Vol.2012-SE-175 No.20
2012/3/16

リング型オーバレイネットワークのルーティングアルゴリズムの提案

手嶋　達也[†1]　　為岡　未来[†2]　　島　和之[†1]

分散ソフトウェアシステムの障害許容性のため，次数が一定のオーバレイネットワーク上で経路長が対数となるルーティングアルゴリズムを提案する．提案するアルゴリズムは，多数のノードからなる分散システムにおいて，指定されたリソースを提供するノードを効率的に検索するために用いることが できる．多数のノードが相互にリソースを提供し，共有することによって障害許容性の高いシステムを実現できる．数学的評価によって，ノード数に対して次数の期待値は定数であり，経路長の期待値は対数であることが示された．シミュレーション評価によって，次数に対して経路長の逆数が対数となることが示された．すなわち，次数を大きくすることによって，検索時間を短くし，障害許容性を高めることが可能である．

Proposal of A Routing Algorithm for Ring Overlay Network

TATSUYA TESHIMA,[†1] MIKU TAMEOKA[†2]
and KAZUYUKI SHIMA[†1]

We propose a routing algorithm in a constant degree overlay network for resilience of distributed software systems. This algorithm can be used to efficiently lookup nodes that provide specified resources in distributed systems with many nodes. High resilient systems can be implemented by providing mutually and sharing resources of many nodes. A mathematical evaluation of the proposed algorithm shows the expected degree is constant not depending on the number of nodes, and the expected path length is logarithmic with the number of nodes. An evaluation by simulation shows the reciprocal of the path length is logarithmic with the degree. That is, it can make the time of lookup short and the resilience high by making the degree large.

†1 広島市立大学大学院情報科学研究科 Graduate School of Information Science, Hiroshima City Univercity
†2 広島市立大学情報科学部 Faculty of Information Science, Hiroshima City Univercity

1. 背　　景

近年，大量のストリーミングデータを扱うアプリケーションが増加している．これらのアプリケーションは大量のデータを扱うため，多くのネットワーク資源を消費する．現在，オーバレイネットワークを用いたマルチキャスト通信が注目を集めてきている．Chord[1)]，Kademlia[2)]，Pastry[3)] などはオーバレイネットワークの代表的なルーティングアルゴリズムであり，不特定多数のノードが随時参加または離脱するネットワークにおいて，管理用ノードを用いずにメッセージのルーティングが可能である．このため，耐障害性があり，スケーラブルな分散システムのソフトウェア開発に利用できる[6)]．例えば，DHT (Distributed Hash Table) を用いたファイル検索や ALM (Application Level Multicast)[4)] を用いたビデオ配信などの応用が挙げられる．オーバレイネットワークを用いた分散システムの品質向上のため，ルーティングアルゴリズムの性能が重要である．本研究では，ノード数に対して次数が一定で，経路長が対数となるルーティングアルゴリズムを提案する．従来手法である Chord ではノード数に対して次数は対数となる．そのため，ノード数が増えるに従って，ノードの参加や離脱時に隣接ノードを更新するための負荷が高くなる．提案手法では，次数が一定であることで，よりスケーラブルなシステムを構築出来る．また，従来手法の経路長は対数以上であるため，対数となる経路長は最適である．

2. Chord

DHT の中でもよく知られているものの一つとして Chord がある．ここでは，時計回りに識別子の値が増加していくリング状の識別子空間を考える．Chord の場合，識別子は SHA1 ハッシュ関数の値（160 ビットの数値）なので，識別子空間の大きさは 2^{160} である．参加ノードは各々が自律的にノード ID を決定する．SHA1 の値をリング状に並べて，時計回りで測った長さを距離とする．この距離の定義から，リング状に並べたノードの間に，近いノードと遠いノードが決まる．この近さは下位ネットワーク層とは無関係である．

図 1 にノードと探索対象ファイルを同じリング上にマップした図を示す（6 ビットの識別子空間）．図中の N はノード，K はキーを表す．各ノードのキーの担当範囲は自分の位置から前のノードの位置までの範囲である. 識別子間の距離の定義から，ファイルとノードにも近さの関係ができる．識別子空間上でファイルから時計回りに辿って一番近いノードが，そのファイルの情報を格納する．

