構造型 P2P ネットワークにおける動的負荷分散法

武田敦志[†1] 生出拓馬[†2] 高橋晶子[†3]

現在までに様々な構造型 P2P ネットワークが提案されているが，従来の構造型 P2P ネットワークにはノードにかかる負荷を十分に分散することが難しいという問題があった．本稿では，ノードにかかる負荷を動的に分散する新しい構造型 P2P ネットワーク Waon を提案する．Waon では，各ノードの管理するコンテンツを動的に変更することにより，ノードにかかるコンテンツ管理負荷の動的な分散を実現する．また，Waon では，新たなルーティングテーブル作成アルゴリズムを用いることにより，通信メッセージの公平な分散を実現する．さらに，Waon を応用することにより，範囲検索や物理ネットワークの負荷軽減も可能である．本稿では，シミュレーションによる Waon の評価を行い，従来手法より効果的な負荷分散が可能であることを検証し，物理ネットワークの負荷軽減が可能であることを示す．

Dynamic Load Balancing for Structured P2P Network

Atushi Takeda,[†1] Takuma Oide[†2]
and Akiko Takahashi[†3]

There are a lot of proposals about structured P2P network, but it is difficult for existing structured P2P network to achieve dynamic load balancing enough. In this paper, we propose a new structured P2P network called *Waon*, which achieves dynamic load balancing among nodes. Each node in a Waon system can change objects stored in the node due to load balancing of objects. In addition, Waon uses a new routing algorithm due to load balancing of messages. Moreover, Waon can support range queries, and Waon can reduce the load on the physical network. In this paper, through simulation results, we confirm that Waon's load balancing is better than existing methods. And, a simulation result in this paper shows that Waon can reduce the load on the physical network.

1. はじめに

P2P ネットワークは，スケーラビリティや耐故障性に優れたオーバーレイネットワークであり，データやサービスなどのコンテンツの効率的な分散管理を可能とする．特に，構造型 P2P ネットワークはコンテンツ検索を確実に行うことができるため，Chord[1)]，Skip Graph[2)]，CAN[3)]，Pastry[4)]，Tapestry[5)] などの様々な構造型 P2P ネットワークが提案されている．これらの構造型 P2P ネットワークでは，分散ハッシュテーブルなどのアルゴリズムに基づいてコンテンツをノードに割り当てることにより，高いスケーラビリティと確実で効率的なコンテンツ検索を実現している．しかし，従来の構造型 P2P ネットワークでは，それぞれのノードが管理するコンテンツを静的に決定しているため，ノードにかかる負荷を十分に分散することが難しいという問題があった．

そこで，本稿では，それぞれのノードに割り当てられたコンテンツを動的に変更することにより，各ノードにかかる負荷を動的に分散する構造型 P2P ネットワーク Waon (Well-distribution Algorithm for Overlay Network) を提案する．Waon は，Chord[1)] と同様に，リング状の仮想的な ID 空間にノードとコンテンツを配置し，その ID 空間上のノード位置に基づいてオーバーレイネットワークを構築する構造型 P2P ネットワークである，しかし，Waon は，ID 空間上の各ノードの位置を動的に変更できる点と新しいルーティングテーブル作成アルゴリズムを導入している点で Chord と異なる．ID 空間上のノード位置が変更された場合，ノードに対するコンテンツの割り当ても変化する．そのため，ID 空間上のノード位置を制御することにより，ノードのコンテンツ管理負荷を制御することが可能である．Waon では，ノードの状況を考慮しつつ ID 空間上でのノードの位置を動的に変更することにより，それぞれのノードにかかるコンテンツ管理負荷を動的に分散させる．この一方で，Waon では，ID 空間の一部のエリアに多数のノードが集中することにより，一部のノードにオーバーレイネットワークの通信メッセージが集中するという問題がある．Waon では，この問題を解決するため，新たなルーティングテーブル作成アルゴリズムを導入する．このアルゴリズムで作成されたルーティングテーブルを用いてノード間の通信を行うことによ

