社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＭＢＬ－29　(19)
2004／5／14

Chord による LIN6 node の位置解決手法

田中　康之[†]　土井　裕介[††]　國司　光宣[†]　寺岡　文男[†,†††]

† 慶應義塾大学大学院理工学研究科
†† 東芝研究開発センター通信プラットホームラボラトリ
††† ソニーコンピュータサイエンス研究所
E-mail: †{yatch,kunishi}@tera.ics.keio.ac.jp, ††ydoi@isl.rdc.toshiba.co.jp, †††tera@ics.keio.ac.jp

あらまし　本論文では，モビリティサポートプロトコルである LIN6 における移動ノードの位置情報を，Chord に準じるアルゴリズムにより構成されたオーバーレイネットワークで管理するための手法を提案する．本稿では，このようなオーバーレイネットワークを Chord ネットワークと呼ぶ．従来の LIN6 では，移動ノードの位置情報を Mapping Agent と呼ばれる位置管理サーバで管理しており，移動ノードと Mapping Agent の関係は静的に決められている．この静的な関係は，ノードの識別子である LIN6 ID に対する Mapping Agent の IP アドレスという形で，DNS によって管理される．しかし，非構造な名前である LIN6 ID の管理に木構造の名前空間を前提とする DNS を用いることは不適切であると考えられ，LIN6 ID と Mapping Agent の静的な対応を DNS を用いて管理するのは運用上の規模拡張性に欠けることが懸念される．そこで，本稿では Chord ネットワーク を利用した LIN6 ノードの位置解決手法を提案する．提案方式は DNS に依存せず，また，規模拡張性を持った方式となっている．
キーワード　オーバーレイネットワーク，モビリティ，IPv6

Resolving LIN6 node's locator using Chord

Yasuyuki TANAKA[†], Yuusuke DOI[††], Mitsunobu KUNISHI[†], and Fumio TERAOKA[†,†††]

† Faculty of Science and Technology, Keio University
†† Communication Platform Laboratory, R&D Center, Toshiba Corporation
††† Sony Computer Science Laboratories Inc.
E-mail: †{yatch,kunishi}@tera.ics.keio.ac.jp, ††ydoi@isl.rdc.toshiba.co.jp, †††tera@ics.keio.ac.jp

Abstract　LIN6 use Mapping Agent to maintain LIN6 nodes' mappings. Mapping is association between LIN6 node's ID and locator. A LIN6 node updates its locator on the Mapping Agent which are designated to store the locator. A network administrator of a LIN6 node decides the designated Mapping Agent of the node, and the administrator registers this relationship to DNS. But we argue that it is inappropriate to maintain the relation by DNS from the aspect of operational scalability. Then, this paper describes a new method to maintain and resolve LIN6 node's locator instead of using DNS and Mapping Agent.
Key words　Overlay Network, Mobility, IPv6

1. はじめに

近年，PDA (Personal Digital Assistants) の普及や無線 LAN を利用した通信が一般化してきたことなどにより，端末が移動しながらインターネットに接続するという形態が増えて来ている．そのような通信では，端末の移動によらず通信を継続したい，という要求があり，そのためのプロトコルとして移動透過性保証プロトコルが提案されている．

LIN6 [1], [2] はネットワーク層の移動透過保証プロトコルの１つであり，ノードの識別子と位置指示子という概念をトランスポート層以上では分離して扱う，という特徴がある．ノードの識別子は LIN6 ID と呼ばれる非構造な 64 bit の識別子であり，また，位置指示子はノードのネットワークインタフェースが持つグローバル IPv6 アドレスのネットワークプレフィックスとなる．この LIN6 では，移動ノードの位置情報は Mapping Agent と呼ばれる位置管理サーバで管理され，各移動ノードは自身の位置情報を Mapping Agent に登録する．Mapping Agent はインターネット上に複数存在するが，各移動ノードが実際に登録を行う Mapping Agent は静的に決められており，その対応は，LIN6 ID に対する Mapping Agent の IP アドレスとして

