの電子黒板[5),6)] がある．このシステムは，1 ノードの画面を他ノードへ一方通行で送るものであるのに対して，本システムは，利用者の間で双方向に情報交換を行うものである．

P2P 技術を使った画面転送・遠隔操作システムとして，comDesk[11)] がある．我々のシステムが，それが持つアプリケーションの遠隔操作しか行えないのに対して，comDesk は遠隔地にあるパソコン画面を手元のパソコンに表示させ，その上で別のパソコンを操作することによって任意のアプリケーションの遠隔操作を行うことができる．しかしながら，comDesk は異なるネットワーク（サブネット）をまたいだ環境で利用することはできないのに対して，我々のシステムは，そのような環境でも利用可能である．また，comDesk について述べた論文 11) では，3 台のノードを使った例が示されているが，我々のシステムは 40 台のノードで利用した実績がある．

P2P 技術を利用したグループ内のマルチキャストに関する研究として，ESM[18)]，RelayCast[10)]，Emma[12)] などの研究も行われている．これらの研究では排他制御やアプリケーション操作の共有については述べられておらず，本システムの要件である，「1 つのグループ内の複数の利用者が同時にアプリケーションの操作を行うことができないこと」が満たされるか否かは不明である．

Raymond は木を使って $O(\log(N))$ のメッセージで critical section に入る排他制御アルゴリズムを示している[13)]．Raymond のアルゴリズムは，starvation free であるが，各ノードで 3 つの変数と 1 つの待ち行列が必要になる．これに対して我々のシステムでは用途を starvation free がそれほど重要ではないものに限ることによって，アルゴリズムを単純にしている．

辰本らは，LOTOS コンパイラのマルチランデブーの実装にあたり，プロセスの生成や消滅の情報を一カ所から全ノードに放送する手法を提案し，応用例として排他制御問題についても述べている[16)]．本システムで使用している排他制御もこれと同様に，1 カ所から全ノードに放送するが，P2P 技術を用いた信頼性のある高速なマルチキャストを使用することによって，排他制御をより信頼性の高いものにしている．

P2P 技術を使った排他制御として，Desai らのプロ

トコルがある[1]．このプロトコルは，排他制御を行うためにノード間を木状に結合するが，この結合を動的に変化させるため，比較的密結合の分散システム向きであり，本システムの要件である「IP リーチャブルな異なるサブネットに接続されたノードも同一のノードのグループに参加できること」を満たそうとすると，排他制御に時間がかかる恐れがある．これに対して我々のシステムは，木の形を変化させずに排他制御を行っており，ノード間接続に時間がかかる環境に向いている．遠隔地の異なるサブネット上に分散配置されたノード間で排他制御を行い，共同作業を行った実績もある．

遠隔地にいるユーザ同士で実時間に描画を行うコミュニケーションツールとして wb が広く使われている[4]．しかしながら wb では複数の利用者で 1 つの操作が共有されるのではなく，複数の利用者のそれぞれの描画が，1 つの画面に表示されるものであり，同時に複数利用者が書き込むことが可能である．このことは，共同作業を並列にできる良い面もあるが，我々のシステムの要件である「アプリケーション内の表示内容だけでなく，アプリケーションの操作そのものもグループ内で共有できること，および，1 つのグループ内の複数の利用者が同時にアプリケーションの操作を行うことができないこと」を満たしていない．我々のシステムはこの要件を満たしているが，共同作業を並列に行えない欠点を持っている．wb は SRM と名づけられた信頼性のあるマルチキャストフレームワーク上で動作している．しかしながら SRM は放送されるデータが，放送された順番にすべてのノードで受信されることは保障していないため，ボタンのクリック，マウスの移動，スクロールバーの動作などが，すべてのノードで同じ順番で行われる必要がある我々のシステムに使うことはできない．

P2P 技術を利用した実時間グループウェアとして SOBA プロジェクト[14] が開発しているツールがある．これも描画プログラムなど様々なアプリケーションを備えているが，SOBA の描画プログラムは，利用者が 1 つの画素を書き終えた時点でそれをグループ全体に表示するようになっており，我々のシステムの要件である「1 つのノードでアプリケーション上のマウスの操作やテキスト入力が行われた場合，その過程についても，グループ内のすべてのノードのアプリケーションで反映されること」を満たしていない．また，SOBA の描画プログラムは wb と同様に，複数の利用者のそれぞれの描画が，1 つの画面に表示されるものである．しかしながら SOBA は，グループですでに作業が進んでいるとき，途中でそのグループ参加するノードに対して，その時点のグループのアプリケーションの状態を伝える機能を備えている．我々のシステムは，まだこの機能を備えておらず，途中参加が頻繁に発生する状況においては，このような機能を追加する必要がある．

7. おわりに

多数の端末間で実時間で操作を共有するための，操作共有システムについて述べた．このシステムは，1 つのノードで行われる操作を短い遅延で信頼性を持って全ノードに伝えるために，P2P 技術を利用している．双方向で操作を共有するため，排他制御を組み込んでいる．