†1 東北学院大学教養学部情報科学科
Department of Information Science, Tohoku Gakuin University
†2 仙台高等専門学校情報工学科
Department of Information Engineering, Sendai National College of Technology
†3 仙台高等専門学校情報システム工学科
Department of Information Systems, Sendai National College of Technology

り，ID 空間の一部のエリアに多数のノードが集中したとしても，オーバーレイネットワークの通信メッセージは各ノードに公平に分散する．

Waon では，ID 空間上のノード位置を動的に変更可能なため，コンテンツが ID 空間上に均一に分散する必要がない．そのため，名前の順序を保持したまま ID 空間上にコンテンツを配置することにより，コンテンツの効率的な範囲検索を行うことが可能となる．さらに，Waon では，ID 空間上のノード位置を物理ネットワークを考慮して決定することにより，物理ネットワークにかかる通信負荷を軽減することが可能となる．

本稿では，2 章で関連研究について述べ，3 章で提案手法 Waon について説明する．また，4 章ではシミュレーション評価を通じて Waon の有効性を検証し，5 章にて本稿をまとめる．

2. 関 連 研 究

サーバを必要としない純粋な P2P ネットワークは，非構造型 P2P ネットワークと構造型 P2P ネットワークに分類される．非構造型 P2P ネットワークは各ノードが自由に通信相手を設定できるという利点を持つが，コンテンツ検索の確実性が低いという問題やコンテンツ検索の効率が悪いという問題がある[6]．一方，構造型 P2P ネットワークはコンテンツ検索のためのネットワーク構造を持っており，確実で効率的なコンテンツの検索が可能である．

分散ハッシュテーブルは構造化 P2P ネットワークの代表的な手法であり，現在までに Chord[1]，CAN[3]，Pastry[4]，Tapestry[5] などが提案されている．分散ハッシュテーブルでは，SHA-1 などの一方向ハッシュ関数を用いてコンテンツとノードを仮想的な ID 空間上に配置し，この ID 空間上のノードとコンテンツの位置に基づいてオーバーレイネットワークを構築する．この手法は，コンテンツ検索の処理数が $O(\log n)$ (n は参加ノード数) となるスケーラブルな手法である．しかし，分散ハッシュテーブルでは，非線形関数であるハッシュ関数を用いてコンテンツを ID 空間上に配置するため，効率的な範囲検索が難しいという問題がある．また，分散ハッシュテーブルでは，ノードに対するコンテンツの割り当てを静的に決定するため，ノードにかかる負荷を動的に変更することが難しいという問題がある．

一方向ハッシュ関数を使用しない構造型 P2P ネットワークとして Skip Graph が提案されている[2]．Skip Graph は，各ノードが作成した Skip List[7] に基づいてオーバーレイネットワークを構築する手法であり，コンテンツ検索の処理数が $O(\log n)$ (n はコンテンツ数) となるスケーラブルな P2P ネットワークである．また，Skip Graph では，コンテンツの名前の順序を保持したままオーバーレイネットワークを構築できるため，コンテンツの効率的な範囲検索が可能である．しかし，この手法でも，ノードに対するコンテンツ割り当てを静的に決定するため，ノードにかかる負荷を動的に変更することは難しい．

ノードにかかる負荷を動的に変更する手法として，仮想サーバという概念を導入した動的負荷分散手法が提案されている[8]．仮想サーバとはコンテンツを管理する仮想的なノードであり，この仮想サーバを物理ノード上で実行することによりオーバーレイネットワークを構築する．この手法では，物理ノードと仮想サーバとの関係を動的に変更することにより，物理ノードにかかる負荷の動的な制御を実現している．しかし，この手法には，コンテンツを管理する P2P ネットワークだけではなく，物理ノードに対する仮想サーバの割りあてを決定するための情報共有ネットワークが必要となる．さらに，1 個の物理ノード上で動作する仮想サーバ数が増加すると，1 個の物理ノードが管理するルーティングテーブルが大きくなるという問題や各物理ノードが P2P ネットワークを保守するために必要となるメッセージ数が増加するという問題がある．