DNS で管理される．よって，あるノードが移動ノードの位置情報を取得する場合には，はじめに，DNS によって移動ノードの LIN6 ID に対応する Mapping Agent の IP アドレスを取得し，その次に，DNS から得られた Mapping Agent に対して移動ノードの位置情報を要求することになる．しかし，このように LIN6 ID に対応する Mapping Agent の静的な関係を DNS を利用して管理すると，管理すべき LIN6 ID の総数が増えた場合に，この対応を管理している DNS サーバの運用上のコストが大きくなってしまうと考えられる．これは LIN6 ID が非構造なため，管理すべき LIN6 ID の領域を複数の DNS サーバで分割管理するということが難しいことによる．よって，このような移動ノードと Mapping Agent の関係を DNS を利用して管理することは適切ではないと考えられる．

そこで本論文では，Chord を利用して LIN6 ID に対する情報を管理する手法を提案する．この提案手法では DNS に依存せず，従来の LIN6 よりも運用コストが低いことがわかった．

2. LIN6

LIN6 [1], [2] とは IPv6 ネットワーク上での移動透過性を保証するプロトコルである．ここで移動透過性とは

- ノードの移動によらず，通信が継続できる性質
- ノードの位置によらず，ノード識別子が変化しない性質

を指す．

従来のネットワークアーキテクチャでは，ネットワークアドレスが位置に関する情報とノード自体を識別する情報を分離することなく保持しており，LIN6 ではこのことを問題として提起している．この問題はネットワークアドレスの二重性と呼ばれ，LIN6 ではこの問題を解決するため，トランスポート層以上では，ネットワークアドレスが保持している位置指示子とノード識別子という 2 つの情報を概念的に分離している．位置指示子とノード識別子の概念を分離することにより，ネットワーク層より上位層では，ノード識別子を用いて位置に依存しないコネクションを確立し，ネットワーク層では位置指示子を用いて経路制御を行うことで移動透過性を保証する．LIN6 では，このノード識別子を LIN6 ID と呼び，64 bit のグローバルユニークな識別子として定義している．また，ノードが持つグローバル IPv6 アドレスのネットワークプレフィックスを locator と呼び，位置指示子として定義している．以下，LIN6 ID を持つノードを LIN6 ノードと呼ぶ．

2.1 従来の LIN6 での位置解決手法

LIN6 ID に対応する locator の情報を mapping と呼び，この mapping は Mapping Agent (MA) と呼ばれる位置管理エージェントで管理される．この MA はインターネット上に複数存在し，どの MA がどの LIN6 ID の mapping を管理するか，という関係は静的に決められている．各 LIN6 ID には 1 つ以上の MA が対応づけされており，この MA と LIN6 ID のこの静的な関係は DNS によって管理される．ある LIN6 ID に対応する MA の IP アドレスを取得する場合は，その LIN6 ID をキーとして DNS に問い合わせればよい．各 LIN6 ノードは自分の LIN6 ID に対して静的に決まっている MA へ定期的に

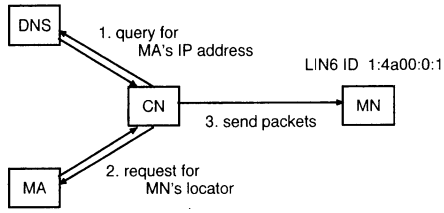

図 1 CN の通信手順

locator 情報を登録し，MA が持つ mapping を最新の状態に維持する．また，LIN6 ノードがネットワークを移動した際にも，その MA に対して locator を登録する．

そこで従来の LIN6 における LIN6 ノードの位置解決は次のように行われる．ここで，あるノード (CN と呼ぶ) が LIN6 ID 1:4a00:0:1 を持つ LIN6 ノード (MN) に対して通信を開始する場合を考える．1:4a00:0:1 に対応する MA には，MN の locator がすでに登録されているものとする．このとき，図 1 のような手順で CN は MN へパケットを送信する．

