Aerie: WWW のための完全分散型匿名プロキシ

阿部洋丈[†] 加藤和彦[†,††]

本論文では，WWW 利用時におけるプライバシの侵害を防ぐための分散システム *Aerie* について述べる．Aerie では，システムに参加しているノードどうしが協調して互いに通信を肩代わりし合い，WWW サーバに対して送信者の匿名性を実現する．Aerie は完全分散型の Peer-to-Peer システムであり，中央のサーバなしに動作する．Aerie は，システム内における高速なノード間通信のために Chord を用いる．Aerie は，Chord を用いる場合に生じる匿名性上の問題を解決している．

Aerie: Fully Distributed Anonymizing Proxy for WWW

Hirotake Abe[†] and Kazuhiko Kato[†,††]

In this paper, we propose a fully distributed system called *Aerie*. It prevents privacy violation which may occur when browsing WWW. Each Aerie node cooperate each other to hide their identity from WWW servers, assuming others' HTTP requests. With Peer-to-Peer technology, Aerie doesn't require any centralized servers. Aerie uses a distributed lookup protocol called Chord for fast inter-node communication. We solve an anonymity problem which occurs when using Chord in Aerie.

1. はじめに

近年，WWW の利用が拡大し，社会のさまざまな局面で重要な役割を担うようになった．その一方で，WWW による個人情報の流出を危惧する声も高まっている．WWW で用いられている通信プロトコル HTTP は，インターネット用プロトコル IP の上位層に実現されている．そのため，HTTP による通信では，互いに相手を特定するために，IP における識別子である IP アドレスが用いられる．IP を用いて相互に通信を行うためには，互いに相手の IP アドレスを知っている必要がある．HTTP はクライアント・サーバモデルに基づいているので，HTTP 通信を開始するためにはクライアントはサーバの IP アドレスを事前に知っている．だが，クライアントの IP アドレスは必ずしもサーバに判明する必要はない．なぜならば，サーバは，IP アドレスが分からなくても，クライアントにデータを転送できれば十分だからである．しかし，クライアントがサーバと直接通信する場合は，クライアントの IP アドレスがサーバに判明することは避けられない．

IP アドレスはインターネットに参加している計算機の中から一意に計算機を特定するための識別子である．そのため，通信相手の IP アドレスを知るということは，相手の計算機を特定するということと等しい☆．計算機が特定されると，それにともなって，その計算機を利用している個人まで特定されてしまう可能性がある．いったん，IP アドレスと個人の結び付きが判明してしまうと，その個人がどの情報に興味を持っているか，どのサービスを利用したか等の，個人のプライバシに関する情報が次々と漏洩する危険がある．

プライバシを保護するためには，匿名性の実現が重要である．一般に，通信における匿名性には，

- 送信者の匿名性（sender anonymity）
- 受信者の匿名性（recipient anonymity）
- 送信者と受信者の非結合性（unlinkability of sender and recipient）

があるとされている[10]．この中で，WWW において特に重要なのは送信者の匿名性，つまり，通信を傍受しても，誰がそのメッセージを送信したのかは分からないということである．

WWW サーバに対しての送信者の匿名性は，通信

† 科学技術振興機構
Japan Science and Technology Agency
†† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Science, University of Tsukuba

☆ ルータにおいてアドレス変換を用いている場合は一意に特定はできない．しかし，そのルータを利用している計算機の中のどれかであるということはいえる．

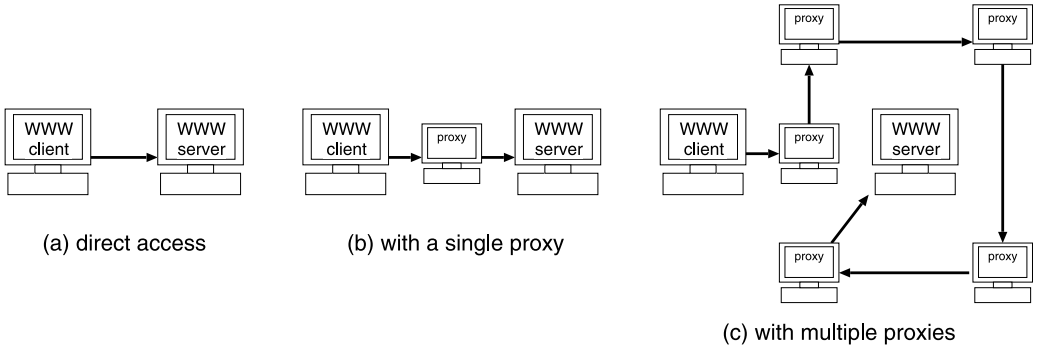

図 1 接続形態の比較
Fig. 1 Comparison of interconnect configurations.

相手の計算機とは異なる第三者の計算機に通信を肩代わりしてもらうことで実現可能である．現状では，WWW サーバに対して送信者の匿名性を得ようとするユーザは，Anonymizer.com [1] に代表される匿名プロキシサービスを利用している．匿名プロキシは，WWW のサーバへの通信を肩代わりし，WWW サーバに IP アドレスを開示せずに WWW サービスを利用することを可能にする．しかし，匿名プロキシにはリクエストの送信者が容易に判明してしまうという問題がある．

匿名プロキシに対しても送信者の匿名性を実現するためには，自分自身も匿名プロキシとなり，他の計算機からのリクエストをさらに他のプロキシへ転送することが重要である．そうすることにより，自分が送信したリクエストが，自分が最初に発行したものなのか，他の計算機から来たものを転送しただけなのかを区別することが困難になる．

本論文では，WWW における送信者の匿名性のための分散型システム Aerie について述べる．Aerie は，システム内の他の計算機にサーバへの通信を肩代わりさせる機能を提供する．Aerie は Gnutella [5] や Freenet [2] 等と同様の完全分散型の Peer-to-Peer（P2P）システムであり，専用のサーバを必要としない．