現在本システムは，グループですでにアプリケーションの操作が進行していたとき，新規ノードが途中でこのグループに参加した場合に，新規ノードに他ノードの状態が反映されない．ノード間は 2 分木状に結合されているため，葉以外の場所にあるノードで障害が発生した場合，操作情報が伝わらないノードが生じる．また，実時間で遠隔地間で共同作業などを行う場合は音声の放送機能が重要になるが，本システムはまだ音声データを放送することはできない．今後，これらの点を改良していく予定である．

謝辞 本システムの実現と評価実験にあたりお世話になりました，九州工業大学情報科学センターと鹿児島大学学術情報基盤センターの皆さまに感謝します．

参考文献

1) Desai, N. and Mueller, F.: A Log(n) Multi-Mode Locking Protocol for Distributed Systems, *Proc. International Parallel and Distributed Processing Symposium* (*IPDPS'03*), IEEE (2003).
2) Dijkstra, E.W.: Solution of a Problem in Concurrent Programming Control, *Comm. ACM*, Vol.8, No.9, p.569 (1965).
3) Farley, A.: Minimum broadcast network, *Network*, Vol.9, pp.313–332 (1979).
4) Floyd, S., Jacobson, V., Liu, C., McCanne, S. and L., Z.: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, *IEEE/ACM Trans. Networking*, Vol.5, No.6, pp.784–803 (1997).
5) Hirahara, T., Yamanoue, T., Anzai, H. and Arita, I.: Sending an Image to a large number of nodes in short time using TCP, *IEEE International Conference on Multimedia and Expo*, pp.987–990, IEEE (2000).

6) 平原貴行，山之上卓，安在弘幸，有田五次郎：TCP を利用した分散ネットワーク環境のための電子黒板システム，情報処理学会論文誌，Vol.43, No.1, pp.176–184 (2002).
7) Knuth, D.E.: Addtional Comments on a Problem in Concurrent Programming Control, *Comm. ACM*, Vol.9, No.5, pp.321–322 (1966).
8) Lamport, L.: A Fast Mutual Exclusion algorithm, *ACM Trans. Computer Systems*, Vol.5, No.1, pp.1–11 (1987).
9) Lea, D.: *Concurrent programming in Java: Design principles and patterns*, Addison Wesley, Reading, Massachusetts (1997).
10) Mimura, N., Nakauchi, K., Morikawa, H. and Aoyama, T.: RelayCast: A Middleware for Application-level Multicast Services, *Proc. 3rd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems* (*GP2PC 2003*), Tokyo, Japan, pp.434–441 (2003).
11) 三浦元喜，志築文太郎，田中二郎：P2P 技術を適用した画面転送・遠隔操作システムの開発，情報処理学会論文誌，Vol.45, No.1, pp.289–299 (2004).
12) 中村嘉隆，山口弘純，廣森聡仁，安本慶一，東野輝夫，谷口健一：映像による複数人のコミュニケーションシステム向けのアプリケーションレベルマルチキャスト Emma の性能評価，マルチメディア，分散，協調とモバイル（DICOMO 2002）シンポジウム論文集，pp.253–256, 情報処理学会 (2002).
13) Raymond, K.: A Tree-Based Algorithm for Distributed Mutual Exclusion, *ACM Trans. Computer Systems*, Vol.7, No.1, pp.61–77 (1989).
14) SOBA 事務局：SOBA Project web site. http://www.soba-project.org/jp/index.html
15) Yamanoue, T., Minami, T., I.R.W.S.: Learning Usage of English KWICly with WebLEAP/DSR, *Proc. 2nd International Conference on Information Technology and Applications* (*ICITA-2004*), *14-6*, Harbin, China (2004).
16) 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，Vol.40, No.1, pp.333–342 (1999).
17) 山之上卓：様々な分散環境で動作する操作共有型教育支援システム，マルチメディア，分散，協調とモバイル（DICOMO 2003）シンポジウム論文集，pp.245–248, 情報処理学会 (2003).
18) Chu, Y.-h., Rao, S.G.S.S. and Zhang, H.: Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture, *Proc. ACM SIGCOMM*, San Diego, CA, ACM (2001).

(平成 16 年 5 月 13 日受付)
(平成 16 年 11 月 1 日採録)

山之上　卓（正会員）

昭和 34 年生．昭和 59 年九州工業大学大学院工学研究科情報工学専攻修士課程修了．昭和 62 年九州大学大学院総合理工学研究科情報システム学専攻博士後期課程単位取得退学．平成 5 年より九州工業大学情報科学センター助教授．平成 15 年より鹿児島大学学術情報基盤センター教授．P2P，教育支援システム，分散システムの評価システム等の研究に従事，博士（工学）．電子情報処理学会，ソフトウェア科学会，IEEE-CS，ACM 各会員．