

アドホックネットワークにおける改良型 動的複製制御方法による分散オブジェクト管理

浅井 健一^{†1}

川越 恭二^{†2}

動的分散環境では複製位置が固定されないため、通常、複雑な更新制御を必要とする。この問題を避けるために、更新処理を行わずに新しいオブジェクトとして扱う方式が提案されている。しかし、この方式では、オブジェクトが最新のものを判別するのは困難であり、古いオブジェクトを参照する可能性がある。また、以前に分散配置されていた場所を利用することができず、初期状態から複製配置を作る必要がある。これらの問題を解決するため、本研究ではオブジェクトの更新を行う新しい改良型動的複製制御方式を提案する。本方式では、まず仮想距離を算出し、その距離を用いて既に更新されている複製から更新を行う、動的複製オブジェクト更新方式を提案する。この方式により、以前の複製配置を用いたまま、オブジェクトの更新を行うことが可能である。

Distributed Object Management with Improved Dynamic Replica Control in Ad Hoc Network

KENICHI ASAI^{†1}

KYOJI KAWAGOE^{†2}

It is difficult to renew a replica object in dynamic content distribution system, as the replica location is varied. Such system, which registers a new object without updating, has been presented so far. However, in the system, it is difficult to distinguish whether the accessed object is the new one or not. Moreover, it cannot use the current replica map in the system. In order to improve the replica object allocation, we propose "Improved Dynamic Replica Control" for efficient update of replica object. A system with the proposed method can use a current replica map, and can update a replica object with balanced load distribution.

1. はじめに

近年の有線ネットワークの広帯域化や、移動体における性能の向上や無線ネットワークの広帯域化により、移動体においても情報サービスを実現することが可能となっている。例えば、車においては、カーナビゲーションシステムが普及し、ディスク媒体による固定地図データではなく、ネットワークの配信による地図データの配信が徐々に開始されている。このような情報サービスでは、地図データだけではなく、利用者や企業などからの情報も取り込み、建物の外観データや写真、さらにはCMといった動画なども配信の対象として考えられる。このため、これらの情報をオブジェクトとして統一して取り扱う必要がある。

現在、ネットワーク環境は、ネットワーク位置が固定された計算機、半固定された一般利用者の計算機、さらに移動し、場所が変化する移動体端末といった端末によって構成される。このようなネットワーク環境における情報共有サービスでは、配信元と接続できない状況でも必要なオブジェクトを参照できる必要がある。同時に狭帯域ネットワークの場所に参照オブジェクトが集中しないように、ネットワーク負荷を分散しなければならない。そのため、通常、ネットワーク上に、オブジェクトの複製を作成する方法によってこの問題解決が行われている。しかし、動的な分散環境においては、負荷分散、ルーティング、レプリケーション、更新制御といった分散制御の機能はネットワーク位置を固定できないことから、直接利用しても効果的に働かず、逆にネットワーク負荷を増大させてしまうという問題が発生する。この問題を解決するためには、動的な端末のネットワーク配置とルーテ

^{†1} 立命館大学大学院 理工学研究科
Graduate School of Science and Engineering, Ritsumeikan Univ.

^{†2} 立命館大学 理工学部
School of Science and Engineering, Ritsumeikan Univ.

イング, 更新方法を検討する必要があると考える. そこで本研究では, 分散配置のために提案されている Plaxton アルゴリズム[1,2]とツリー構造を組み合わせ, オブジェクト配置リストによるネットワーク仮想距離を用い更新制御を行う方法を提案する.

以降, 2章では, 分散配置方法を静的ネットワーク配置と動的ネットワーク配置で分けて説明し, 問題点を述べ, 3章で本提案を実現するための, オブジェクト配置リスト, Plaxton アルゴリズムとツリー構造を説明し, ネットワークへの参加手順と離脱手順を述べる. 4章では, 複製オブジェクトの作成と本研究の提案方式である複製オブジェクトの更新制御を述べる. 5章にシミュレーションによる実験と結果. 6章でまとめと今後の課題について述べる.

2. 分散配置法

本章では, 既存の分散配置方法を, 静的ネットワークと動的ネットワークに分けて説明し, 両方式の利点と問題点を述べる. また, 両方式の利点を活かしつつ, 問題点を解決する手段を提案する.