Aerie では，専用のサーバを用いずに，通信を肩代わりさせる計算機を選択し発見するために，分散ハッシュ表による探索方式 Chord [14),15)] を利用している．それにより，完全分散型の P2P システムでありながら十分実用的なパフォーマンスを得ることができる．しかし，我々は Chord 方式を単純に適用した場合，匿名性に関して問題が生じることを発見した．Aerie では，Chord による探索手順にランダム性を持たせることによってこの問題に対処している．

2. Aerie の設計

2.1 概　　観

通常，WWW において，クライアントはサーバに直接接続してリクエストを送信する（**図 1** (a)）．そのため，サーバはクライアントの IP アドレスを知ることが可能である．プロキシは，クライアントから受け取ったリクエストに従って，クライアントの代わりに目的のサーバに接続しリクエストを送信する（図 1 (b)）．この場合，サーバは送信者の IP アドレスを直接知ることはできない．しかし，プロキシはクライアントの IP アドレスを知ることができる．

サーバに対してのみでなく，プロキシに対しても送信者の匿名性を実現するためには，図 1 (c) のように，いくつかのプロキシを介して通信を行い，誰が最初にリクエストを送信したのかを容易には判断できなくすることが重要である．そのような通信を実現するためのシステムとして Crowds [11] が知られている．Crowds は，協調し合う複数のノードのグループで構成される分散システムである．各ノードはそれぞれプロキシの役割を果たし，他のノードから送信されたリクエストを他へ中継する．Crowds では，クライアントからのリクエストを受信したノードは，自分がそのリクエストをサーバに送信するか，もしくは，さらに他のいずれかのノードに送信するかをランダムに選択する．この動作を繰り返すことにより，図 1 (c) のような接続形態が構成され，最初にリクエストを送信したノードを容易には特定できなくなる．Crowds はオープンなシステムであり，誰でも参加できるように設計されている．

Crowds は，グループのメンバシップは Crowds 専用のサーバによって集中的に管理される．サーバによって集中管理する方式は，ノード数の増加に対して

スケーラブルに動作することが難しい．また，そのサーバが単一故障点であるため，そのサーバの障害はシステム全体の機能不全を意味する．そのため，可用性の低下や，DoS（Denial of Service）等の攻撃に対する脆弱性等の問題がある．

Aerie は，Crowds と同様の，WWW における送信者の匿名性の実現するための分散システムである．Aerie は完全分散型の P2P システムであり，メンバシップを集中的に管理するサーバを必要としないことを目標にして設計された．Aerie は Crowds と同様にオープンなシステムとして設計されているので，誰でも Aerie に参加することができる．また，Aerie においてすべてのノードはまったく平等な立場である．そのため，認証によるユーザの識別は行わない．

メンバシップを集中管理せずにランダムなノードの選択を実現することは容易ではない．Crowds のようにメンバシップが集中管理されていれば，Crowds のサーバからノードの一覧を入手して，その中から選択することができる．しかし，メンバシップが集中管理されていない場合は，システム内の全ノードを誰も把握していないので，そのようなリストを構成することが困難である．

完全分散型のシステムにおいてランダムなノードの選択を実現するために，Aerie は Chord と呼ばれる方式を応用する．Chord は，完全分散型の P2P システムにおいて効率的な探索を実現するための方式である．Chord は，検索のためのキーをシステム内の各ノードに写像する機能を提供する．Chord では，集中管理を行うサーバが存在しない代わりに，各ノードがそれぞれ小さな経路表を保持している．キーの写像されたノードを発見するためには，経路表に基づいて複数のノードに繰り返し問合せを行う必要がある．N ノードの中から目的のノードの発見に要する問合せの回数は，定常状態では，最悪でも $O(\log N)$ 回である．Chord は他の類似の方法[8),12)] に比べてアルゴリズムがシンプルであり，探索のコストの上限や，負荷分散性能等に関する性質が証明されているという利点がある．Chord の探索アルゴリズムについては，2.2 節で簡単に説明する．

以降，Chord を用いた Aerie の動作を直感的に説明する．**図 2** は，Aerie のノードによってリクエストが中継される様子を表している．説明のため，最初にリクエストを送信するノードを**開始ノード（initiator node）**，他のノードからのリクエストをさらに他へ転送するノードを**中間ノード（intermediate node）**，サーバにリクエストを送信するノードを**終端ノード（terminal node）**と呼ぶ．

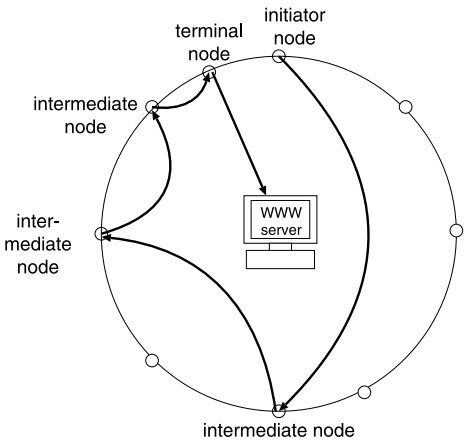

図 2 Aerie の動作
Fig. 2 Behavior of Aerie system.