（1） DNS から 1:4a00:0:1 に対応する MA の IP アドレスを取得

（2） MA から 1:4a00:0:1 の現在の locator を取得

（3） 1:4a00:0:1 を LIN6 ID として持つ MN に対してパケットを送信

2.2 従来の位置解決手法における問題点

前節で説明したように，従来の LIN6 における位置解決手法では DNS に依存しており，DNS サーバでは LIN6 ID に対する MA の IP アドレスが静的に設定されている．しかし，LIN6 ID に対応する MA の静的な関係を DNS サーバに対して設定する場合，管理すべき LIN6 ID の数が増大するにつれて，DNS サーバの負荷や管理コストが高くなるという問題がある．管理すべき LIN6 ID は最大で 2^{40} 個であることを考えると，LIN6 ID に対応する情報を静的に DNS で管理することは非現実的であると言える．DNS サーバの負荷や管理コストを軽減するために，複数の DNS サーバで LIN6 ID と MA の IP アドレスの関係を分散管理することも可能であるが，そのためには LIN6 ID に構造を持たせる必要がある．しかし，LIN6 では LIN6 ID の構造化について定義されておらず，このままでは LIN6 ID に対応する情報を複数の DNS サーバで分散管理できない．

そこで本論文では，LIN6 ID に対応する情報を DNS に依存しない形で管理する方法を提案する．提案方式では LIN6 ID を非構造な識別子として扱い，Chord を利用して，LIN6 ID に対応する情報の管理を複数のノードで分散管理することが可能である．3. 章では，提案方式で使用されている，Chord について説明をする．

3. Chord

peer-to-peer アプリケーションでは，データを保持するノードをどのように効率よく決定するか，ということが問題となる．Chord [3] はこの問題を解決するためのアルゴリズムである．以

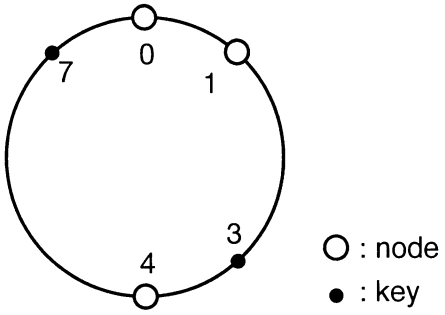

図 2 ID 空間の例 (2^3 の ID 空間の場合)

下，キーに対するデータの取得のことを参照と呼ぶ．

Chord によって形成されたオーバーレイネットワーク (以下，Chord ネットワークと呼ぶ) では，データをどのノードに保持させるかの決定にハッシュ関数を使用し，Chord ネットワークを構成するノード間の負荷分散を実現している．よって，この Chord では，データの保管が Chord ネットワークを構成するノードで分散的に行われ，1 箇所のノードにデータが集中するということは起こりにくい．また，Chord は任意の構造を持つキーを扱うことができ，非構造な識別子を扱うこともできる．

では，以下で，この Chord ネットワークの動作の概要について説明をする．

3.1 キー ID とノード ID の割り当て法

Chord ネットワークでは，データに対するキー ID と Chord ネットワークを構成するノード ID は SHA-1 のハッシュ関数を用いて作成され，同じ ID 空間上に配置される．ここで，キー ID はデータに対するキーワードのハッシュ値であり，ノード ID はそのノードの IP アドレスのハッシュ値となる．各 ID が配置される ID 空間はリング状になっており，ノード ID とキー ID はその ID 空間上に分散した状態で配置される．図 2 は 3 bit の ID 空間の例である．図 2 ではノード ID 0，1，4 の各ノードと，キー ID が 3，7 の各キーが 3 bit の ID 空間に配置されている様子を示している．

3.2 キー値のマッピング

Chord ネットワークでは各ノードが図 2 のような仮想的なリングを形成し，時計方向に接続を行っている．各キー値とデータの対は，仮想的なリング内で時計周りに最も近いノードに格納される．図 2 ではキー ID 3 に対応するデータはノード ID 4 を持つノードに格納されることになる．

3.3 キーに対するデータの参照方法

Chord では参照を高速にするため，各ノードが finger table と呼ばれるルーティングテーブルを持つ．finger table は，ID 空間が m bit であった場合，自ノードの ID より $2^k(0 \leq k < m)$ 先の ID の IP アドレスを保持するテーブルである．もし自ノードの ID より $2^k(0 \leq k < m)$ 先の ID にノードが存在しない場合は，その ID から時計周りに一番近いノードの IP アドレスが finger table に格納される．図 2 の例では，各ノードは $2^k(0 \leq k < 3)$ の finger table を管理している．表 1 に図 2 のノード ID 0 における figer table を示す．表 1 では対応するノードの IP アドレスの代わりに，ノードの ID を使用している．また表 1 内の「ID の範囲」は，対応するデータの参照時に参照されるものである．[x,y) は $x \leq i < y$ の範囲を表し，finger table では $[2^k, 2^{k+1})$ として計算される．