表 1 に Chord のノードが持つ他ノードへの経路表を示す．ノードの参加や離脱があると，

Vol.2012-SE-175 No.20
2012/3/16

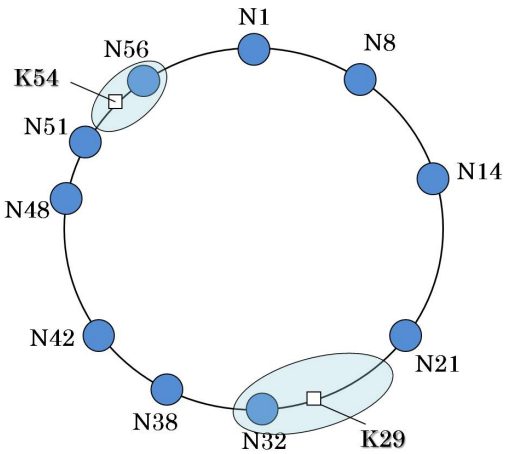

図 1 ノードとファイルをマップした Chord のリング（6 ビット空間）

当然経路表の情報を更新しなければならない．経路情報を正確に維持できないと，耐障害性の低いシステムになる．successor list が不正確になると Look Up が成立しなくなるので，この更新の優先順位は高い必要がある．一方 finger table は不正確でも Look Up の効率が落ちるだけで Look UP そのものは成立するので，更新の頻度は低くても構わない．ノードの参加と離脱への耐性を高める方法として，Chord では stabilize と呼ばれる作業をバックグラウンドで行っている．これは，各ノードが定期的にリング上の前後の複数ノードに対して successor list および predecessor list を問い合わせて，リング構造を維持する．

表 1 Chord のノードが持つ他ノードへの経路

名称	説明	Look Up での役割	維持コスト
successor list	識別子距離がもっとも近いノードへの経路	successor node に Look Up 命令を順々に伝えて Look Up 可能	低い
finger table	2^k ずつ遠い識別子距離のノードへの経路	finger table が全て正しければ，ノード総数 N に対して O(logN) で Look Up 可能	高い

3. リング型オーバレイネットワーク

概念的にキーとノードを円周上に配置する．キーの位置はハッシュ関数を用いて決める．ノードの位置は擬似乱数を用いて無作為に決める．ノードが複数のとき，ノードを境界として円周を複数の円弧に分割する．各円弧について，時計回りの始点にあるノードに，終点を除く円弧上にあるキーを割り当てる．ただし，ノードが 1 つのときは，そのノードにすべてのキーを割り当てる．円周上のキーまたはノードを数学的に表現するため，それらの位置を基点から時計回りの円弧の長さによって示す．基点とは円周上に定めた 1 点である．円周の長さは 1 とする．一般にキーは任意のデータであり，キーそのものとキーの位置とは異なるが，本論文では記述を簡潔にするため,「円周上の位置 k にあるキー」を「キー k」と略記する．また，同様に，ノードそのものとノードの位置とは異なるが,「円周上の位置 x にあるノード」を「ノード x」と略記する．次のように定義する．

$\mathbb{I} = \{x \in \mathbb{R} | 0 \leq x < 1\}$　はキーまたはノードの位置の区間である．
$\mathbb{R}$ は実数の集合である．

V　はノードの位置の有限集合である．$V \subset \mathbb{I}$.

$\lfloor x \rfloor = \max\{j \in \mathbb{Z} | j \leq x\}$　は $x \in \mathbb{R}$ 以下の最大の整数を返す関数（床関数）である．$\mathbb{Z}$ は整数の集合である．

$\{x\}_1 = x - \lfloor x \rfloor$　は $x \in \mathbb{R}$ である. 特に，$x \geq 0$ のとき，$\{x\}_1 = x - \lfloor x \rfloor$ は x の小数部分と等しい．

区域 $A(x, r) = \{y \in \mathbb{I} | \{y - x\}_1 < r\}$　：円周上を基点から時計回りに長さ $x \in \mathbb{R}$ 回った位置 $\{x\}_1$ から時計回りの長さが $r \in \mathbb{R}$ 未満のキーまたはノードの集合である．始点 $\{x\}_1$ は含む．$r < 1$ ならば，終点 $\{x + r\}_1$ は含まない．$r \geq 1$ ならば，円周上のすべての点を含む．

担当区域 $T(x) = A(x, |T(x)|)$　：ノードを境界として円周を円弧に分割したとき，ノード $x \in \mathbb{V}$ が時計回りの始点にある円弧であり，ノード x に割り当てるキーの集合である．