2.1 静的ネットワーク配置法

静的ネットワーク配置法(図 1)は, 決まったネットワーク配置において, 効率の良い分散配置をする方法である. この方式では前もって各オブジェクトへのアクセス量を予測, 計測し, アクセス量が多く, かつオブジェクトまでのネットワーク距離が大きい場合には, 近くにオブジェクトの複製を配置し, ネットワーク負荷を減少させる方法である. ネットワークを予測するため, 確実な効果を得ることが可能である. しかし, 通常オリジナルのオブジェクトを中心として中央集中型の分散配置をとるため, オブジェクトの更新が行われた場合に, 多くの複製された端末へ, 同時にアップデートが行われることがあり, ネットワーク負荷が一時的に集中する問題がある.

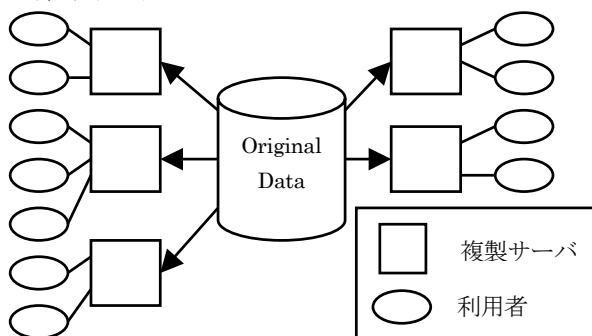


図 1. 静的ネットワーク配置

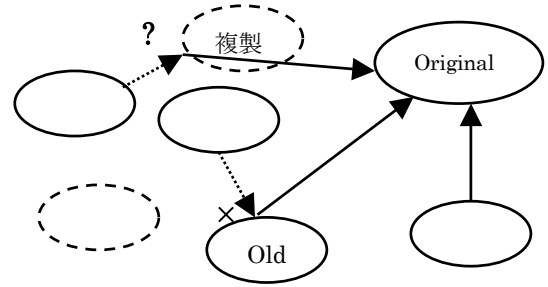


図 2. 動的ネットワーク配置

2.2 動的ネットワーク配置法

動的ネットワーク配置法(図 2)は, ネットワーク配置が固定していない. そのため, オブジェクトの複製位置を前もって確定する必要がない. しかし, 逆に, 端末が常にネットワーク上に存在していない場合があり, 複製されたオブジェクトの位置が不確定となり, 以前に複製された物が, その後もその場所にあるとは限らない. このため, オブジェクトに対するルーティングが正確に行われないことがある. また, 複製位置が固定できないことから更新が困難であるため, 利用者からそのオブジェクトが更新されたものであるかが確認できないため, 古いオブジェクトに対して, アクセスをしてしまい再度新しいオブジェクトを取り出す必要がある. このようなことからネットワーク負荷を増加させる問題点があげられる.

2.3 問題点の解決方法

先に挙げた分散配置の問題点を解決するため, 参加端末を仮想的なツリー構造とし, オブジェクトの場所を示すオブジェクト配置リストを構築する. このオブジェクト配置リストを用い, 更新済みオブジェクトを区別し, 端末間の仮想距離を算出, その距離を利用して分散的なオブジェクトの更新を実現する.

3. ルーティングと配置場所

本章では, 端末の扱いと本提案で利用する端末探索アルゴリズムである, Plaxton アルゴリズム, 仮想距離を算出するツリー構造を説明し, さらにオブジェクト配置リストの概要を説明し, さらにネットワークの参加と離脱の方法を述べる.

3.1 端末名と IP

本研究で利用する Plaxton アルゴリズムでは, 端末に IP とは別に名前付けをし, その名前を元にルーティングを行う. この名前付けは, Tapestry[3,4]の Dynamic Algorithms で定義される, 端末名を利用する. これは, 端末がネットワークに参加した

場合、その端末のゲートウェイから自分のネットワーク内に参加している近接端末のマップを作成し、それを利用して、自己端末の名前を決定する。その中で見つけることが不可能であった場合は、さらに次の上位ルータへと上がり、先ほどと同様にマップを作成し、検索を行っていく。このように端末名を決定することで近接の端末間で名前付けを行うことが可能となる。また、距離的に近い物は、名前を似せることが可能となる。本研究では、3.3で述べるツリー構造にこの名前を適応させるため、Tapestry の名前付けに、階層構造を加える。