表 1 finger table の例 (ノード ID 0)

k	自ノード $ID + 2^k$	対応するノード	ID の範囲
0	1 $(= 0 + 2^0)$	1	[0,1)
1	2 $(= 0 + 2^1)$	4	[2,4)
2	4 $(= 0 + 2^2)$	4	[4,0)

キーに対応するデータを格納しているノードを発見する場合はこの finger table を用いて参照を行う．例えば，ノード ID 0 のノードがキー ID 3 に対応するデータを取得したい場合，finger table を参照し，キー ID 3 が含まれる ID の範囲に対応するノード ID に対してデータのクエリを送信する．例ではクエリをノード 4 に送信することになる．また，ノード ID 0 のノードがキー ID 7 に対応するデータを取得したい場合は，finger table を参照し，ノード ID 4 に対してクエリを送信する．ノード ID 4 のノードではキー ID 7 のデータを保持していないため，自身の finger table を参照して受信したクエリの転送を行う．そして，最終的にそのデータを保持するノードにクエリが到達する．このような参照を行うことで，最悪でもクエリの転送を $O(logN)$ の回数行えば目的のデータを得ることができる．

前述の説明では Chord ネットワークを構成するノードによるデータの参照を説明したが，Chord ネットワークに参加していないノード (クライアントと呼ぶ) によるデータの参照も可能である．その場合，データを要求するクライアントは参照のためのクエリを Chord ネットワークを構成するノードに送信する．クエリを受信したノードはクエリを元に，データの参照を前述の方法で行う．そして，参照によって得られたデータはそのノードからクライアントに送信され，クライアントは要求したデータを受信することができる．

3.4 ま と め

以上が Chord ネットワークの概要である．その他，ノードが Chord ネットワークに参加する場合や，ネットワークから離脱する場合の詳しい処理内容や Chord ネットワークの性能評価などは文献 [3] を参照されたい．

4. 提案方式

提案方式では，3. 章で説明した Chord ネットワークを利用し，DNS に依存しない形で，LIN6 ノードの locator を管理する．このように，Chord を利用して LIN6 ノードの locator を管理する場合，次の 2 つの方法が考えられる．

a) Chord で LIN6 ID と MA の IP アドレスの対応を管理する方法

この方法では，LIN6 ID と MA の IP アドレスの対応を Chord ネットワークで管理する．よって，あるノードが LIN6

ノードの locator を取得する場合，そのノードは，はじめに Chord ネットワークから LIN6 ID に対応する MA の IP アドレスを取得し，その後，得られた MA に対して LIN6 ノードの locator を問い合わせる．従来の方式では DNS を信用し，DNS から得られた MA の IP アドレスが正しいかどうかの検証を行わないが，(a) のように Chord から MA の IP アドレスを取得する場合には得られた IP アドレスの検証が必要となる．これは攻撃者が偽の MA の IP アドレスを Chord へ登録している状況や，攻撃者が Chord ネットワークに参加している状況に対処するためである．

b） Chord で LIN6 ID と locator の対応を管理する方法

この方法では LIN6 ID と locator の対応を Chord ネットワークで管理する．あるノードが LIN6 ノードの locator を取得する場合，直接 Chord ネットワークに対して LIN6 ID に対応する locator を問い合わせる．従来の方式では DNS から得られた MA の IP アドレスを信用していたため，MA から取得された LIN6 ノードの locator が正しいかどうかの検証を行わないが，Chord から LIN6 ノードの locator を取得する場合には，得られた locator の検証が必要となる．これは (a) の場合と同様に，攻撃者によって偽の locator が通知された場合に対処するためである．

Chord から得られたデータの正当性を検証できると仮定した上で (a) と (b) を比較すると，(b) で考えられるように，Chord が直接 locator の管理をすることは可能であるため，(a) のように Chord で LIN6 ID と MA の対応を解決することは冗長であると言える．また，(b) の方法では Chord ネットワークが MA の機能を吸収しているため，MA を設置する必要がない．以上から，(a) と (b) を比較した場合，(b) のほうが優れていると考えられ，本論文では (b) のように Chord ネットワークから直接 locator を解決する方法を提案する．