Crowds においては，開始ノードは次にリクエストを送信すべき中間ノードをランダムに選択する．それに対し，Aerie においては，開始ノードは初めに終端ノードをランダムに選択する．開始ノードは，直接ノードを選択する代わりに，ランダムなキーを発生させ，そのキーが写像されるノードを終端ノードとして間接的に選択する．

Aerie において，中間ノードの選択は，終端ノードの場合と同様のランダムな選択によってではなく，Chord のアルゴリズムよる探索過程を利用して決定される．そうすることによって，終端ノードを探索する手続きと，中間ノードの選択，およびそれらを経由させながらリクエストを送信する手続きが同時に実行され，少ないメッセージのやりとりで一連の手続きが実行可能である．

いくつかの中間ノードによる中継を経て，リクエストが終端ノードに到達すると，終端ノードはサーバに接続してリクエストを送信する．サーバとの通信に成功してデータが得られると，終端ノードは自分にリクエストを送信してきた中間ノードにそのデータを送信する．中間ノードは，中継先のノードにリクエストを送信した後もその接続を維持しているので，その接続を経由してデータを送信することができる．このようにして，リクエストが中継された経路を逆にたどって，開始ノードにデータが返される．

中間ノードや終端ノードが応答しなくなってしまう場合を想定し，本システムはタイムアウト処理を行う．各ノードは，一定時間経過しても応答が得られない場合は，タイムアウトと見なして接続を切断する．開始ノードは，タイムアウトを検出すると，別のランダム

キーを用いて新たに通信を開始する．

2.2 Chord [14),15)] による探索

Aerie の通信方式を詳述するための準備として，Chord による探索方式を概説する．

Chord では，各ノードには，SHA-1 [9)] 等のハッシュ関数によって計算された識別子が与えられる．ハッシュ値の計算には，そのノードの IP アドレス等の固有の情報が用いられる．検索するためのキーに対しても，同じハッシュ関数によって識別子が計算される．ハッシュ関数によって出力されるバイト列の識別子を整数値としてとらえると，それぞれの識別子には順序が定義される．識別子は modulo 2^m（m は識別子のビット数）の識別子の輪（identifier circle）によって順序づけられる．

Chord において，キーは，そのキーから計算された識別子 k と等しいか，もしくは k よりも大きい識別子を持つノードの中で最も k に近いノード上に写像される．このとき，そのノードは k の **successor node** と呼ばれる．逆に，k よりも小さい識別子を持つノードの中で最も k に近いノードを，k の **predecessor node** と呼ぶ．

あるノードから k の successor node を発見することは，すべてのノードが自分の識別子の successor node を知っていれば，全探索を行うことで最低限実現可能である．しかし，それでは探索に多くの問合せを必要としてしまう．Chord では，successor node の効率的な発見のために，各ノードは **finger table** という表を保持する．ノード n の finger table には，$n + 2^{i-1} \bmod 2^m (1 \leq i \leq m)$ の successor node が保持されている．これらのノードのことを **finger** と呼ぶ．

Finger table を用いて k を探索する手順は次のようになる．(1) まず，finger table の中から，k よりも前にあり k に最も近い finger を調べる．その finger は **closest preceding finger** と呼ばれる．(2) 次に，その closest preceding finger の保持する finger table に対して同様の探索を行う．その動作を，k の predecessor node に到達するまで繰り返す．もし k の predecessor node が見つかれば，そのノードの次のノードが k の successor node ということになり，そのノードが k が写像されたノードということになる．

2.3 Aerie の通信方式

Chord を用いて Aerie を実現する方法について説明する．

Aerie は，WWW ブラウザ等からリクエストを受け取ると，乱数を用いてキーを生成する．Aerie は，それを識別子として Chord による探索を開始する．Chord による探索には，反復的および再帰的という 2 つのバリエーションがある [14),15)]．Aerie では，再帰的な探索を用いている．再帰的な方法とは，探索を行うノードが closest preceding finger の問合せを行うと，successor node が発見されるまで問合せが再帰的にノード間を連鎖して行われる方式である．

```
n.forward_request(id, request) ≡
  if (id is nil)
    id = n.get_random_number();
  n' = n.search_next_node(id);
  if (n' is nil)
    return submit_request_to_server(request);
  else
    return n'.forward_request(id, request);

n.search_next_node(id) ≡
  if (id ∈ (n, n.successor])
    return nil;
  else
    return n.closest_preceding_finger(id);
```

図 3 擬似コードによるアルゴリズム
Fig. 3 The pseudocode of inter-node communication.

Chord による再帰的探索によって，識別子の successor node に次第に近づいていく．Chord では，ある識別子の successor node に到達する前には，必ずその識別子の predecessor node を経由する．Aerie は，successor node に到達する前に，predecessor node に到達した時点で探索を終了する．つまり，successor node ではなく predecessor node を終端ノードとする．

ノード間の通信に用いられている手続きのうち，基本となる 2 つのアルゴリズムを**図 3** に示す．これらのアルゴリズムは，ノード n に対する遠隔手続きとして書かれた擬似コードである．擬似コード内に現れる “n.*some_procedure*()” は，ノード n に対して手続き some_procedure を遠隔呼び出しすることを表す．また，$(x, y]$ は半開区間を表す．図 3 の中で呼び出されている他の手続きの役割は次のとおりである：手続き submit_request_to_server は，与えられた request をサーバへ送信し，サーバから受信したデータを返す．手続き get_random_number は，探索に用いるランダムな識別子を返す．手続き closest_preceding_finger は，finger table を調べて closest preceding finger を返す．

手続き forward_request は，Aerie によるノード間通信の中核となる，再帰的な探索手続きである．開始ノードは，クライアントからリクエストを受け取ると，自分自身のノードに対してこの手続きを呼び出す．そのとき，引数 request にはクライアントからのリクエストを，引数 id には nil 値を与える．中間ノードおよび終端ノードは，他のノードによってこの遠隔手続き