3.2 Plaxton アルゴリズム

Plaxton アルゴリズムでは、端末名の末端文字から順に一致する端末を検索し、ルーティングを行う。本研究では、ルーティングに Plaxton アルゴリズムを利用する。アルゴリズムを図 3 に示す。

3.3 ツリー構造

3.2 で述べた Plaxton アルゴリズムでは、アクセス前のネットワーク距離の予想が困難である。そのため、アクセスが分散されない場合がある。そこで 3.1 で付けられた端末名をツリー構造として、距離の算出を行う。距離算出のアルゴリズムを図 4 に示す。この距離は実際に利用されるルーティングではないが、3.2 での Paxton アルゴリズムとは逆に最大距離と仮定することで、すべての複製(オリジナルを含む)オブジェクトに対し、最低条件における条件で比較が可能となることで、複製に対する、アクセス分散可能である。各端末は、ツリーを構築するため、自分の親端末と子端末の IP、そしてルートの IP を持つ。

```
function search( node_name , i )
    /* 端末名 node_name の文字の i 番目以降が一致する */
    /* 近接端末を検索し、その端末名を返す */
function move( node_name )
    /* node_name の端末へ移動 */
procedure Plaxton_alg
    c_node = 要求端末; g_node = 目的端末;
    L = c_node と g_node で少ない方の文字数
    dif = length(g_node) - length(c_node);
    for( i=0; i <= L-1; i++ ) {
        move( Search(g_node , L-1) );
    }
    if dif > 0
        for( j=0; j < dif-1; j++ ) {
            move(search( g_node , dif-j ));
        }
    fi
end
```

図 3 Plaxton アルゴリズム

```
procedure near_search
    c_node = 要求端末;
    while リストの最後まで
    {
        o_node = node_name /* 端末名 */
        x = c_node と o_node での長い端末名を持つ文字数
        y = c_node と o_node での短い端末名を持つ文字数
        dif = x - y;
        compare(c_node,o_node)
        /* c_node と o_node の両端末名の末端から文字列を比較 */
        j = 最初に異なる文字列の場所
        if distance > (y - j + 1) * 2 + dif
            distance = (y - j + 1) * 2 + dif
        fi
    }
end
```

図 4 近接端末検索

3.4 オブジェクト配置リスト

オブジェクト配置リストは、オブジェクト Object ID とそのオブジェクト Object ID が存在する端末の名前（端末名）の組を格納したリスト(表 1)である。このオブジェクト配置リストを OPL と呼ぶこととする。これらの OPL は、3.3 のツリー構造のルート端末に集約され管理される。ルート端末は、利用者の端末とは異なり、大きなネットワーク帯域幅と、処理能力を持つデータセンタ的な計算機を想定する。データベース上には、Object ID と端末名の対応を持ち、オリジナルのオブジェクトが更新された日時を格納する。この集約されたデータベースを R-OPL と呼ぶこととする。利用者は、検索するとき、仮想構築されたツリーを使いツリーのリーフからルート方向へアクセスしていき、この OPL を探索する。OPL 内に Object ID を見つけた場合は、その OPL を用いて、仮想距離を算出して、近接端末を検索する。探索できなかった場合にはルートの R-OPL から Object ID 単位で取り出し OPL を作成し、同様に近接端末を検索する。ルートから端末を検索した場合には、リーフへの途中の通過端末にオブジェクト配置リストを配置していき、次回検索時の高速化を行う。以上のアルゴリズムを図 5 に示す。

表 1 オブジェクト配置リスト(OPL)

Object ID	端末名
36EDA4165	5B
36EDA4165	226A
36EDA4165	5AEF

```

procedure search_node
while root 端末に移動するまで
{
    move_parent /* ツリーの親端末に移動 */
    m_node[i] = node_name /* 中間端末を格納 */
    rs = sql(SELECT ObjectID FROM OPL WHERE ObjectID =
        oid;) /* OPL から oid を含むレコードを取り出す */
    cnt = rs_count(rs) /* レコードの数を格納 */
    if cnt > 0 /* レコードが存在するならば */
        g_node = near_search(rs, c_node) /* 近い端末を検索 */
        break; /* 繰り返しを抜ける */
    fi
    i++
}
if root 端末
    opl = sql(SELECT ObjectID FROM R-OPL WHERE ObjectID =
        oid;) /* R-OPL から oid を含むレコードを取り出す */
    g_node = near_search(opl, c_node) /* 近い端末を検索 */
fi
while 要求端末に移動するまで
{
    move(m_node[i]) /* 移動してきた端末へ戻る */
    update_opl(opl) /* OPL を opl を適応して更新する */
}
end