提案方式では図 3 のような構成を考える．MN で表されるノードは移動ノード，CN で表されるノードは MN と通信するノードである．以下の説明では，CN から MN へ通信を開始する場合を考えている．

はじめに，提案方式を考える上で次の仮定をおく．

（1） 使用する Chord ネットワークは既に存在する

（2） MN や CN は Chord ネットワークに対し，データの put/get ができる状態である

（3） MN と CN はお互いの公開鍵を知っている

仮定の (1) と (2) は，Chord ネットワークの構築や Chord ネットワークを利用するクライアントの初期化処理については提案方式の範囲外であることを意味している．データの put というのは，MN や CN が Chord ネットワークに対してデータを保存することを示し，データの get というのは，キーに対するデータを MN や CN が Chord ネットワークから取得することを表す．また，仮定の (3) は，MN と CN の間でなんらかの安全な方法を用いてお互いの公開鍵を交換しているとし，公開鍵の交換方法については提案方式の範囲外であることを意味している．

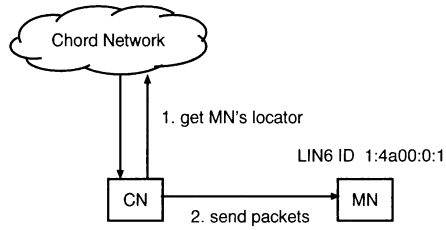

図 3 提案手法を用いた場合の通信手順

4.1 提案方式における位置登録処理

提案方式も従来の方式と同様に，MN は定期的に自身の locator 登録を行い，Chord ネットワークが保持する MN の locator を最新に保つ必要がある．

MN が Chord ネットワークに登録するデータを式 (1) に示す

$$locator, ts, \{locator, ts\}_{K_s} \quad (1)$$

ts で表されるものはタイムスタンプである．また，$\{locator, ts\}_{K_s}$ は $locator, ts$ に対して MN の秘密鍵 K_s で署名することを表す．locator やタイムスタンプだけでなく，MN の署名を付加するのは攻撃者による MN のなりすましを防ぐためである．MN は式 (1) で表されるデータを用意すると，$Hash(LIN6ID_{MN})$ をキーとしてデータの put を行う．$LIN6ID_{MN}$ は MN の LIN6 ID であり，$Hash(X)$ は X のハッシュ値を表す．

以上により，Chord ネットワークに MN の locator が保持される．ノードが MN の locator を要求する場合は，Chord ネットワークに対し，$Hash(LIN6ID_{MN})$ をキーとしてデータの get を行えばよい．データには MN の秘密鍵 K_s で署名がされているので，get を行ったノードが MN の公開鍵 K_p を知っていれば，そのデータが正しく MN によって put されたデータかどうか検証可能である．

4.2 提案方式における通信手順

CN が MN に対し通信を開始する場合は，図 3 のような通信手順を行う．ここで，CN は Chord ネットワークから得られた locator が正しく MN に put されたデータであるかどうかを，MN の公開鍵を用いて検証を行う必要がある．locator が正しく MN によって put されたものであるならば，CN はその locator を受理し，CN は MN に対し通信を開始することができる．

CN も LIN6 ノードである場合には，図 3 の手順に加え，CN の locator を MN に通知するという処理が加わり，図 4 のような通信手順となる．もし図 4 のように，CN から MN へ locator の通知を行わないとすると，MN から CN へパケットを送信する前に，MN は CN の locator を Chord ネットワークから get しなければならない．このような処理を行うと，MN が Chord ネットワークから CN の locator を get する際に生じる遅延が，MN と CN の通信に影響を与えてしまう危険性が考えられる．提案方式ではこの問題を回避するため，CN が MN に対して通信を開始する前に，CN から MN へ直接 CN の locator

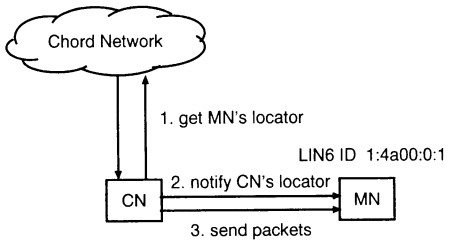

図 4　CN が LIN6 ノードの場合の通信手順

を通知することとする．CN から MN に対して locator の通知を行う際には，式 (1) と同様に，タイムスタンプと locator の情報対して CN の秘密鍵を用いた署名を行い，このタイムスタンプと locator と署名のデータを MN へ送信する．MN は得られたデータを CN の公開鍵を用いて検証することができる．