図 2 は，区域と担当区域を示す．区域 $A(k, r)$ は，キー k から $k + r$ まで時計回りの円弧上のキーの集合を示す．k は含まれるが，$k + r$ は含まない．$x_1, x_2, \cdots, x_5$ はノードを示す．担当区域 $T(x_3)$ は，ノード x_3 から x_4 まで時計回りの円弧上のキーの集合を示す．点 x_3 は含まれるが，点 x_4 は含まない．

提案するオーバレイネットワークを有向グラフとして表現する．この有向グラフの頂点はノードに対応し，頂点 x から頂点 y への有向辺は「ノード x がノード y のアドレスを知っ

Vol.2012-SE-175 No.20
2012/3/16

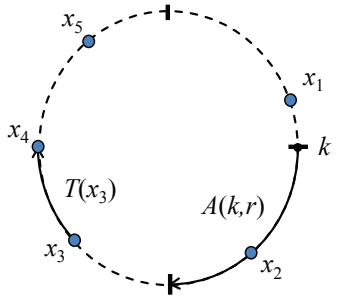

図 2 区域と担当区域

ていること」を意味する．次のように定義する．

前ノード $v_{-1}(x)$ ：キーまたはノード $x \in \mathbb{I}$ から反時計回りに最初のノードである．特に $V = \{x\}$ のとき，$v_{-1}(x) = x$ である．

後ノード $v_{+1}(x)$ ：キーまたはノード $x \in \mathbb{I}$ から時計回りに最初のノードである．特に $V = \{x\}$ のとき，$v_{+1}(x) = x$ である．

担当ノード $v_{\pm}(k)$ ：キー $k \in \mathbb{I}$ を割り当てられるノードである．ノードが k にない限り，k の前ノードが担当ノードである．しかし，ノードが k にある場合，そのノードが担当ノードである．そこで，k の後ノードの前ノードを担当ノードと定義する．$v_{\pm}(k) = v_{-1}(v_{+1}(k))$.

$E_{\pm}$ ：任意のノードからそれぞれの前ノードと後ノードへの有向辺の集合である．$E_{\pm} = \{(x, y) \in V \times V | y = v_{-1}(x) \vee y = v_{+1}(x)\}$.

親ノード ：あるノードに割り当てられたキーを β 倍したキーの担当ノードである．すなわち，ノード $x, y \in V$ が $^{\exists}k \in T(x) : \{\beta k\}_1 \in T(y)$ を満たすとき，ノード y を x の親ノードと呼ぶ．β は $\beta \geq 2$ の任意の整数である．ただし，すべてのノードにおいて同一であり，システムの実行中に変化しない定数とする．β を base と呼ぶ．

$V_*(\beta, x)$ ：ノード $x \in V$ の親ノードの集合である．
$V_*(\beta, x) = \{y \in V |^{\exists} k \in T(x) : \{\beta k\}_1 \in T(y)\}$.

$E_*(\beta)$ ：ノードからその親ノードへの有向辺の集合である．
$E_*(\beta) = \{(x, y) \in V \times V | y \in V_*(\beta, x)\}$.

図 3 は，前ノード，後ノード，担当ノードを示す．ノード x_1 の前ノードは x_5，後ノードは x_2 である．キー k の前ノードは x_1，後ノードは x_2 である．ノード x_2 の前ノードは x_1 である．よって，キー k の担当ノードは x_1 である．

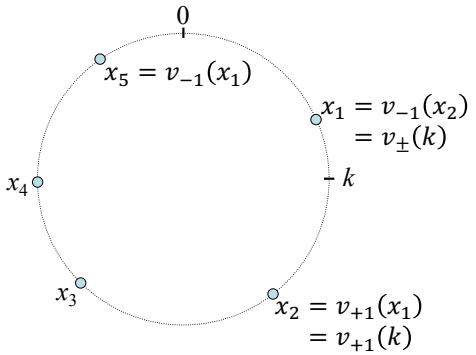

図 3 前ノード，後ノード，担当ノード

図 4 は，ノード $x_3 \in V$ の親ノードを示す．ノード x_3 の担当区域の長さを r とおくと，区域 $A(\beta x_3, \beta r)$ を担当区域に含むノード，すなわち，ノード x_1 と x_2 がノード x_3 の親ノードとなる．