```

図 5 オブジェクト配置リスト アルゴリズム

3.5 離脱と参加

複製を持った端末のネットワークの離脱は、ルートに通知を行う。ルートは離脱通知を受けた Object ID と端末名の組をデータベースから削除し、R-OPL を更新する。障害などによる通知が無い場合、最初にアクセスに失敗した端末がルートへと障害報告を行い、離脱処理を行う。参加の際は、保有している複製オブジェクトの Object ID と端末名の組をルートに通知し、OPL とオブジェクトの更新時間を受け取る。更新の必要がないオブジェクトのみオブジェクト ID と自分の端末名をルートに通知する。ルートはその通知を元に R-OPL を更新し、OPL を配信する。

4. 複製

本章では、複製に関して、複製を行う条件と複製の更新について述べる。

4.1 複製の作成

複製の作成するための条件は、複製を持つことが可能な親端末で、その端末の子端末からの平均アクセス間隔を p 、オブジェクトの平均更新間隔を q 、複製を作成しようとしている端末からの、近

接オブジェクトまでの距離を D としたとき、ネットワークコストの関係より、以下の条件が考えられる。

$$\frac{D}{p} > \frac{D}{q} + \frac{1}{p} \quad \dots (1)$$

左辺は複製を行わない場合、右辺は複製をした場合における単位時間あたりのコストである。さらに、動的ネットワークに対応させるためこの複製を行った端末が今後のアクセスの際、ネットワークに参加している可能性として、ネットワーク参加率、 λ を以下のように定め

$$\lambda = \frac{\text{ネットワーク参加時間}}{\text{計算機稼働時間}}$$

①式の右辺にを λ 掛け以下の式とする。

$$\frac{D}{p} > \left(\frac{D}{q} + \frac{1}{p} \right) \lambda \quad \dots (2)$$

で、確率を含ませ、②式をまとめる。

$$\left(\frac{1}{\lambda} - \frac{p}{q} \right) D > 1 \quad \dots (3)$$

と表すことができる。この条件式が成り立つ場合に、複製を行うものとする。

4.2 複製の更新

複製の更新手順は、オリジナルオブジェクトが更新された場合の処理と、複製を更新する場合の処理を考える。

4.2.1 オリジナルオブジェクト更新

オリジナルオブジェクトが更新された場合には下記の処理を実行する。

Step1 send_root_org_renewal

クライアントがルート端末にアップデートを通知。

Step2 update_org

ルート端末が通知を受け、R-OPL を更新。

Step 3 send_OPL

ルート端末は、R-OPLにある更新された Object ID を持つ端末に新しい OPL を送信。

Step 4 update_OPL

受信した端末は OPL の更新をする。

4.2.2 複製の更新

複製を更新する場合の処理を以下に示す。

Step 1 obj_update

アップデート可能な端末を検索し、複製を更新。

Step 2 send_root_renewal

ルートに複製の更新が完了した事を通知

Step 3 update_ROPL

ルートは R-OPL を更新

Step 4 send_OPL

ルート端末は、R-OPLにある更新された Object ID を持つ端末に新しい OPL を送信。

4.2.2 を繰り返すことで、複製端末への更新を行う。新たにオリジナルオブジェクトの更新が発生した場合には 4.2.1 を再び行う。また、オブジェクトの更新通知を複数回受けたにも関わらず、アクセスが無い場合更新されておらず、式(2)を満たさない場合、ルートに通知する。ルートは、R-OPL から削除し、更新通知を停止する。

5. シミュレーション実験

これまでに説明した方法の有効性を示すためにシミュレーションプログラムを作成し、仮想ネットワーク距離とオブジェクト配置リストにおける複製分散オブジェクトにおける 4 つの評価実験を行った。シミュレーションにおける実験条件と結果を述べる。また、以下に示す実験 1,2,3 では、比較対象として、更新を行わずに新しいオブジェクトとして登録していく方式との比較を行った。