LIN6 ノードの locator が変化した場合には Chord ネットワークへの locator の put と同時に，通信ノードに対して新しい locator の通知を行うが，この通信ノードへの locator の通知は図 4 の (2) の処理と同様の処理が行われる．

5. 評　　価

この章では提案方式の性能について評価する．評価は提案方式の耐故障性，locator の探索に要する時間が通信に与える影響，また規模拡張性について行う．

5.1 耐故障性

提案方式の場合，ある LIN6 ID に対する locator を保持しているノードが故障すると，その locator の情報は失われる．よって，locator を保持していたノードが故障してから LIN6 ノードが新たに locator を Chord ネットワークに登録するまでの間，LIN6 ノードに対して通信を開始することができなる．耐故障性を向上させるため，locator の登録間隔を短くすることも考えられるが，登録間隔を短くすることによる Chord ネットワークへの影響は未評価であり，現段階では，どの程度の登録間隔を設定するのが妥当であるかの判断をすることができない．

従来方式の場合，LIN6 ID と MA の関係を管理している DNS サーバが故障したときには，その DNS サーバを復旧するか代替の DNS サーバを設置するまで LIN6 ノードに対して通信を開始できない状態になる．このような事態を防ぐため，セカンダリの DNS サーバを用意しておくことで耐故障性を向上することが可能である．また，MA の故障に関しても同様で，ある LIN6 ID に対する locator を 1 つの MA で管理している場合には一点障害が発生するが，LIN6 ID に対して複数の MA を設定することで耐故障性を向上することができる．

以上から，1 つの LIN6 ID に対する locator 情報や MA の情報を複数のノードで管理することができるという点で従来の LIN6 のほうが提案方式よりも耐故障性に優れていると言える．提案方式において耐故障性を向上させるためには，LIN6 ノードが locator を登録する時間間隔を短くすることや，Chord ネットワークで locator 情報を複数のノードで冗長に管理することなどが考えられる．

5.2 通信に与える locator の探索時間の影響

Chord ネットワークを構成するノード数にもよるが，一般的に Chord ネットワークを用いた参照には時間がかかると考えられており(注1)，LIN6 ID に対する locator の探索にかかる時間は従来方式に比べ提案方式の方が大きくなると予想される．この理由の 1 つにはオーバーレイネットワークが物理ネットワークの構造を無視して構築されることがあげられる．この参照にかかる時間を短縮するために，一度参照されたキーに対するノードの IP アドレスをキャッシュすることができる．キャッシュを持つことにより，そのキーに対する参照が次に行われた時には，該当のノードに対して直接データを要求することができるため，参照にかかる時間を短縮することができる．

しかし，提案方式における locator の探索時間が与える通信の影響は，通信開始までに遅れが生じるということに留まっている．つまり，提案方式において，locator の探索時間が大きくかかってしまうからといって，通信を開始することができなかったり，通信が途中で途切れたりすることはない．なぜならば，通信をした 2 つのノードが，その通信中に Chord ネットワークを利用して locator 探索を行うことがないからである．よって，ノード間で開始された通信に関しては，提案方式の locator にかかる探索時間の影響はほとんど無いと考えられる．

5.3 規模拡張性

2. 章で述べたように，従来方式には管理すべき LIN6 ID が増大に伴い，DNS サーバの負荷や管理コストが高くる．現在割り当て可能な LIN6 ID は 2^{40} 個であり，これだけのエントリを 1 箇所で管理することは非現実的であると言える．この問題を解決するために，DNS の木構造を利用して LIN6 ID の管理を分散的に行うことも考えられるが，このためには LIN6 ID に構造を持たせる必要がある．

しかし，提案方式では LIN6 ID を非構造のまま扱うことができ，また，1 箇所に LIN6 ID に対応するデータが集中するということもない．さらに，従来の LIN6 のように，管理者が事前に静的な関係を設定する必要がないため，管理上のコストも低い．