提案するオーバレイネットワークは，2 つのネットワーク $(V, E_{\pm})$ と $(V, E_*(\beta))$ を組み合わせたネットワーク $G = (V, E_{\pm} \cup E_*(\beta))$ である．ネットワーク $(V, E_*(\beta))$ は，ノードが多い場合でも，キーの担当ノードを効率良く検索することを目的とする．ただし，このネットワークだけでは，ノードの参加や離脱におけるキーの割当てや隣接ノードの更新が難しい．ネットワーク $(V, E_{\pm})$ は，キーの割当ての更新とネットワーク $(V, E_*(\beta))$ の隣接ノードの更新を目的とする．また，ネットワーク $(V, E_{\pm})$ 自体の隣接ノードの更新は容易である．ただし，ネットワーク $(V, E_{\pm})$ だけでは，ノードが多い場合，キーの担当ノードを効率良く検索することができない．

図 5 は，親ノードの更新におけるノード x の状態遷移を示している．ノード x は，開始状態（黒丸）から親ノード検索送信状態 S5 と受信待ち時間状態 S2 へ並行して遷移する．親ノード検索送信状態 S5 では，親ノード検索メッセージ $M*$ を，時計回りに最初の親ノード $v_{\pm}(\beta x)$，あるいは，自分 x へ定期的に送信する．親ノード検索メッセージには，子ノード x_c とその後ノードのノード情報を記す．親ノード検索メッセージを x よりも $v_{\pm}(\beta x)$ へ送信する方が親ノードを早く検索できるので，なるべく $v_{\pm}(\beta x)$ へ送信する．しかし，$v_{\pm}(\beta x)$

Vol.2012-SE-175 No.20
2012/3/16

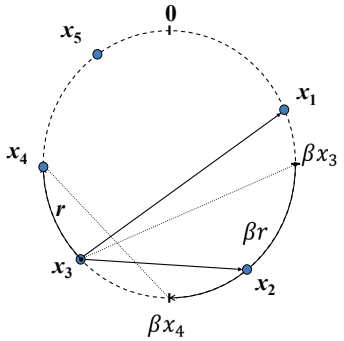

図 4 親ノード

がノード表にない，あるいは，離脱している場合，x へ送信する．受信待ち状態 S2 では，受信キューから取り出したメッセージによって遷移先を決定する．受信待ち状態 S2 において親ノード検索メッセージ $M*$ を取り出した場合，親ノード検索受信状態 S6 へ遷移する．親ノード受信状態 S6 において x が x_c の親ノードならば，x_c へノード通知メッセージ M_N を送信し，親ノード通知状態 S7 へ遷移する．ノード通知メッセージには，送信ノードとその後ノードのノード情報を記す．親ノード受信状態 S6 において x が x_c の親ノードでなければ，後ノード x_{+1} へ親ノード検索メッセージ $M*$ を転送し，受信待ち状態 S2 へ遷移する．親ノード通知状態 S7 において x_{+1} が x_c の親ノードならば，x_{+1} へ親ノード検索メッセージ $M*$ を転送し，受信待ち状態 S2 へ遷移する親ノード通知状態 S7 において x_{+1} が x_c の親ノードでなければ，受信待ち状態 S2 へ遷移する．受信待ち状態 S2 においてノード通知メッセージ M_N を取り出した場合，ノード通知受信状態 S8 へ遷移する．ノード通知受信状態 S8 では，ノード通知メッセージの送信ノードである親ノード y をノード表に追加し，親ノードとその後ノードの間のノードは離脱していると見なし，ノード表から削除し，受信待ち状態 S2 へ遷移する．親ノード検索メッセージを $v_{\pm}(\beta x)$ へ送信してから一定時間以上経過しても，$v_{\pm}(\beta x)$ からノード通知メッセージを受信しない場合，$v_{\pm}(\beta x)$ が離脱していると見なし，ノード表から削除する．

図 6 はノード x_2 の親ノードの更新におけるメッセージの流れを示している．

(1)　x_2 は親ノード検索メッセージを定期的に自分へ送信する．

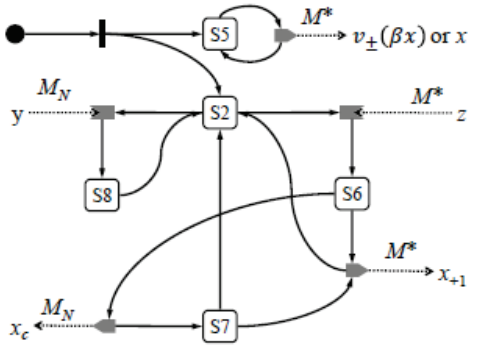

図 5 親ノードの更新におけるノード x の状態遷移

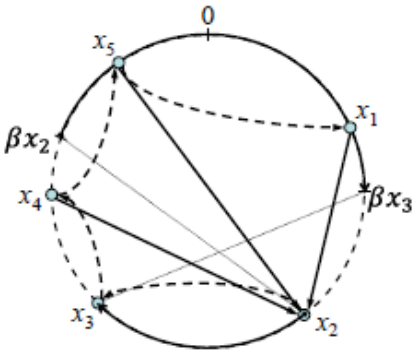

図 6 親ノードの更新におけるメッセージの流れ

(2) 親ノード検索メッセージを受信したノードは，自分が親ノードでなければ後ノードへ転送する（破線の矢印 $x_2 \to x_3 \to x_4$）.