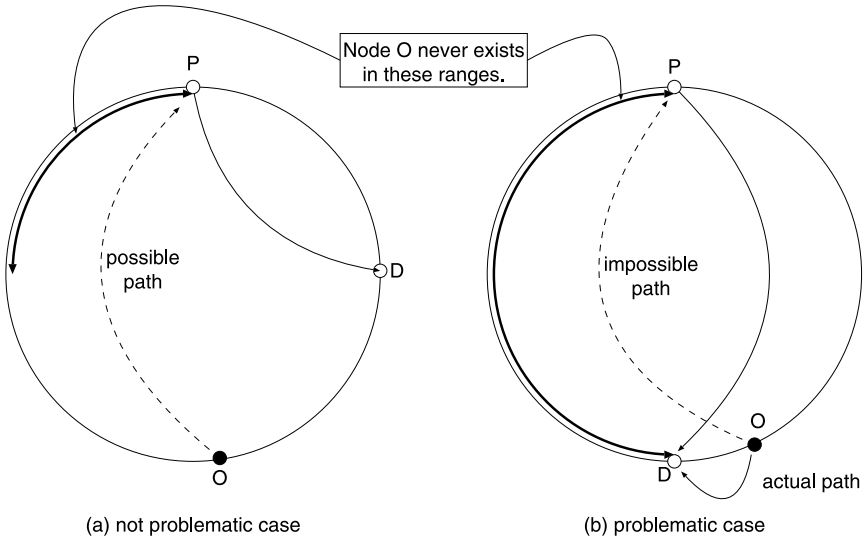

図 4 問題点
Fig. 4 The problem.

が呼び出される．

手続き forward_request では，まず引数 id が nil でないかを検査し，nil であったら get_random_number によってランダムな識別子を得る．次に，引数 id の値を用いて，次にリクエストを送信すべきノードを調べる．この処理は search_next_node として独立した手続きになっている．もし，search_next_node の返戻値が nil であったならば，それは自分が終端ノードであることを意味する．その場合は，手続き submit_request_to_server を用いてリクエストをサーバに送信する．返戻値が nil でない場合は，次のノードにリクエストと識別子を送信するために，次のノードの forward_request を呼び出す．

手続き search_next_node は，ノード n の持つ情報を参照し，与えられた識別子 id の写像されたノードへ近づくためには，次はどのノードに問い合わせるべきかを判断する．もし，id $\in$ (n, n.*successor*] である場合は，n は id の predecessor node であるということになり，n が終端ノードであることが判明する．その場合は nil を返す．そうでない場合は，手続き closest_preceding_finger の結果を返す．

3. 匿名性上の問題点とその解決法

3.1 問 題 点

Aerie のリング内部における匿名性は，あるノード P からあるノード D へのリクエストが，P が自発的に発行したのか，P 以外の他のノード O が P へ発行したのを転送しているのかを D は容易には区別できないということに基づいている．しかし，Chord による探索の特徴を考慮すると，ある条件のもとで P が開始ノードであることが D に判明してしまう．その条件とは，D が P の m 番目の finger，すなわち D が $P + 2^{m-1} \bmod 2^m$ の successor node である場合である．P が WWW ページを参照しようとする場合，まず自分の持つ finger table を見て，次にリクエストを転送すべきノードを探す．探索のための識別子が自分の識別子と 2^{m-1} 以上離れている場合はすべて $P + 2^{m-1} \bmod 2^m$ の successor node である D に転送される．しかし，他のノード O から P を介して D にリクエストを転送することはありえない．これについての形式的な説明を付録 A.1 に添付する．

この結論を直感的に表しているのが**図 4** である．ノード間で転送されるメッセージは，転送されるたびに次第に目的のノードに近づいていく．その際，転送されるたびに送受信者間の距離は単調減少する．もし，D が P の m 番目の finger だとすると，P と D の距離は 2^{m-1} 以上 — 角度でいうと 180° 以上 — 離れているはずである．その場合，P と D の距離よりも O と P の距離は長くなければならない．しかし，そうすると O と D の位置関係が逆転し，もともと P を介すはずがないことになる（図 4 (b)）．そのため，P から D へのリクエストは転送ではなく，P 自身がそのリクエストを発行したことが確実に判明してしまう．つまり，180° 以上離れたノードに転送されるメッセージは，すべて P 自身が発行したリクエストである．

この現象は m 番目の finger のノードにリクエスト

```
n.forward_request(id, request) ≡
  if (id is nil or get_random_double() < p_r)
    id = n.get_random_id();
  n' = n.search_next_node(id);
  if (n' is nil)
    return submit_request_to_server(request);
  else
    return n'.forward_request(id, request);
```

図 5 ランダム性を付与された forward_request の疑似コード
Fig. 5 Randomized behavior.

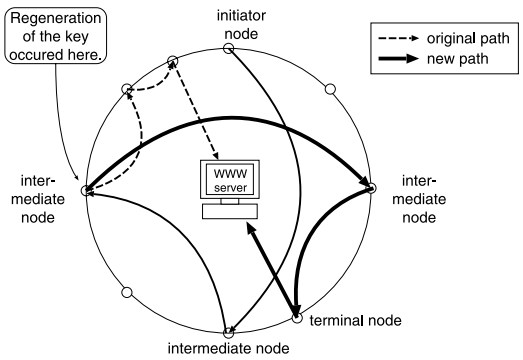

図 6 ランダム化された探索動作
Fig. 6 The modified pseudocode with randomness.

を転送する場合に生じる．各ノードが適度に分散していることを仮定すると，P の自発的なリクエストのうち約半数が m 番目の finger のノードに転送され，それが自発的なリクエストであるということが判明する．

3.2 解 決 法

この問題を解決するために，Aerie では，Chord による探索に次の方法でランダム性を付与する．他のノードからリクエストの転送要求を受け取ったノードは，確率 p_r でキーの再生成を行う．キーの再生成が行われると，それまでの古いキーによる探索は中止され，新しく生成されたキーによる探索が開始される．**図 5** は，この方法を採用したバージョンの手続き forward_request である．なお，この手続き中の get_random_double は，区間 $(0, 1)$ 内の double 型の一様乱数を返す関数である．

図 6 は，途中でキーの再生成が 1 回起きた場合の通信の例を表している．この例では，2 番目の中間ノードでキーの再生成が起きている．その結果，そのノードは，古いキーによって決定される次の中間ノードではなく，新しいキーによって決定される中間ノードにリクエストを転送する．その後，新しいキーによって決定された，新しい終端ノードまでリクエストは転送され，そのノードから WWW サーバへリクエストが送信される．