以上から，提案方式のほうが規模拡張性に優れていると言える．

6. 考　　察

本提案方式では，通信を行うノード間でお互いの公開鍵を事前に持つことを仮定としている．しかし，提案方式を実装し運用するためには通信を行うノード間で公開鍵を持つことは必要不可欠な問題であり，提案方式を使用するために解決しなければならない問題の 1 つである．そこで，この章では，通信を行うノード間でどのようにお互いの公開鍵をどのように取得するか，について考察する．ここでは，公開鍵を安全に取得するた

(注1)：文献 [3] では，実験結果として，180 ノードで構築した Chord ネットワークにおける参照に要する時間はおよそ 300 msec であった，ということが示されている．このネットワークは 10 のサイト間構築され，各サイト間の RTT は 60 msec であった．

めの方法として2つの方法を挙げ，それぞれについて考察する．

6.1 PKI の利用

世界規模の PKI (Public Key Infrastructure) が存在すればノードが任意の通信相手の公開鍵を安心して使用することができる．そのため，そのような通信基盤が整備されていれば，本提案方式を実運用することも可能となる．しかし，現在の PKI は未整備な状態であり，実用段階ではない．

そこで，www.lin6.net が認証局となることを考える．LIN6 ID の発行は www.lin6.net で行われているため，LIN6 ノードと www.lin6.net の間には信頼関係があると考えられる．提案方式で公開鍵暗号方式を用いた認証を行うのは LIN6 ノード間に限られるため，移動ノードも通信ノードも www.lin6.net を認証局として認めている，と仮定できる．そして，このように www.lin6.net など，LIN6 ノードが信頼することができる認証局を利用することで，LIN6 ノードの公開鍵を安全に取得することができ，提案方式が利用可能となる．

6.2 ユーザによる公開鍵の確認

この方法は ssh (secure shell) クライアントなどで行われている方法である．この方法では通信ノードの公開鍵を受け入れるかどうかの確認をユーザに対して行い，公開鍵が受け入れられた場合に限って，その後の処理を継続するというものである．

この方法を LIN6 の枠組の中で使用する場合，次のように考える．

通信の開始前には通信を行うノード同士でお互いの公開鍵を交換しあうものとする．もしお互いが通信をするのがはじめての場合，相手の公開鍵を受理するかどうかの確認をシステムがユーザに対して行う．そして，ユーザによって公開鍵が受け入れられると，通信相手 LIN6 ID に対する公開鍵としてユーザのシステム内に保存される．もし，お互いが以前に通信を行ったことがある場合は，システム内に保存されている通信相手の LIN6 ID に対する公開鍵と，相手相手から通知された公開鍵が一致するかの検証を行う．一致する場合にはその公開鍵を受け入れる．一致しない場合には，システムはユーザに対して，通知された公開鍵を受理するかどうかの確認を行う．確認がされた場合は，システム内の相手の LIN6 ID に対する公開鍵の情報を更新する．

このような方法を使用して通信相手の公開鍵を受け入れることで，提案方式を使用することができる．この方法によりユーザに対する負担は増えることが予想されるが，PKI のように認証局を用意する必要がなく，LIN6 をより decentralized なシステムにすることが可能である．

7. まとめと今後の課題

本稿では Chord を利用した LIN6 の位置解決手法を提案した．この手法を用いることにより DNS によらない LIN6 の位置解決が可能となり，LIN6 ID と locator の対応を管理する上での

- 負荷分散
- 管理，運用コストの軽減

を実現することができる．

今後は提案提案方式の実装と評価を行いたいと考えている．

文　献

[1] Masahiro Ishiyama, Mitsunobu Kunishi, Keisuke Uehara, Hiroshi Esaki, and Fumio Teraoka. Lina: A new approach to mobility support in wide area networks. *IEICE Transactions on Communication*, E84-B(8).

[2] 國司光宣, 石山政浩, 植原啓介, and 寺岡文男. 移動体通信プロトコル lin6 の性能評価. **情報処理学会論文誌**, 43(2):398–407, February 2002. マルチメディアコミュニケーションシステム特集.

[3] David Karger M. Frans Kaashoek Ion Stonica, Robert Morris and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *ACM SIGCOMM*, Aug. 2001.