提案手法である Waon では，ノードのコンテンツ管理負荷を変更するために仮想サーバを使用しないため，動的負荷分散のための特別な情報共有ネットワークを必要としない．さらに，従来の仮想サーバを用いた動的負荷分散手法に比べて，各ノードのルーティングテーブルが小さいという利点や P2P ネットワークを保守するために必要になるメッセージ数が少ないという利点がある．

3. 提案: 構造化 P2P ネットワーク Waon

3.1 Waon の概要

Waon では，Chord[1] と同様に，仮想的なリング状の ID 空間にノードとコンテンツを配置し，ID 空間上でのノード位置とコンテンツ位置を基にノードに対するコンテンツの割り当てを決定する．ただし，Waon では，各ノードが ID 空間上の自身の位置を動的に変更することにより，そのノードが管理するコンテンツとコンテンツ管理負荷を動的に制御することができる．Waon のネットワークに参加している過負荷状態のノードは，自身のコンテンツ管理負荷を減すように ID 空間上を移動する．Waon では，全てのノードがこの動作を行うことにより，コンテンツ管理負荷を全てのノードに平等に分散させることを目指す．この一方で，Waon では，各ノードが ID 空間上での自身の位置を動的に変更可能なため，ID 空間の一部のエリアに多数のノードが集中する可能性がある．従来の構造型 P2P ネットワークのルーティングテーブル作成アルゴリズムは，ID 空間上のノード位置が一様に分布していることを想定している．そのため，従来手法を Waon に適用した場合，一部のノードに通信メッセージが集中する可能性がある．そこで，Waon では，ID 空間の一部のエリアに

多数のノードが集中したとしても通信メッセージを平等に分散させるため，新しいルーティングテーブル作成アルゴリズムを導入する．

Waon の特徴は，従来は静的に固定されていた ID 空間上のノード位置を動的に変更できる点である．この特徴により，コンテンツが ID 空間上に均一に分散する必要なくなるため，コンテンツの名前の順序を保持したまま ID 空間上に配置することが可能となる．これは，コンテンツの効率的な範囲検索が可能であることを意味する．また，ID 空間上のノード位置を柔軟に変更できるため，物理ネットワーク上でのノードの位置を考慮して ID 空間上にノードを配置することが可能となる．これは，物理ネットワークへの通信負荷を減らすことが可能であることを意味する．

3.2 Waon のネットワークモデル

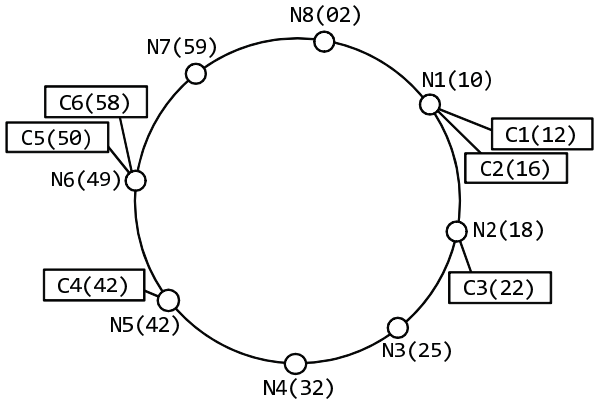
図 1(a) に Waon の ID 空間を示す．Waon の ID 空間は，0 以上 2^m 未満の数直線を円形にしたリング状の ID 空間である．Waon では，ノードとコンテンツを ID 空間上に配置し，この ID 空間上のノード位置に基づいてオーバーレイネットワークを構築する．ID 空間上のノード位置はノードが持つ属性値 *location* によって決定され，各ノードは自身の属性値 *location* を変更することにより ID 空間の任意の位置に移動できる．

Waon ではノードが持つ属性を以下のように定義する．

$$
\begin{aligned}
node &:= < location, successor, predecessor, route, content > \\
location &:= INTEGER \\
successor &:= \{node_0, node_1, \cdots, node_{r-1}\} \\
predecessor &:= \{node_0, node_1, \cdots, node_{r-1}\} \\
route &:= \{node_0, node_1, \cdots, node_{\log(n)-1}\} \\
content &:= \{content_0, content_1, \cdots\}
\end{aligned}
$$