(3) 親ノード検索メッセージを受信したノードは，自分が親ノードであればノード通知メッセージを子ノードへ送信する（実際の矢印 $x_4, x_5, x_1 \to x_2$）.

(4) 親ノード検索メッセージを受信したノードは，自分が親ノードであり，後ノードも親ノードであれば，親ノード検索メッセージを後ノードへ転送する（破線の矢印 $x_4 \to x_5 \to x_1$）.

4. ルーティングアルゴリズム

キーの担当ノードを検索するためにはネットワーク $(V, E_*(\beta))$ 上で担当ノード検索メッセージを担当ノードまで中継する．まず，担当ノードを検索するノードは，キーを指定して担当ノード検索メッセージを自分に送信する．担当ノードが担当ノード検索メッセージを受信すると，自分のアドレスを返信する．担当ノードではないノードが担当ノード検索メッセージを受信すると，親ノードの中から担当ノードまでの経路が短いノードを選び，メッセージを中継する．親ノードから担当ノードまでの経路の長さは，次の補題と定理より求まる．

補題 1 $^{\forall}x, r \in \mathbb{R}, ^{\forall}k \in A(\beta x, \beta r), ^{\exists}k' \in A(x, r) : k = \{\beta k'\}_1$.

証明: $k \in A(\beta x, \beta r)$ より，$\left\{k - \{\beta x\}_1\right\}_1 < \beta r$. $i = \lfloor \beta x \rfloor$，$j = \lfloor k - \{\beta x\}_1 \rfloor$ とおくと，$\left\{k - \{\beta x\}_1\right\}_1 = k - (\beta x - i) - j = k + i - j - \beta x$. $k' = (k + i - j)/\beta$ とおくと，$\left\{k - \{\beta x\}_1\right\}_1 = \beta k' - \beta x < \beta r$. $0 \le \left\{k - \{\beta x\}_1\right\}_1 < 1$ より，$0 \le k' - x < 1/\beta < 1$. よって，$\{k' - x\}_1 = k' - x < r$. このとき，$k' \in A(x, r)$，$k = \{\beta k'\}_1$. □

補題 2 $^{\forall}x \in V, ^{\forall}k \in A(\beta x, \beta |T(x)|) : v_{\pm}(k) \in V_*(\beta, x)$.

証明: 補題 1 より，$k = \{\beta k'\}_1$ を満たす $k' \in A(x, |T(x)|)$ が存在する．$k' \in T(x)$ は $k = \{\beta k'\}_1 \in T(v_{\pm}(k))$ を満たすので，親ノードの定義より，ノード $v_{\pm}(k)$ はノード x の親ノードである． □

定理 1 ノード $x \in V$ とキー $k \in \mathbb{I}$ について $^{\exists}L \in \mathbb{Z}, L \ge 0 : k \in A(\beta^L x, \beta^L |T(x)|)$ ならば，ノード x からキー k の担当ノード $v_{\pm}(k)$ までの長さが L 以内の経路が存在する．

証明: 以下の数学的帰納法により，定理 1 が成立する．

(1) $L = 0$ のとき,$k \in A(x, |T(x)|)$ より，ノード x がキー k の担当ノードであり，経路の長さは 0 であるのでこの定理が成り立つ．

(2) $L > 0$ のとき，$k' \in A(\beta^{L-1} x, \beta^{L-1} |T(x)|)$ ならば，ノード x からキー k' の担当ノード $v_{\pm}(k')$ までの長さが $L - 1$ 以内の経路が存在すると仮定する．補題 2 より，$k \in A(\beta^L x, \beta^L |T(x)|)$ ならば，ノード $v_{\pm}(k)$ はノード $v_{\pm}(k')$ の親ノードである．よって，ノード x から $v_{\pm}(k)$ までの長さが L 以内の経路が存在する． □