このようにして探索にランダム性を与えることによって，あるノードが 2^{m-1} 以上離れたノードへリクエストを転送したとしても，それだけでそのリクエストが自発的なものであるとは判明しない．なぜならば，そのリクエストが自発的なものなのか，再生成によって新たに探索が開始された直後なのかを区別することができないからである．

上記の方式により匿名性上の問題は解決されるが，その代償として通信性能が悪化する．それは，キーの再生成を行うことによって，開始ノードから終端ノードまでの通信を中継する中間ノードの数が増加するためである．どれほどノード数が増加するかは，キーの再生成を行う確率 p_r に依存する．確率 p_r が大きいほど匿名性は向上すると考えられるが，その代わりに通信性能が低下する．

この方式では，非常に多数の転送が繰り返され，ユーザが非常に長い時間待たされてしまう恐れがある☆．そのような場合は，前述のタイムアウト処理が行われ，通信がやり直される．これにより，非常に長い時間ユーザが待たされ続けることを防ぐことができる．

4. 実装と性能評価

4.1 実 装

Java 言語を用いて Aerie を実装した．Aerie は，Sun Microsystems 社の J2SDK 1.4 で動作するように設計した．Aerie では，各ノードの識別子は，そのノードの IP アドレスおよびポート番号を元に，SHA-1 によって計算される．Chord における finger table の更新方法には積極的な方法と消極的な方法がある．積極的な方法は，ノードが新たに参加したという情報を積極的に伝播させる方法である．消極的な方法は，あるノードが新たに参加したという情報を積極的に伝播させず，各ノードによるポーリングで徐々に伝播させるという方法である．消極的な方法は，伝播が即時に行われないので一時的に性能が低下するという欠点があるが，複数のノードが同時に参加したり，ノードが障害を起こして突然離脱したりした場合等でも探索が妨げられないという利点がある．Aerie では，これらの特徴を利用するために，消極的な方法を採用している．本実装では，複数のリクエストを並行して処理するために，リクエストごとにスレッドを割当てる．スレッドの割当ては，スレッドプールを用いて実現されてい

☆ このような場合に対する対処としては，TTL（Time To Live）を用いる方法が一般的である．しかし，TTL が匿名性を低下させる原因になりうるので，本提案システムでは導入を行わなかった．

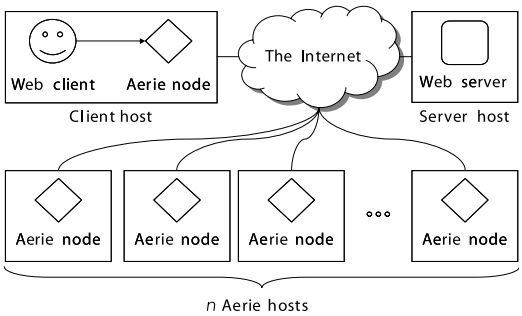

図 7 シミュレーションに用いたネットワーク構成
Fig. 7 The network configuration used in the simulation.

表 1 Aerie および HTTP 関連の設定
Table 1 Configurations of the simulation: Aerie and HTTP.

項目	内容
シミュレーション時間	60 時間
finger table の stabilize 間隔	5 秒
各ホスト間のリンクの帯域幅	100Mbps
各ホスト間のリンクの伝搬遅延	15 ミリ秒
クライアントによるセッションの間隔	指数分布 ($\lambda = 0.01$)
セッションあたりの参照ページ数	指数分布 ($\lambda = 0.2$)
ページ参照の間隔	パレート分布 ($k = 25.0,\ \alpha = 2.0$)
ファイルのリクエストの間隔	パレート分布 ($k = 0.1667,\ \alpha = 1.5$)
ページから参照されるオブジェクト数	パレート分布 ($k = 0.6667,\ \alpha = 1.2$)
オブジェクトのサイズ	パレート分布 ($k = 2000.0,\ \alpha = 1.2$)

表 2 TCP 関連の設定
Table 2 Configurations of the simulation: TCP.

項目	内容
初期シーケンス番号	10000
最大セグメントサイズ	2000 バイト
受信ウィンドウサイズ	32 個
最大送信ウィンドウサイズ	32 個
送信バッファサイズ	128 個
最大再送回数	12 回
TCP slow timer の間隔	0.5 秒
TCP fast timer の間隔	0.2 秒
最長セグメント寿命	60 秒
最長アイドル時間	600 秒
遅延 ACK	なし
高速復帰アルゴリズム	あり

る．Aerie ノード間の通信は，現在の実装では暗号化は行っていない．

4.2 性能評価

Aerie を利用して WWW を閲覧する場合にどれくらいの通信遅延が発生するかを調べるために，Scalable Simulation Framework（SSF）[13] を利用してシミュレーション実験を行った．SSF の実装は，Renesys 社の Raceway 1.5 を利用した．

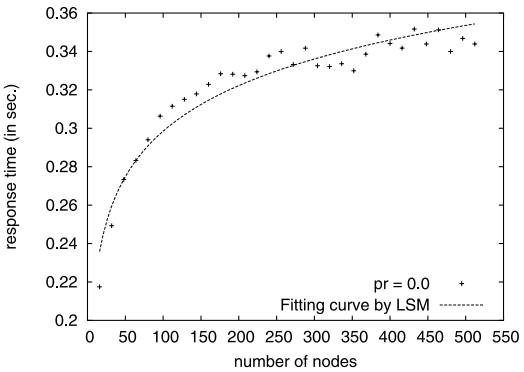

図 8 シミュレーション結果（$p_r = 0.0$ の場合）
Fig. 8 Simulation results ($p_r = 0.0$).

