2003－DSM－29 (3)
2003／4／25

P2P 技術を応用した分散システムの排他制御機構の試作

山之上卓[†1]

P2P 技術を応用した分散システムの排他制御機構の試作を行い、この機構を教育支援システムで利用したことについて示す。Critical section に入っているノード番号を全ノードが知るために、信頼性のあるマルチキャストを利用している。このマルチキャストは、ノードの結合が２分木状になるように TCP で接続し、メッセージをノード間でバケツリレーする P2P 通信を用いることによって実現されている。ノード数を n とすると、Critical section に入る時間は最大 $O(\log n)$ である。この機構は starvation free ではないが、ここで述べる教育支援システムの利用では大きな問題にはならない。80 台のノードで本排他制御機構の性能を実測し、本機構がスケーラブルであることを確認した。

Experimental implementation of a Mutual Exclusion Mechanism for Distributed Systems using a P2P technology

Takashi Yamanoue[†1]

Experimental implementation of a mutual exclusion mechanism for distributed systems, which uses a P2P technology, and its application to a computer assisted teaching system, are shown. A reliable multicast is used in order to broadcast the ID number of the node which is entering the critical section. This multicast is realized by a P2P communication in which the nodes relay messages. The connection of the nodes has the shape of a binary tree. The time complexity of entering the critical section is $O(\log n)$ where n is the number of the nodes. This mechanism is not starvation free. However starvation free is not important to our computer assisted teaching system. We measured the performance of this mutual exclusion mechanism by using 80 nodes, and we have confirmed that this mechanism is scalable.

1. はじめに

E-commerce やネットワークを使った遠隔機器操作を実現したり、多くの参加者がリアルタイムで共同作業を行い、一つの作品を仕上げるような CSCW システムや教育支援システムを実現したりするために、分散システム上の排他制御機構が必要となる場合がある。

分散システムの排他制御に関して数多くのアルゴリズムが提案されている。この中で、Raymond[1] や Agrawal and El Abbadi[2] は、ノードを木状に結合することにより、ノード数が n の場合に $O(\log n)$のメッセージ数で critical section に入るアルゴリズムを示している。

近年 P2P 技術を使った分散システム上のアプリケーションが数多く出現している。P2P 技術を使ってノード間を木状に接続する

†1 九州工業大学情報科学センター, Information Science Center, Kyushu Institute of Technology

ことにより、Raymond や Agrawal and El Abbadi らの排他制御アルゴリズムを容易に実現することが可能となる。

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箇所から全ノードにブロードキャストする手法を提案し、応用例として排他制御問題についても述べている[3]。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、この手法をより信頼性の高いものにすることができる。

我々は P2P 技術を利用した教育支援システム[4]を開発しているが、この教育支援システムも、ノード間を P2P 技術で木状に接続しており、木構造を用いた排他制御アルゴリズムを簡単に導入することができる。

Lamport によると、最も簡単な排他制御アルゴリズムの一つは、以下の Fischer のものである[5]。

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

ここで、<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(プロセス)の ID を表し、ID が 0 のノードはないものとする。このアルゴリズムは、共有メモリー上にある変数 *p* の使用を前提としており、このままでは分散システムには利用できない。また、競合(contention)が発生した場合に、少なくとも $O(n)$の時間が必要となる。

本論文で述べる排他制御機構は、Fischer のアルゴリズムを、信頼性のあるマルチキャストを使って分散システムで利用できるように改造したものであり、競合も抑制する。信頼性のあるマルチキャストを実現するため、P2P 技術を応用している。競合があっても、なくても、$O(\log n)$の時間で critical section に入ることができる。

本論文では、２章で本排他制御アルゴリズムとその理論的な性質について述べ、３章で教育支援システムへの応用と性能の実測について述べる。

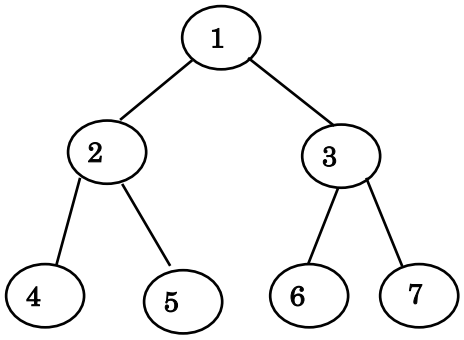

図 1　本排他制御機構におけるノードの結合

2. P2P 技術を応用した排他制御アルゴリズムとその性質

2.1 ノード間の結合(データ構造)