ここで，r は各ノードが保持する隣接ノードリストの大きさを示し，n は Waon ネットワークに参加しているノードの数を示す．通常，r は 2 以上 $\log n$ 以下の値となる．この定義において，*successsor* は ID 空間における前方ノードの一覧であり，*predecessor* は ID 空間における後方ノードの一覧である．また，*route* はコンテンツを検索するときの中継ノード一覧であり，*content* は自身が管理しているコンテンツの一覧である．図 1(b) に，Waon に参加しているノードの属性値の例を示す．

Waon では，ID 空間におけるコンテンツの位置をコンテンツの識別子によって決定し，各ノードは ID 空間上で自身と前方ノードの間に位置するコンテンツを管理する．すなわち，ノードとコンテンツの関係は以下のように定義される．



(a) Identifier ring of Waon

```
node:N1
 N1.location = 10
 N1.successor = {N2, N3, N4}
 N1.predecessor = {N8, N7, N6}
 N1.route = {N2, N3, N5}
 N1.content = {C1, C2}
```

```
node:N2
 N2.location = 18
 N2.successor = {N3, N4, N5}
 N2.predecessor = {N1, N8, N7}
 N2.route = {N3, N4, N6}
 N2.content = {C3}
```

(b) Examples of nodes' properties

図 1 Waon で用いるリング状の ID 空間とノードの属性値の例
Fig. 1 Identifier ring of Waon and examples of node's properties

$$node.content[i].id \in [node.location, node.successor[0].location)$$

図 1 の例の場合，ノード N1 は，前方ノードであるノード N2 との間にあるコンテンツ C1 及び C2 を管理する．

3.3 ノード負荷の分散

Waon では，ID 空間上でのノードの位置を変更することにより，ノードのコンテンツ管理負荷を変更する．図 2 に，ノードの位置の変更手順の疑似コードを示す．Waon に参加するノードは，図 2 に示される関数 **updateLocation** を一定時間おきに実行する．まず，自身のリストに含まれる全ての *predecessor* に対して管理しているコンテンツ数を問い合わせ，*predecessor* が管理しているコンテンツ数の平均を算出する．次に，*predecessor* が管理しているコンテンツ数の平均と自身が管理しているコンテンツ数を比較し，自身の管理コンテン

```
// update location of a node n
n.updateLocation()
    sum = count(n.content);
    for(i = 0; i < r; i = i + 1)
        sum = sum + count(n.predecessor[i].content);
    ave = sum/(r + 1);
    num = count(n.content) − ave;
    if(num > 0)
        content = sortById(n.content);
        n.location = content[num].id;
        for(i = 0; i < num; i = i + 1)
            n.predecessor[0].addContent(content[i]);
            n.removeContent(content[i]);
```

図 2 ノード位置の更新手順の疑似コード
Fig. 2 Pseudo-code for updating node's location

ツ数が *predecessor* の平均より多い場合は，自身の ID 空間上の位置を *successor* 方向に移動する．この位置の変更により，ノードが担当する ID 空間上の領域が狭まり，ノードが管理するコンテンツの数が減少する．最後に，自身の管理から外れたコンテンツを *predecessor* に委譲する．Waon に参加する全てのノードが上記の動作を繰り返すことにより，コンテンツ管理負荷を全てのノードに平等に分散させることを目指す．

3.4 ルーティングテーブルの作成

Waon では，全てのノードが各自の判断で ID 空間上を移動するため，ID 空間の一部のエリアに多数のノードが集中する可能性がある．従来の分散ハッシュテーブルでは，ID 空間の一部のエリアに多数のノードが集中した場合，一部のノードに通信メッセージが集中するという問題があった．そこで，Waon では，この問題を解決するため，従来の分散ハッシュテーブルとは異なる新しいルーティングテーブル作成アルゴリズムを導入する．