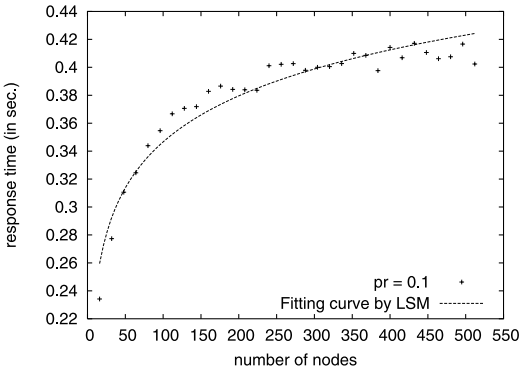

図 9 シミュレーション結果（$p_r = 0.1$ の場合）
Fig. 9 Simulation results ($p_r = 0.1$).

シミュレーション実験を行うために，インターネットをモデル化した簡単なネットワークを SSF 上に構成した．その構成を**図 7** に示す．1 つのサーバホストと，1 つのクライアントホストと，多数の Aerie ホストがインターネットで相互に接続されている．サーバホスト上には HTTP サーバ（Raceway 1.5 に付属する SSF.OS.WWW.httpServer クラスのオブジェクト）を配置した．クライアントホストには，HTTP クライアント（SSF.OS.WWW.httpClient クラスを，プロキシサーバを参照するように改造したもの），および Aerie ノードを配置した．Aerie ホストには Aerie ノードを配置した．シミュレーションに用いた諸々の設定を**表 1**，**表 2** に示す．

図 8，**図 9**，**図 10**，**図 11**，**図 12**，**図 13** は，それぞれ $p_r = 0.0$，0.1，0.2，0.3，0.4，0.5 とした場合の，ノード数と応答時間の関係のシミュレーション結果である．$p_r = 0.0$ の場合とは，Chord による探索そのものを意味する．つまり，匿名性上の問題が解決されていない状態を意味する．図中の点は，そのノード数のときの応答時間の平均を表している．また，図

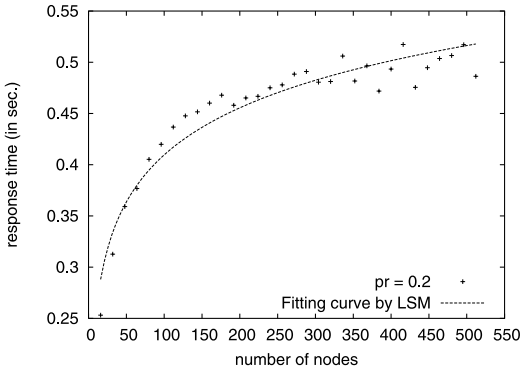

図 10 シミュレーション結果（$p_r = 0.2$ の場合）
Fig. 10 Simulation results ($p_r = 0.2$).

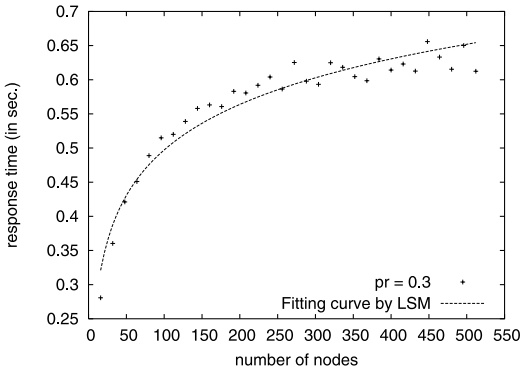

図 11 シミュレーション結果（$p_r = 0.3$ の場合）
Fig. 11 Simulation results ($p_r = 0.3$).

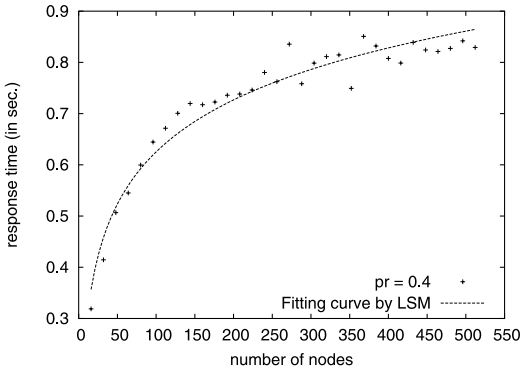

図 12 シミュレーション結果（$p_r = 0.4$ の場合）
Fig. 12 Simulation results ($p_r = 0.4$).

中の曲線は，ノード数 x および応答時間 y の関係が $y = a \log_{10} x + b$ と表現できると仮定した場合の，最小二乗法による推定結果である．それぞれの p_r の場合の定数 a，b の推定結果を**表 3** に示す．なお，最小二乗法による推定には，gnuplot 3.7 を利用した．

図 14 は，最小二乗法によって得られた曲線を，x 軸が対数スケールのグラフに描画したものである．これにより，ノード数が多数になった場合の応答時間が予想可能である．また，**図 15** は，$p_r = 0.0$ の場合に対する倍率を計算して描画しものである．この結果は，ノード数の増加に応じた応答時間の増加を表している．図 15 より，ノード数が大きくなるほど，$p_r = 0.0$ の場合に対する倍率が大きくなっていることが読みとれる．

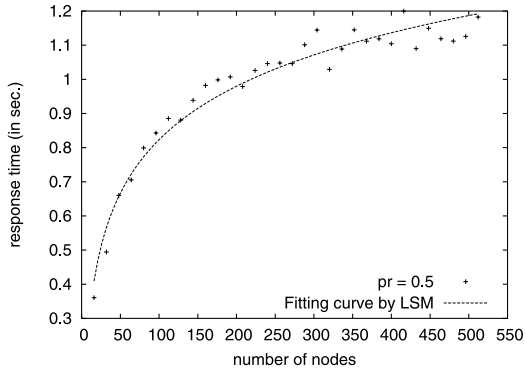

図 13 シミュレーション結果（$p_r = 0.5$ の場合）
Fig. 13 Simulation results ($p_r = 0.5$).