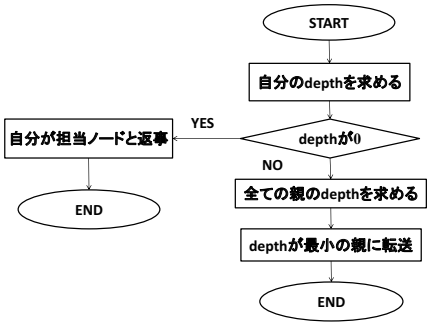

図 7 ルーティングアルゴリズム

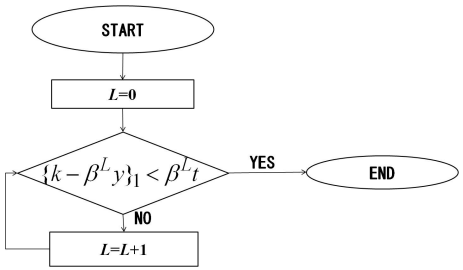

図 8 depth の求め方

定理 1 より，$x \in V$ は，その親ノード $y \in V_*(\beta, x)$ から担当ノードまでの経路の長さ

$$L_*(\beta, y, k) = \min\{L \in \mathbb{Z} | L \ge 0, k \in A(\beta^L y, \beta^L |T(y)|)\}$$

を求め，経路の長さが最小となる親ノードへメッセージを中継する．

図 7 は，提案手法のルーティングアルゴリズムを示している．メッセージを受信した各ノードは以下の手順で経路を決定する．自ノードの depth を求める．depth が 0 ならば，自ノードが root であると送信元に返事を送る．そうでなければ，全ての親の depth を求める．depth が最小の親にメッセージを転送する．

図 8 は，depth の求め方を示している．L の初期値を 0 とする．条件 $\left\{k - \beta^L y\right\}_1 < \beta^L t$

を満たすまで，L の値を 1 つずつ増やす．この条件を満たしたときの L を depth とする．ここで，k はキー，β は定数，y はノード，t はノード y の担当範囲の長さである．$\{x\}_1$ は x の小数部であり，$\{x\}_1 = x - \lfloor x \rfloor$ と定義される．

図 9 は，$\beta = 2$ においてノード 8/64 がキー 54/64 の担当ノードを検索するときのメッセージの経路を示す．ノード 8/64 の担当区域は $T(8/64) = A(8/64, 14/64 - 8/64)$．区域 $A(2 \times 8/64, 2 \times 6/64) = A(16/64, 12/64)$ を担当区域に含むノード 14/64，21/64 がノード 8/64 の親ノードである．$|T(14/64)| = 21/64 - 14/64 = 7/64$ より，

$$
\begin{aligned}
&54/64 \notin A\,(14/64, 7/64)\\
&54/64 \notin A\,(2 \times 14/64, 2 \times 7/64)\\
&\qquad = A\,(28/64, 14/64)\\
&54/64 \notin A\left(2^2 \times 14/64, 2^2 \times 7/64\right)\\
&\qquad = A\,(56/64, 28/64)\\
&54/64 \in A\left(2^3 \times 14/64, 2^3 \times 7/64\right)\\
&\qquad = A\,(48/64, 56/64)
\end{aligned}
$$

$|T(21/64)| = 32/64 - 21/64 = 11/64$ より，

$$
\begin{aligned}
&54/64 \notin A\,(21/64, 11/64)\\
&54/64 \in A\,(2 \times 21/64, 2 \times 11/64)\\
&\qquad = A\,(42/64, 22/64)
\end{aligned}
$$

$L_*(2, 14/64, 54/64) = 3$，$L_*(2, 21/64, 54/64) = 1$ より，ノード 8/64 は，担当ノードまでの経路が短いノード 21/64 を選び，検索メッセージを送信する．検索メッセージを受信した 21/64 も同様にして，ノード 51/64 を選び，検索メッセージを送信する．ノード 51/64 は，キー 54/64 の担当ノードであるので，検索元 8/64 へ返信する．

5. シミュレーション

ノード数に対する経路長および，次数に対する経路長の関係を明らかにする．

使用した計算機のスペックは以下のとおりである．

- CPU　Intel(R) Core(TM)2 Duo CPU
- 実装メモリ　4.00GB(2.87GB 使用可能)
- OS　Microsoft Windows 7
- コンパイラ　gcc 4.3.4