ノードは n 個あり、各ノード は 1 から n までの番号がついているものとする。

node 1 を頂点として、図 1のように完全２分木になるよう、TCP でノード間を接続する。ここで、木の高さを h とすると、ノード数 n は、$2^{h-1} \leqq n \leqq 2^h - 1$ となる。接続されたノード間は全２重通信が可能になるが、互いに接続されたノード間でしか直接メッセージ交換はできない。ノード１からメッセージが放送されるとき、直接接続されたノードに、１つずつ同じメッセージを送る。メッセージを受け取ったノードは、それが送られてきたノード以外の、直接接続されているノードに、受け取ったメッセージを一つずつ送る。このとき、それぞれのノードにおいてメッセージの送受信は同時並行で行われると仮定する(スイッチを使ったネットワークでこれを実現することができる)。このようにメッセージが送られることによって、すべてのノードに、木の高さに比例した時間($O(\log n)$)以内でメッセージが届く。なお、葉の方からメッセージが放送される場合も、$O(\log n)$の時間でそれがすべてのノードに届く。任意のノードか

らノード１へメッセージを送るときも、最大 $O(\log n)$の時間で届く。Minimal broadcast tree を使うと、根にある node から放送が行われるときの時間が最小になるが、本論文の実装では、完全２分木を使っている。

2.2 アルゴリズム

Dijkstra の排他制御問題の定義[6]では、すべてのノードは対称的であり、優先順位があってはならないとしているが、ここでは現実的な解を求めるため、この条件は満たさなくても良いこととする。根のノードを node 1 とし、このノードは、他のノードとは異なる動作を行わせることにする。また、あるノードが critical section を要求しても、場合によっては、これが拒否されることがあるものとする。

このアルゴリズムは Fisher のアルゴリズムにおける変数 p をすべてのノードに持たせ、p への値の代入の代わりに、その値を全ノードに送信して、各ノードがその値を受け取って p に代入することを行う。このとき、最初に node 1 に値が送信され、node 1 から全ノードにその値が放送される。以下に、アルゴリズムを示す。

```
node 1
  integer p:0..n
  initialize
    p:=0;
  when this request critical section, do
    if p=0 then begin
        p:=1;
        broadcast 1;
        critical section;
        p:=0;
        broadcast 0;    /* release */
    end
  od
  when this received x,   do
      if x=0 or p=0 then begin
          p:=x;
          broadcast x;
      end
    /* otherwise, the node p is already
       being to enter the critical section */
  od

node i (2 ≤ i ≤ n )
  integer p:0..n
  initialize
    p:=0
  when this request critical section, do
    if p=0 then begin
        send i to node 1;
    end
  od
  whien this received x, do
    p:=x;
    if p=i then begin
        critical section;
        send 0 to node 1;   /* release */
    end
  od
```

任意のノードが node 1 に番号を送信したとき、$O(\log n)$の時間内に、node 1 はその番号を受信する。node 1 が番号を放送したとき、すべてのノードは $O(\log n)$の時間内にその番号を受信する。

2.3 critical section に入っているノードの数は高々１個であることの説明

初期状態: すべてのノードで p の初期値は 0 であり、最初は、どのノードも critical section に入っていないものとする。このときのノードの集合を S_0 とする(図 2)。

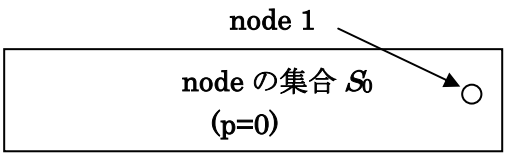

図 2　初期状態

状態 1: 初期状態の後、任意のノードは p の

値が 0 であるため、 critical section に入ることを要求できる。node 1 が 最初に node x からのメッセージ(x)を受け取った場合、node 1 から x が全ノードに送信され、それを受け取ったノードの p の値が x になる。node x が x を受け取った時点から、node x は critical section に入る。node x が critical section から離脱するまで(node x が node 1 に 0 を送信して、それを node 1 が受け取るまで)、node 1 の p の値が 0 にならないため(要求を送っても node 1 からその番号が放送されることはないため)、他のノードは critical section に入ることができない。したがって、この時点で critical section には入っているのは、node x だけである。このとき、p=x であるノードの集合を S_1 とすると、**図 3**のようになる。node x は、最初は S_0 の要素であるが、critical section に入ると、S_1 の要素になる。このとき、node x は critical section に入る。時間の経過と共に、S_0 の要素は、S_1 に吸収されていく。