表 3 最小二乗法による a, b の推定結果
Table 3 Estimated values of a, b by Least Square Method.

p_r	a	b
0.0	0.0787539	0.140993
0.1	0.109431	0.127736
0.2	0.152708	0.104142
0.3	0.221459	0.0542545
0.4	0.337385	−0.0496103
0.5	0.520033	−0.216781

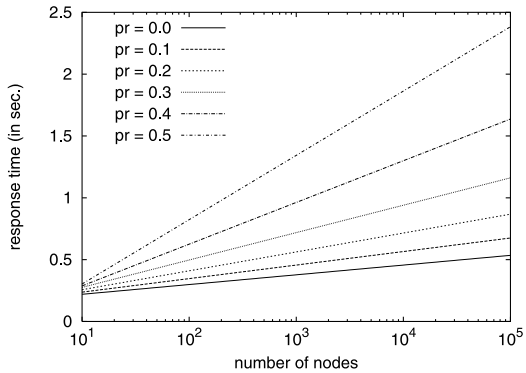

図 14 最小二乗法による結果の比較
Fig. 14 Comparison of response time estimated by Least Square Method.

5. 議　　論

Aerie はオープンなシステムなので，悪意を持ったユーザが，意図的に改変したノードを Aerie システ

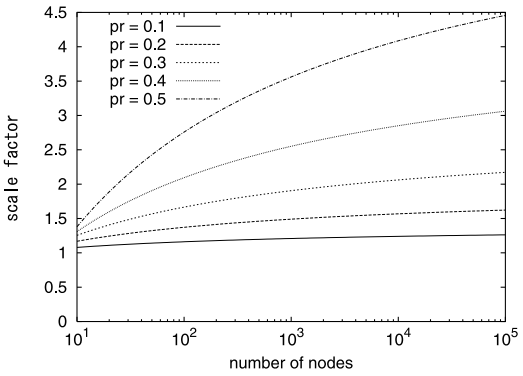

図 15 $p_r = 0.0$ と比較した場合の倍率
Fig. 15 Scale factors of response time comparing to $p_r = 0.0$.

ムに混入させるという事態が起こりうる．その結果，Aerie の持つ匿名性や可用性が攻撃される可能性がある．もし，そのような悪意を持つユーザが Aerie に参加するのを防ぐことができれば，そのような可能性を回避することができる．しかし，しかし，悪意のあるユーザとそうでないユーザを識別することは難しいので，この方法は現実的ではない．そのため，そのような攻撃に対する対処を体系的に組み込むことを検討しておく必要がある．

匿名性に対する攻撃には，共謀攻撃（Collaborating attacks），トラフィック解析（Traffic analyses），タイミング解析（Timing analyses）がある．

共謀攻撃とは，共謀する複数のノードがシステム内に潜入し，それぞれが知りえた情報を集約し，その結果を元に送信者の特定を試みる攻撃である．文献 6), 16) では，この種の攻撃に対する詳細な解析が行われている．Aerie のモデルは，これらの文献で用いられているモデルにはあてはまらないので，そのまま結果を適用することはできない．Aerie を元にした確率モデルを構築し，それに対して同様の解析を行うことは今後の課題である．

トラフィック解析とは，あるホストから上位のネットワークへのトラフィックを監視することで，そのホストから送信されたメッセージが，そのノードが自発的に送信したのか，他のノードから依頼されて転送しているのか判別するという攻撃である．これは，定期的に Aerie ノードがダミーのトラフィックを発生させることで防ぐことができる．

タイミング解析とは，メッセージを送るタイミングを観測して送信者を特定を試みる攻撃である．これは，HTML ファイルを受け取ったノードは，そのファイルから参照されている URL をすぐに見にいくというヒューリスティックに基づいている．この攻撃に対しては，文献 11) で提案されている，目的のファイルからリンクされている他のファイルも終端ノードがまとめて取得するという手法はによって防ぐことができる．

可用性に対する攻撃で最も懸念されるのは，いわゆる DoS（Denial of Service）攻撃である．DoS 攻撃の手段としては，意図的にメッセージの転送を怠ったり，わざと遅らせたり，誤ったルーティングをしたりする，という方法が考えられる．また，可用性に対する別の攻撃としては，メッセージを転送する際にそのメッセージを改変することが考えられる．

これらの攻撃に対しては，リクエストを多重化することで対処できると考えている．開始ノードは，リクエストを発行するときに 1 つのリクエストを複数の終端ノードに向けて送信する．そうすることで，もし経路の途中にメッセージの転送を怠るノードがいたとしても，別の経路で送られたリクエストのレスポンスが返ることが期待できる．また，もし改変があった場合は，複数のレスポンスを比較することで改変を検出できることが期待できる．

6. 関 連 研 究

Tarzan[3] は，HTTP ではなく，IP 層において Aeire と同様の匿名通信を実現する P2P システムである．Tarzan は，IP で通信を開始する際に，システムに参加している他の複数のホストを経由するトンネルを作成する．IP パケットはすべてそのトンネルを介してサーバに送信される．Aerie は，Tarzan と違い，通信に際して明示的にトンネルを作成する必要はない．そのため，各ノードはセッション情報を管理する必要がないので，より高いスケーラビリティを実現する．Tarzan は，以前のバージョン[4] では Aerie と同様に Chord を利用していた．現在のバージョンでは，IP プレフィクスに基づいたドメイン分割を利用する探索アルゴリズムを採用している．これは，連続した IP アドレスを持つ複数ホストによる共謀攻撃を排除するためである．

Achord[7] は，Chord を利用して匿名性のためのシステムを実現するのではなく，Chord 自体に匿名性を持たせようとする試みである．キーの探索時に反復的な方法ではなく再帰的な方法を使って匿名性を確保しようとしている点は本提案と同じであるが，本論文で指摘した匿名性上の問題点に関しては言及されていない．