Waon のノードが持つルーティングテーブルでは，$2^m (m = 0, 1, 2, \cdots)$ ホップ先の *successor* を中継先のノードとして記録する．すなわち，Waon のノードの属性値 *route* は以下のように定義される．

$$node.route[i+1] = node.route[i].route[i]$$

```
// update routing-table n
n.updateRoute()
    n.route[0] = n.successor[0];
    for(i = 0; ; i = i + 1)
        if(distance(n, n.route[i].route[i]) > distance(n, n.route[i]))
            n.route[i + 1] = n.route[i].route[i];
        else
            return;
```

図 3 ルーティングテーブル作成手順の疑似コード
Fig. 3 Pseudo-code for updating routing-table

図 3 にルーティングテーブルの作成手順の疑似コードを示す．Waon の全てのノードは属性値 *route* を更新するために図 3 の関数 **updateRoute** を一定時間おき実行する．これにより，ID 空間の一部のエリアに多数のノードが集中したとしても，通信メッセージを全てのノードに平等に分散する．また，このルーティングテーブルを用いてコンテンツ検索をするときに必要な処理数は $O(\log n)$ (n はノード数) となる．

4. 評　　価

4.1 シミュレーションの概要

提案手法の有効性を検証するため，Java で実装したシミュレータを用いて Waon の性能評価を行った．このシミュレーションでは，各ノードは 1 step 毎に以下の動作を行う．

(1) オーバーレイネットワークを保守するため，すべての *predecessor* と *successor* に対し *ping* メッセージを送り，すべての近接ノードの状態を確認する．
(2) 3.3 で述べた手順に従って，ID 空間上での自身の位置を変更し，ノードのコンテンツ管理負荷を分散する．
(3) 3.4 で述べた手順に従って，ルーティングテーブルを作成する．

各ノードは 10 step ごとにコンテンツの検索を行う．Waon が用いるリング状の ID 空間の大きさは 2^{64} とし，各ノードが保持する *predecessor* と *successor* の数はそれぞれ 7 とした．

図 4(a) にシミュレーションで使用したコンテンツ一部を示す．このシミュレーションでは，P2P ネットワークで UNIX のファイルシステムのデータを共有することを想定しており，P2P ネットワークで共有するコンテンツとして UNIX 系の OS である FreeBSD 8.0 に

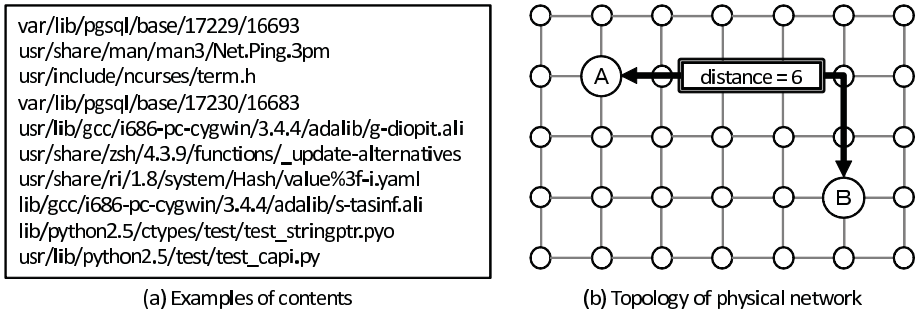


図 4 シミュレーションで想定したコンテンツと物理ネットワーク
Fig. 4 Contents and the physical network assumed in our simulation