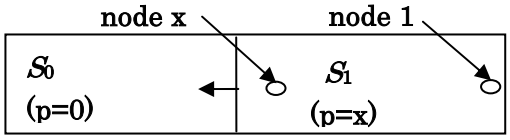

図 3 状態 1

状態 2: 状態 1 の後、node x が critical section を離脱するとき、node x の p が 0 になってから、node 1 より全ノードに 0 が送信される。このとき、ノードの集合を図のように、p の値が x であるノードの集合 S_1 と、node x が critical section に入る前から p の値が 0 であったノードの集合 S_0 と、p の値がいったん x になった後、0 になったノードの集合 S_2 の、互いに疎である 3 つの集合に分ける(S_0 は空集合である場合もある)。時間の経過とともに、S_0 の要素は S_1 に吸収されていき、S_1 の要素は S_2 に吸収されていく。この時点で critical section に入っているノードはない。この後、critical section に入ることのできるノードは、S_0 の要素と S_2 の要素のみである**(図 4)**。

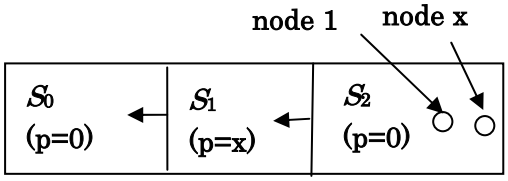

図 4 状態 2

状態 3: 状態 2 の後、node 1 が 最初に $S_0 \cup S_2$ に属する node y から critical section を要求するメッセージ y を受け取った場合、node 1 から y が全ノードに送信され、それを受け取った node の p の値が y となる。node y が y を受け取った時点から、node y は critical section に入る。node y が critical section から離脱するまで、node 1 の p の値が 0 にならないため、他の node は critical section には入ることができない。状態 2 で critical section に入っている node は 0 個であったので、node y が critical section に入る唯一のノードとなる。このとき、p=y であるノードの集合を S_3 とすると、図のようになる。この後は、y を x と読み替え、$S_0 \cup S_1 \cup S_2$ を S_0 と読み替え、S_3 を S_1 と読み替えることによって、状態 1 と同じになる**(図 5)**。

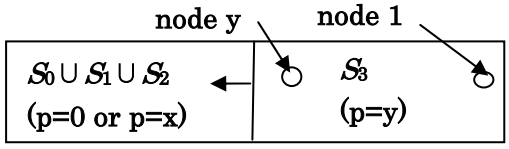

図 5 状態 3

以上、初期状態から状態 3 までの、すべての状態で、critical section に入っているノードは高々1 である。

2.4 critical section に入る時間

葉の node から node 1 に critical section

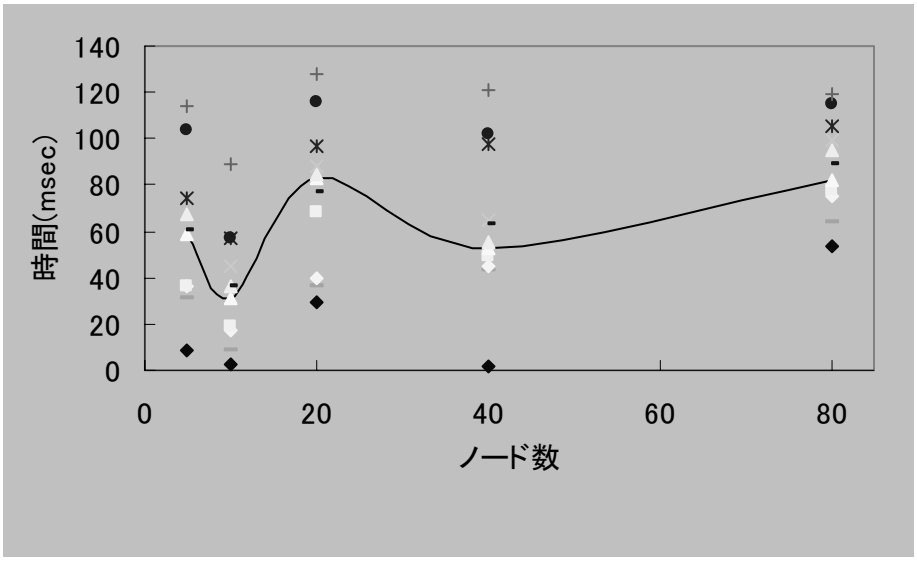

図 6 競合の無い状態において、葉のノードが Critical Section に入る時間の実測値(実線は平均値)

に入る要求(葉の node の番号)が送られ、node 1 からその番号が放送されて、このノードに届くまで、それぞれ最大 $O(\log n)$ の時間がかかり、合計も $O(\log n)$である。葉のノードが critical section に入る時間が最大となるので、critical section に入るために必要な時間は、最大 $O(\log n)$となる。Critical section から離脱する時間も最大 $O(\log n)$となる。

2.5 競合(contention)の抑制

ある node が node 1 にその番号を送って critical section に入ろうとしているとき、他のノードで critical section の要求メッセージを送ることができるものの数(p が 0 のノードの数)は、node 1 がその番号を受信してから $O(\log n)$時間後に 0 となる。これはデザインパターンにおける一種の Balking パターンとなっており、critical section には入るための競合(contention)を抑制している。