5.1 実験 1

実験 1 では、複製を行う際に、近接オブジェクトを参照した場合での更新作業にどの程度の時間差があるかを計測する。

5.1.1 条件

- ・端末数は 200 とする。
- ・端末間ネットワーク速度は、1Mbps と仮定する。
- ・異なる 5 か所からのアクセスを行う。
- ・ランダムな順序でオブジェクトの更新を行う。
- ・オブジェクトサイズは 500KB とする。

5.1.2 結果と考察

図 6 に更新処理時間のグラフを示す。Case1 と Case4 では、更新を行わない場所と同じ場所からの複製が行われ、他の Case2,3 と 5 では、本方式

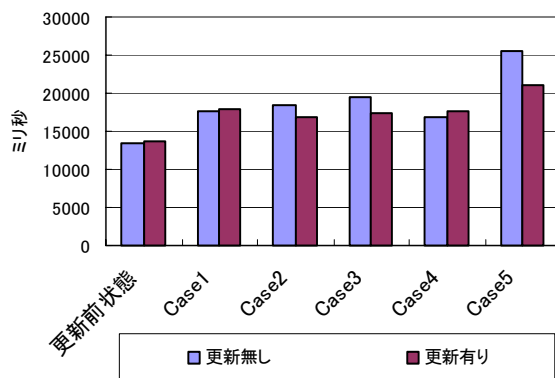


図 6 実験 1—結果

では別の近接の場所からの複製が可能となった場合である。図 6 に示すように、オブジェクト更新の際、参照先がオリジナルオブジェクトよりも近い場所から可能になった場合には、最初に行われる更新や、同じオリジナルデータから更新を行う必要な場合には、距離計算などにおける処理量が増えた分、若干遅くなるが、近い場所からの更新が可能となった場合には 1 割～2 割の処理時間の短縮とすることが可能となった。このため、全体的には効率が良くなっていることが明らかとなった。

5.2 実験 2

実験 2 では、更新後にオブジェクトに対するアクセスが発生したときに新しいオブジェクトが、どれだけ場所に更新されたかということを調べる。すなわち、実験 2 では、更新を行わない場合に新たな分散配置が配置された個数を比較する。

5.2.1 条件

- ・端末数を 300 とする
- ・オブジェクトへ 50 回ランダムにアクセスを行う。
- ・複製場所されていた場所を 20 箇所とする。

5.2.2 結果

図 7 に 50 回の更新数(更新しない場合は分散配置数)の結果のグラフに示す。実験 2 では、図 7 に示すように、アクセス量が増えるほど更新される数の差が大きくなることを確認できる。実験 2 では、更新のみしか行わないために 上限は 20 となるが、新たな複製場所を 20 以上に増加させることが可能である。

5.3 実験 3

実験 3 では、更新されていないオブジェクトから、同時に複数の更新要求が発生した場合の処理時間を測定する。これにより、分散制御による、効果を計測することができる。

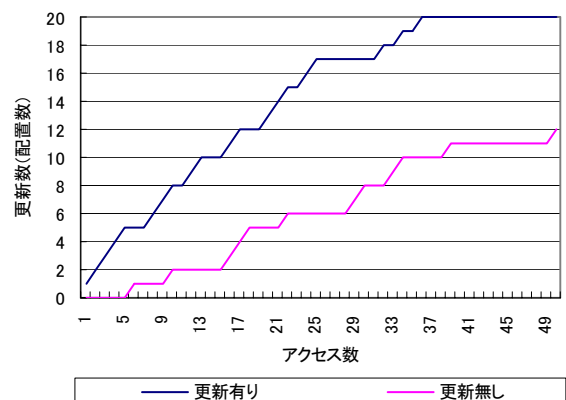


図 7 実験 2—結果

5.3.1 条件

- ・ 端末数を 1000 とする。
- ・ ネットワーク速度は、実験 1 と同じ値とする
- ・ 更新要求箇所 は 50 箇所とする。
- ・ 5 箇所ずつ同時に発生させ、前の 5 箇所が終わらないタイミングで次の更新要求を発生させる。
- ・ オブジェクトサイズは実験 1 と同様に 500KB とする。
- ・ それぞれ 5 箇所の配置変更作業が終了したときまでの時間を計測する。

5.3.2 結果

図 8 に 5 回ごとの更新処理時間の結果を示す。図 8 に示すように更新が行われるにつれ、分散処理されており、さらに更新を行わない処理よりも効率が良い分散ができていることが得られた。