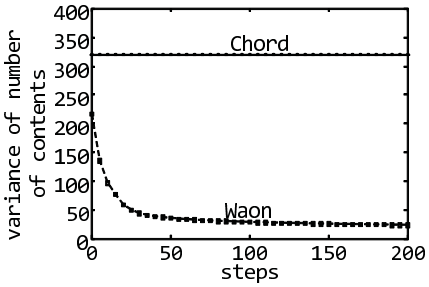


図 5 ノードが管理するコンテンツ数の分散値
Fig. 5 Variance of number of contents managed by a node

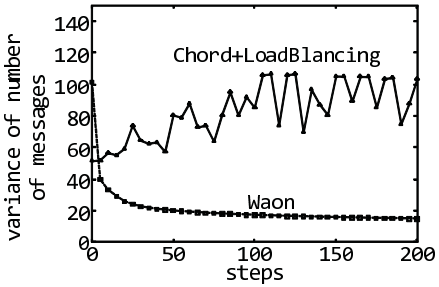


図 6 ノードが受信するメッセージ数の分散値
Fig. 6 Variance of number of message received by a node

含まれるファイルの一部を使用した．これらのコンテンツを検索する場合，ファイルパスを検索キーとして使用する．また，ID 空間上でのコンテンツの位置を決定するコンテンツ ID は，ファイルパスの先頭 32bit とファイルパスのハッシュ値の先頭 32bit を繋げたものとした．そのため，同じディレクトリに属するファイルコンテンツは，ID 空間上においても近い位置に配置される．シミュレーションでは，1 個のノードが Waon ネットワークに参加するごとに，10 個のコンテンツを登録することとした．また，各コンテンツの大きさは小さいものであり，1 個のコンテンツがノード間を移動するのに必要なメッセージ数は 1 とした．これは，コンテンツの内容が実データへのリンクである場合を想定している．

図 4(b) に，このシミュレーションで想定した物理ネットワークのトポロジを示す．物理ネットワークは格子状のネットワークとし，遅延やパケット損失は発生しないものとした．例えば，100 個のノードを対象としたシミュレーションを行う場合，物理ネットワークのトポロジは 10×10 の格子状であるとしてシミュレーションを行った．

上記の条件下でシミュレーションを行った．この章では，シミュレーション結果を通じ，最も有名な構造型 P2P ネットワークである Chord[1] と比較することで，提案手法である Waon の有効性を示す．

4.2 コンテンツ管理の負荷分散

図 5 に，参加ノード数が 1024 のとき，各ノードが管理するコンテンツ数の分散値を示す．Chord はノードが管理するコンテンツ数を変更できないため，管理コンテンツ数の分散値は一定である．これに対し，Waon は 1 step 毎にノードが管理するコンテンツ数を動的に変更し，各ノードが平等にコンテンツを管理するように動作するため，管理コンテンツ数の分散値は step の進行に従って減少する．この結果より，Waon ではコンテンツ管理負荷の動的な分散が実現されていることがわかる．

4.3 コンテンツ検索の通信メッセージ

図 6 に，参加ノード数が 1024 のとき，1 個のコンテンツを検索するときに必要になるメッセージ数の分散値を示す．ここで，Chord+LoadBalancing は，Chord にコンテンツ管理負荷の動的分散機能を追加したものである．これに対し，Waon には，動的負荷分散機能だけではなく，3.4 で提案した新しいルーティグテーブル作成アルゴリズムを実装している．図 6 の結果より，Waon の検索メッセージ数の分散値は，Chord+LoadBalancing の検索メッセージ数の分散値より小さいことがわかる．これは，Waon のルーティングテーブル作成アルゴリズムによって，コンテンツ検索の通信メッセージがすべてのノードに平等に分散されることを示している．

4.4 コンテンツ検索時のホップ数

図 7 に，コンテンツの検索に必要なホップ数の平均を示す．ここでも，Chord+LoadBalancing は，Chord にコンテンツ管理負荷の動的分散機能を追加したものである．図 7 の結果より，Waon の検索時のホップ数は，Chord+LoadBalancing のホップ数より小さいことがわかる．また，Waon 検索時のホップ数は $O(\log n)$ (n は参加ノード数) であり，Waon がスケーラブルな手法であることを示している．

4.5 物理ネットワークへの負荷

図 8 に，オーバーレイネットワークの保守を目的としたメッセージが物理ネットワークに

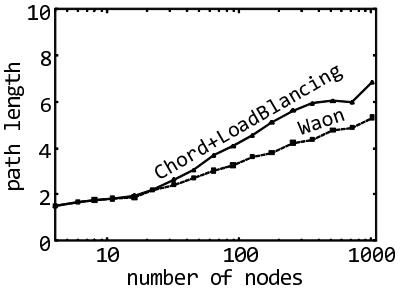


図 7 コンテンツの検索の平均ホップ数
Fig. 7 Average of path length for searching a content

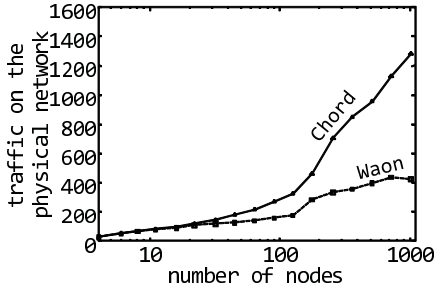


図 8 ネットワーク保守のための通信が物理ネットワークに与える負荷
Fig. 8 Traffic on the physical network for network maintenance