また、複数の node が同時に critical section を要求した場合も、木の節で交通整理が行われて、少ない競合でどれか 1 つが最初に node 1 に届き、その番号のみが全ノードに放送される。従って、競合が発生するような場合でも、$O(\log n)$の時間で critical section には入ることができる。このとき、木の根に近い node が有利となる欠点があるが、共同作業ツールで操作の排他制御を行う場合など、ユーザが他を意識して操作を行うようなアプリケーションの場合は、それほど大きな影響はない。またこのような用途で使う排他制御は starvation free である必要はない。

3. 教育支援システムへの応用と性能の実測

我々は、様々な分散システムで利用できる教育支援システムを開発している。この教育支援システムは、お絵かきソフト、簡単なプログラミング環境、簡易 Web ブラウザなどのアプリケーションを備えており、教師側端末でこれらのアプリケーションの操作が行われると、多くの学生端末でも同じ操作がリアルタイムで再現される。この教育支援システムに本排他制御機構を試験的に実装し、複数の端末間で、一つの操作を共有する機能を付け加えた。このことによって、複数の学生がリアルタイムで共同作業を行うことができる。

この教育支援システムを九州工業大学情報科学センター教育システム[7]で動かして、

操作を共有する状態にし、critical section に入る時間を計測した。ネットワークは、100Mbps のポートを 48 口と 1Gbps のポートを 1 口持つスイッチを、ギガスイッチで結合したものを使用した。ノードは、CPU:Intel Celeron 400MHz、OS: Linux 2.4.17(Turbo Linux 7)と Java 1.4.0(Java2 RE, Standard Edition(build1.4.0-b92))の環境の上で動作させた。node 数は 5,10,20,40,80 とし、node の番号が最大の node において critical section に入る時間を 10 回ずつ計測した。Breadth first の順番にノードの番号を付けているので、このノードが葉のノードとなる。この計測は競合が無い状態で行った。ネットワークにスイッチを使っているので、その性能とトポロジーが対称で、コンピュータの性能や状態が同じであれば、node の番号が最大のものが critical section に入る時間が最大になるはずである。この計測結果を**図 6**のグラフで示す。

このグラフを見ると、データのばらつきがかなり大きく、また、平均値は単調増加になっていない。これは、実際の測定環境ではネットワークやコンピュータの状態に偏りが存在することなどが原因の一つと考えられる。しかしながらノード数を増加しても critical section に入る時間はあまり増加しておらず、本排他制御機構の効果が表れていると考えることができる。

4. おわりに

P2P 技術を応用した分散システムのための排他制御機構と、その教育支援システムへの応用、および 80 台のノードにおける、この排他制御機構の性能測定について述べた。性能測定の結果、80 台のノードまでは、この排他制御機構がスケーラブルな性能を示すことを確認した。

あるノードが critical section に入るために node 1 にメッセージを送るとき、不要な通信を排除するために、各ノードがルーティングを行う必要があるが、現時点では、宛先番号(この場合 1)を付加したメッセージを放送している。宛先ノードは、自分と同じ番号の宛先番号を持ったメッセージが届いたとき、それを受信する。

今後は、競合のある状態での性能測定などを行う予定である。本排他制御機構を組み込んだ教育支援システムは、http://www.tobata.isc.kyutech.ac.jp/~yamanoue/researches/dsr/で公開している。

謝辞

本論文の性能実測を手伝ってくれた学生諸君に感謝します。

参考文献

[1] Raymond, K., “A Tree based algorithm for Distributed Mutual Exclusion”, ACM Transactions on Computer Systems, Vol. 7, No. 1, pp.61-77, 1989.

[2] Agrawal,D., El Abbadi,A., “An Efficient Solution to the Distributed Mutual Exclusion Problem”, ACM Transactions on Computer Systems, Vol. 9, No. 1, pp.1-20, 1991.

[3] 辰本比呂記ほか,“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.

[4] 山之上 卓ほか,"分散システムのためのプラットフォーム独立な教育支援システム", 情報処理学会研究報告 2001-DSM-24, pp.19-24,2001.

[5] Lamport, L.,” A Fast Mutual Exclusion Algorithm”, ACM Transactions on Computer Systems, Vol. 5, No. 1, pp.1-11, 1987.

[6] Dijkstra, E.W.,” Solution of a problem in concurrent programming control”. CACM,Vol.8, No.9, p.569, 1965.

[7] 中山仁ほか,”Linux thin client を端末とする集合教育用計算機環境の構築”, 情報処理学会研究報告 2000-DSM-18, pp.31-36 , 2000.