今回のシミュレーションは以下のような条件で行った．

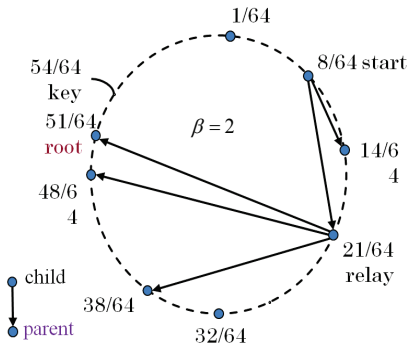

図 9　メッセージ経路

- ノード数　32, 64, 128, 256, 512
- 送信パケット　10000
- base　$\beta = 2, 4, 8, 16, 32$

通信の遅延時間などをシミュレーションに反映させるため，離散イベントシミュレーション方式を用いた[5]．提案手法を記述するための言語としては，最も普及しているプログラミング言語の 1 つである C 言語を用いた．スケーラビリティを評価するため，ノード数が 32, 64, 128, 256, 512 のときの経路長を計測した．次数は 6 とした．

遅延時間を計測するため，メッセージの送信時刻と受信時刻を記録する．メッセージ到達率を計測するため，メッセージ送信回数と受信回数を記録する．また，誤差 1 %以内でメッセージ到達率を求めるため，送信パケット数は 10000 とする．

表 2 はログの例を示している．各行はイベントの発生時刻を示している．イベントの型が ID となっている行は，ノード ID の設定を示している．イベントの型が pred となっている行は，predecessor の更新を示している．イベントの型が succ となっている行は，successor の更新を示している．イベントの型が deliver となっている行は，メッセージの到着を示している．ここで，TraceRoute の第 1 引数がホップ数を示している．イベントの型が Exit となっている行は，シミュレーションの終了を示している．

表 2 ログの例

時刻	ノード	イベントの型	パラメータ
0.1	Log Port:-14	ID	0.51531
0.1	Log Port:-15	ID	0.97958
	⋮		
0.1	Log Port:-13	ID	0.37902
5.2	Log Port:-13	pred	NULL -> 0.51531@Port:-14
5.2	Log Port:-1	pred	NULL -> 0.24499@Port:-3
	⋮		
5.2	Log Port:-11	pred	0.07684@Port:-12 -> 0.51531@Port:-14
5.3	Log Port:-9	succ	NULL -> 0.89158@Port:-8
5.3	Log Port:-14	succ	NULL -> 0.07684@Port:-12
	⋮		
17.3	Log Port:-14	succ	0.61071@Port:-16 -> 0.52237@Port:-1
3605.1	Log Port:-16	deliver	TraceRoute 0 2443 Port:-16
3605.1	Log Port:-3	deliver	TraceRoute 0 8124 Port:-3
	⋮		
3605.8	Log Port:-7	deliver	TraceRoute 7 2614 Port:-7
3665	Log Port:-12	Exit	

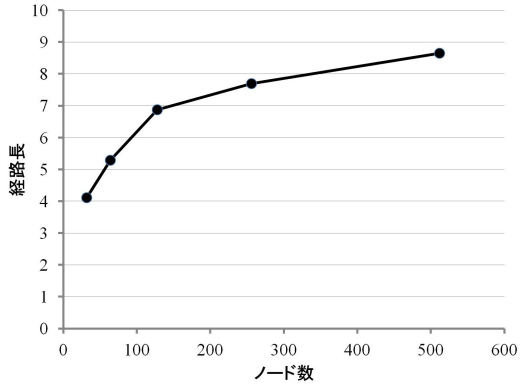

図 10 ノード数と経路長

図 10 に，ノード数と経路長の関係を示す．このグラフより，ノード数に対して，経路長が対数となっていることが分かる．

図 11 に base を変えた場合のノード数と経路長の関係を示す．base を変えても経路長に大きな差がなかった．この要因については，successor リストを 16 に固定していたため，親ノードが successor の範囲内にない場合があるからと考えられる．

図 12 は base と successor リストのサイズを等しくした場合のノード数と経路長の関係を示す．base と successor リストのサイズを大きくすると，経路長が短くなることがわかる．この要因としては，base が大きくなるほど担当ノードまでに通る親ノードの数が少なくなること，および，successor リストのサイズが十分に大きく親ノードが含まれる可能性が高かったからと考えられる．

図 13 に経路長の逆数をとったものを示す．経路長の逆数は次数に対して対数となっている．この理由として，ノード数に対して経路長は対数となるが，その底が次数であるからと考えられる．