与える負荷を示す．ここでは，あるノードが物理ネットワーク上で距離 d だけ離れている別のノードへ m 個のメッセージを送信した場合，物理ネットワークへ与える負荷は $d \times m$ とした．例えば，図 4(b) のネットワークでノード A からノード B に 4 個のメッセージが送信された場合，物理ネットワークに与える負荷は 24 とした．また，ノードが Waon ネットワークに参加する場合，物理ネットワーク上で最も近いノードの *predecessor* として Waon の ID 空間に配置されるものとした．これにより，物理ネットワーク上のノード位置を ID 空間上のノード位置に反映させている．これに対し，Chord では，従来と同様に一方向ハッシュ関数から得られた ID を基にノードを ID 空間に配置した．

図 8 の結果より，Waon ネットワークの保守メッセージが物理ネットワークに与える負荷は，Chord ネットワークの保守メッセージが物理ネットワークに与える負荷より小さいことがわかる．Waon や Chord のネットワークの保守メッセージは，ID 空間上で近くに位置するノード間で頻繁に送受信される．Waon では物理ネットワーク上で近いノードを ID 空間上でも近くに配置することが可能なため，Waon が物理ネットワークに与える負荷は Chord が物理ネットワークに与える負荷より小さい．

5. おわりに

従来の構造型 P2P ネットワークは，それぞれのノードが管理するコンテンツを静的に決定しているため，ノードにかかる負荷を動的に分散することが難しいという問題があった．そこで，本稿では，それぞれのノードが管理するコンテンツを動的に変更することにより，各ノードにかかる負荷を動的に分散する構造型 P2P ネットワーク Waon を提案した．Waon は，ノードの状況を考慮しつつ ID 空間上でのノードの位置を動的に変更することにより，それぞれのノードにかかるコンテンツ管理負荷の動的な分散を実現する．また，新たなルーティングテーブル作成アルゴリズムを導入することにより，各ノードに対する通信メッセージの公平な分散を実現する．本稿では，シミュレーション結果を通じ，従来手法である Chord と比較することにより，Waon ではコンテンツ管理負荷やコンテンツ検索の通信メッセージが十分に分散されることや，物理ネットワークにかかる通信負荷を減らすことが可能であることを示した．

謝辞 本研究の一部は，文部科学省科学研究費補助金若手研究 (20700069) の助成を受けて実施したものである．

参 考 文 献

1) Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications, *IEEE/ACM Transactions on Networking*, Vol.11, No.1, pp.17–32 (2003).
2) Aspnes, J. and Shah, G.: Skip Graphs, *ACM Transactions on Algorithms*, Vol.3, No.4 (2007).
3) Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: A Scalable Content-Addressable Network, *Proceedings of ACM SIGCOMM*, pp.161–172 (2001).
4) Rowstron, A. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pp.329–350 (2001).
5) Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D. and Kubiatowicz, J.D.: Tapestry: A Resilient Global-scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, pp.41–53 (2004).
6) Clarke, I., Miller, S.G., Hong, T.W., Sandberg, O. and Wiley, B.: Protecting free expression online with Freenet, *IEEE Internet Computing*, Vol.6, No.1, pp.40–49 (2002).
7) Pugh, W.: Skip Lists: A Probabilistic Alternative to Balanced Trees, *Communications of the ACM*, Vol.33, No.6, pp.668–676 (1990).
8) Rao, A., Lakshminarayanan, K., Surana, S., Karp, R. and Stoica, I.: Load Balancing in Structured P2P Systems, *Lecture Notes in Computer Science*, Vol.2735, pp. 68–79 (2003).