5.4 実験 4

実験 4 では、ネットワークからの離脱が発生した場合に、オブジェクトの更新時間が分散的に行われるかを確認する。

5.4.1 条件

- ・ 端末数を 1000 とする
- ・ ネットワーク速度は、実験 1、2 と同様にする。
- ・ 更新済み複製を 10 箇所作成。(オリジナルオブジェクトを含む)する。
- ・ 10 箇所の更新済み複製に対し $\lambda=0.7\sim0.95$ としてネットワークから離脱する場合を考える。
- ・ 更新要求を 2 箇所ずつ 10 回、計 20 箇所発生させる。
- ・ オブジェクトサイズは、実験 1、2 と同様に 500KB とする

5.4.2 結果

図 9 に 10 回の更新処理の処理時間を測定した結果を示す。図 9 では、端末が離脱した時に、ルートからのオブジェクト配置リストの配信が行われているため、余分なアクセスがほとんど見られず、ほぼ一定の値を示している。しかし、3 回目のみ オブジェクト配置リストの配信前にアクセスが発生

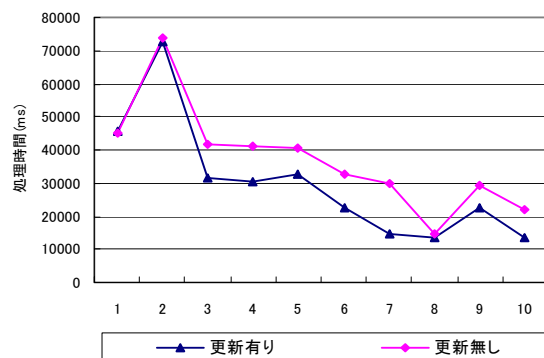


図 8 実験 3—結果

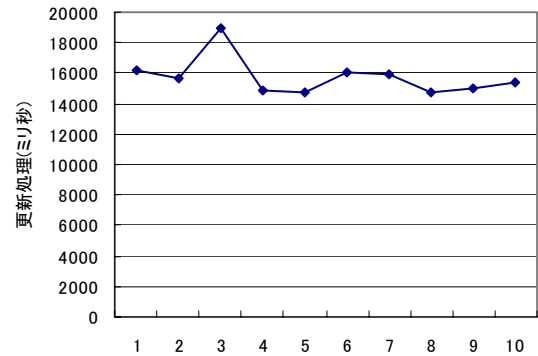


図 9 実験 4—結果

し更新の際に無いオブジェクトを参照したため、処理時間が増加した。この点については、今後、OPL の配置と配信方法を考える必要がある。

6. おわりに

本稿では、改良型動的複製制御方式による分散オブジェクト管理を提案した。提案した方法は、動的な環境における情報共有サービスにおける更新を分散して行う方法であり、ネットワーク負荷を小さくすることが可能で、さらに、更新前に分散された配置を更新後も利用可能である。

シミュレーション実験の結果、分散した更新制御の有効性が示された。なお、提案する方法は今後下記の点を解決する必要がある。

- ・ 複製の条件において、ネットワークの帯域幅を考慮していないため、狭帯域ネットワークに複製が作成されてしまうおそれがあるため、複製を行う条件式のさらなる検討を行う必要がある。
- ・ 端末数が増えることで、複製配置場所も増えるため、ルートにかかる負荷の増大があげられる。ルートの負荷を減らす方法の検討をし、OPL の配置方法も検討する必要がある。

参考文献

- 1) C.Greg Plaxton, Rajmohan Rajaraman, Andrea W, Richa : “Accessing Nearby Copies of Replicated Objects in a Distributed Environment.” Proc of the SCP SPAA,(1997)
- 2) Yan Chen, Randy H.Katz and John D.Kubiatowicz : “Dynamic Replica Placement for Scalable Content Delivery” , Department of EECS, University of California at Berkeley(2002)
- 3) Ben Y.Zhao, John Kubiatowicz , and Anthony D.Joseph : “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing”,Computer Science Division University of California Berkeley, UCB/CSD-01-1141(2001)
- 4) Ben Y.Zhao : “A Decentralized Location and Routing Infrastructure for Fault-tolerant Wide-area Network Applications”, Ph.D. Qualifying Examination(2001)
- 5) 浅井健一, 川越恭二:「広域分散環境における複数データベース間でのレプリカ制御方式」,情報処理学会 63 回 全国大会講 演論文集(2002)