7. おわりに

本論文では，WWW のための分散型匿名プロキシ Aerie の実現について述べた．Aerie は，中央サーバを必要としない完全な P2P アプリケーションでありながら，分散ハッシュ表による探索方式 Chord を利用することで実用的なパフォーマンスを実現している．我々は，SSF によるシミュレーションによって Aerie 利用時のオーバヘッドを示した．また本論文では，Aerie で Chord を用いる場合に発生する匿名性に関する問題点について述べ，解決法を示した．

今後は，各種の攻撃に対する防御策を検討したい．特に，5 章で述べたように，文献 6)，16) と同様の，匿名性に関する確率的な解析を行い，その結果を元により高い匿名性を実現できるよう改良を行いたい．また，より大規模な実験環境を用いた性能評価実験を行いたい．

謝辞 筑波大学システム情報工学研究科の姚強氏には，SSF によるシミュレーションを手伝っていただいた．東京大学大学院情報理工学系研究科の大山恵弘氏，東京工業大学大学院情報理工学研究科の光来健一氏には，論文の執筆に際して有益な助言をいただいた．また，実験に際しては，東京大学情報理工学系研究科コンピュータ科学専攻米澤研究室，京都大学大学院情報学研究科通信情報システム専攻湯淺研究室の協力をいただいた．

参考文献

1) Anonymyzer.com. http://www.anonymizer.com/
2) Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System, Lecture Notes in Computer Science, Vol.2009, pp.46+ (2001).
3) Freedman, M.J. and Morris, R.: Tarzan: A peer-to-peer anonymizing network layer, *Proc. 9th ACM Conference on Computer and Communications Security* (*CCS 2002*), Washington, D.C. (Nov. 2002).
4) Freedman, M.J., Sit, E., Cates, J. and Morris, R.: Tarzan: A Peer-to-Peer Anonymizing Network Layer, *Proc. 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*) (Mar. 2002).
5) Gnulella. http://gnutella.wego.com/
6) Guan, Y., Fu, X., Bettati, R. and Zhao, W.: A Quantitative Analysis of Anonymous Communications, *21st Annual Joint Conference of the IEEE Computer and Communications Societies* (*IEEE INFOCOMM 2002*) (June 2002).
7) Hazel, S. and Wiley, B.: Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer, *Proc. 1st International Workshop on Peer-to-Peer Systems* (*IPTPS '02*) (Mar. 2002).
8) Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B.: Oceanstore: An architecture for global-scale persistent storage, *Proc. ACM ASPLOS*, ACM (Nov. 2000).
9) National Technical Information Service (NIST). *Secure Hash Standard*, FIPS 180-1 (Apr. 1995).
10) Pfitzmann, A. and Waidner, M.: Networks without user observability—design options, *Eurocrypt '85*, LNCS 219, pp.245–253, Springer-Verlag, Berlin (1986).
11) Reiter, M.K. and Rubin, A.D.: Crowds: Anonymity for Web Transactions, *ACM Trans. Information and System Security*, Vol.1, No.1, pp.66–92 (1998).
12) Rowstron, A. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, Lecture Notes in Computer Science, Vol.2218 (2001).
13) Scalable Simulation Framework. http://www.ssfnet.org/
14) Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications, *Proc. 2001 ACM SIGCOMM Conference*, pp.149–160 (2001).
15) Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications, Technical Report TR-819, MIT (Mar. 2001).
16) Wright, M., Adler, M., Levine, B. and Shields, C.: Defending anonymous communication against passive logging attacks (2003).

付　　録

A.1 匿名性上の問題点に関する形式的な説明

O から P へリクエストが送信されるということは，P が O の finger table のどこかに入っているということである．仮に，O の x ($1 \leq x \leq m$) 番目の finger が P であったとする．その場合，$O + 2^{x-1} \leq P \bmod 2^m$ が成り立つ．また，識別子 k の successor node へのリク

エストが x 番目の finger P に転送されたということは，x+1 番目の finger は k よりも大きかったということであるので，$P<k<O+2^{(x+1)-1} \bmod 2^m$ が成り立つ．これらの関係より，$P<k<P+2^{x-1} \bmod 2^m \cdots (1)$ という関係が導かれる．同様に，P から D にリクエストが転送されるということは，$D < k < D+2^{m-1} \bmod 2^m \cdots (2)$ が成り立つ．O から P へ転送されたリクエストが再び P から D へ転送されるには，k は式 (1) および式 (2) を同時に満たす必要がある．そのためには，$D<P+2^{x-1} \bmod 2^m \cdots (3)$ でなければならない．D は P の m 番目の finger であるということ，すなわち $P+2^{m-1} \leq D \bmod 2^m$ を式 (3) に適用すると，$m<x \cdots (4)$ となる．しかし，前提条件より $1 \leq x \leq m$ であるので，式 (4) はつねに成り立たない．つまり，D へ P がいかなる O からのリクエストを転送することはありえないので，P から D へのリクエストはすべて P の自発であるということになる．

(平成 16 年 5 月 19 日受付)
(平成 16 年 7 月 22 日採録)

阿部　洋丈（正会員）

1976 年生．1999 年筑波大学第三学群情報学類卒業．2004 年筑波大学大学院博士課程工学研究科修了．2004 年より独立行政法人科学技術振興機構 CREST 研究員，現在に至る．博士（工学）．分散システム，コンピュータセキュリティに興味を持つ．日本ソフトウェア科学会，ACM，IEEE 各会員．

加藤　和彦（正会員）

1962 年生．1985 年筑波大学第三学群情報学類卒業，1987 年工学修士（筑波大学大学院工学研究科），1992 年博士（理学）（東京大学大学院理学系研究科）．1989 年東京大学理学部情報科学科助手，1993 年筑波大学電子・情報工学系講師，1996 年同助教授，現在に至る．オペレーティングシステム，プログラミング言語システム，データベースシステム，分散システム，モバイルオブジェクト計算に興味を持つ．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE 各会員．