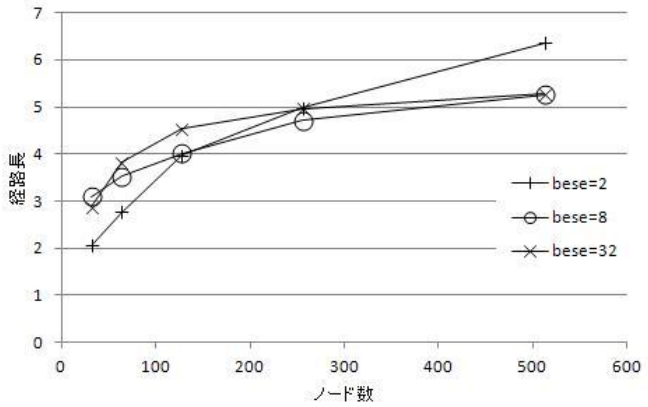

図 11 ノード数と経路長

Vol.2012-SE-175 No.20
2012/3/16

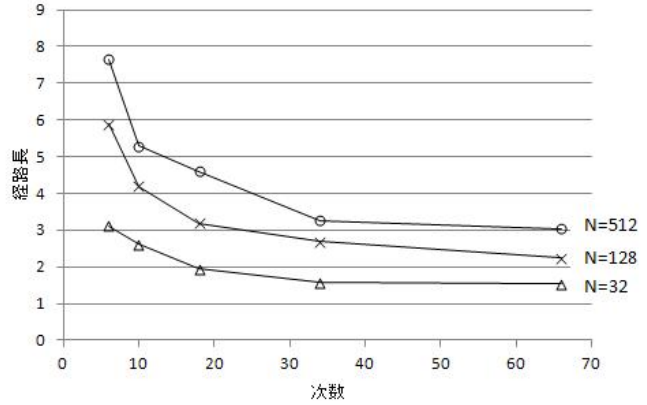

図 12 次数と経路長

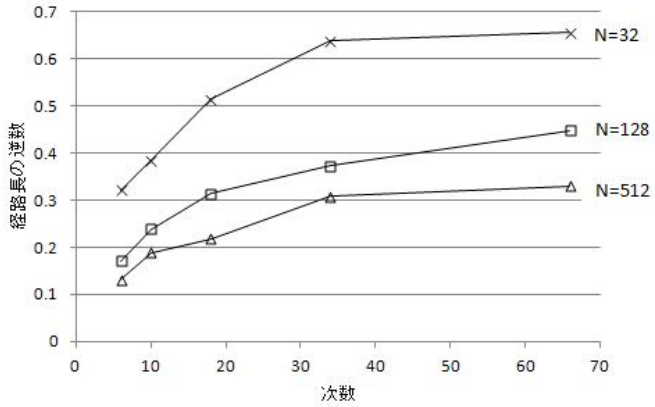

図 13 次数と経路長の逆数

6. 結　　論

本研究では，オーバレイネットワークを用いた分散システムの品質向上のため，ノード数に対して次数が一定で経路長が対数となるルーティングアルゴリズムを提案した．提案手法のシミュレーションにより，ノード数が 8 から 1024，次数が 6 のとき，経路長が対数となることを示した．ノード数が 32 から 512 までのとき，次数を大きくすると経路長が短くなることを示した．今後の課題としては，従来手法との次数，経路長，遅延時間，メッセージ到達率の比較が挙げられる．

参 考 文 献

1) Ion Stoica et al: Chord: A scalable peer-to-peer lookup service for Internet applications, ACM SIGCOMM 2001, San Diego, CA, pp. 149-160, (Aug. 2001)
2) Petar Maymounkov et al: Kademlia: A peer-to-peer information system based on the xor metric, IPTPS ’02, (Mar. 2002)
3) Antony Rowstron et al: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, the 18th IFIP International Conference on Distributed Systems Platforms, (Nov. 2001)
4) 伊藤和也 ：オーバレイネットワーク上の ALM における遅延時間の PlanetLab における評価， 広島市立大学 平成 21 年度卒業論文, (2010 年 2 月)
5) 手嶋達也 ：離散イベントシミュレーションを用いた Chord シミュレータの開発， 広島市立大学 平成 22 年度卒業論文, (2011 年 2 月)
6) Hyojin Kwon, Kazuyuki Shima, Yasuomi Sato, and Misuru Ohba : A simple routing algorithm in a constant degree overlay network, International Symposium on Software Reliability Engineering, Hiroshima, (Dec. 2011)