DHTを活用したアプリケーションの考察

西谷　智広

（Tomo’s Hotline&Tomo’s Homepage）

2005年6月26日（日）

第1回DHTオフ会

自己紹介

経歴

◇1999年　通信系企業の研究所にてP2Pによるコンテンツ流通及びセキュリティの研究に従事

◇2002年　Tomo’s Homepage(HP)にてP2Pトピックの連載開始

◇2004年　Tomo’s Hotline(Blog)にてP2Pトピックの連載開始

現在：セキュリティサービス、コンテンツ配信等のシステムの方式検討に従事

■関心のあること（P2P)

DHTのアプリケーションの考察、P2P・DHTのセキュリティ面の対策

■趣味：

バイオリン、クラシック全般、旅行、グルメ（お茶、生牡蠣など）

プレゼン内容

1）DHTとP2P型分散SNS

2）DHTと部分列検索、P2P型分散データベース構築

3）DHTとP2P型分散プロキシ

4）DHTとP2P－VPN

第1章：SNSとDHT

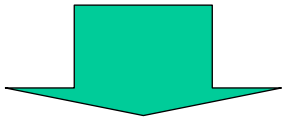
１－１．現在のSNSの状況

☆ユーザ数急増

⇒[1]レスポンスタイム等の低下

⇒[2]サーバ、回線の増強

⇒[3]SNS運用企業の設備投資増加



SNSを利用する際ユーザのPCリソースを活用する事で
SNS運用企業の負担を軽減することはできないか？

DHTを使えば解決できそうだ！

１－２．P2PでSNSはできる？

□P2Pアプリの「Ringoch」「新月」で次のようなことが既に可能。

☆掲示板

☆Blog

☆wiki

☆ファイルアップロード

※Pure-P2P（後述）で動いている

□SNSでは上記の他に

☆公開の制限（友達、友達の友達まで）

☆招待した人でないと加入できない

本プレゼンの焦点！

という機能が必要かも。

１－３．分散的なSNS～affelio

□affelio　　http://affelio.jp

□ユーザのデータは自PC内で管理

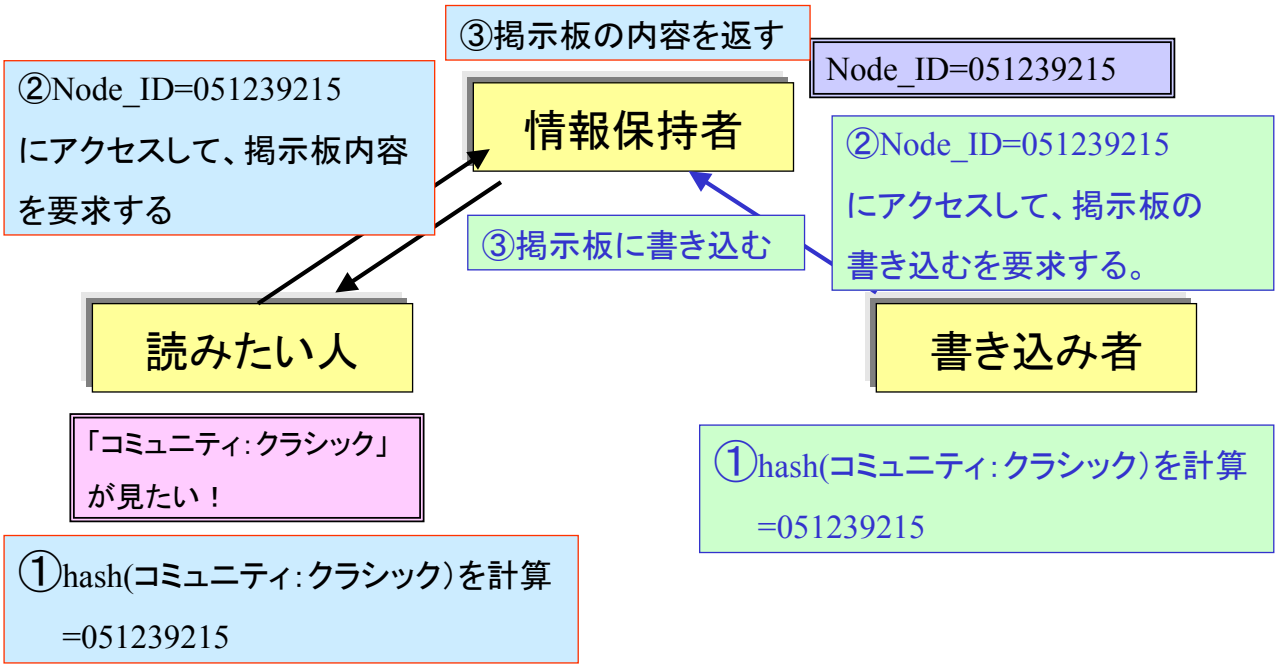
⇒ユーザのPCの電源を落ちたら他人からその情報を参照できない！

（※もともと想定がユーザがレンタルサーバを借りる事。）

本提案では数10億台の接続が可能で、ノードの電源が落ちても

そのノードユーザ情報が消えないSNSを説明する。

１－４．DHTを使った例：掲示板



□Blog、Wikiも同様の手法で実現可能

１－５．P2Pでの信頼性の保障

「ユーザの身分」を誰が保証するか？

□方法は大きく分けて2つ。

案1）信頼できる人（SNS運用会社）などがその人の身分を保証

⇒PKI的なアプローチ

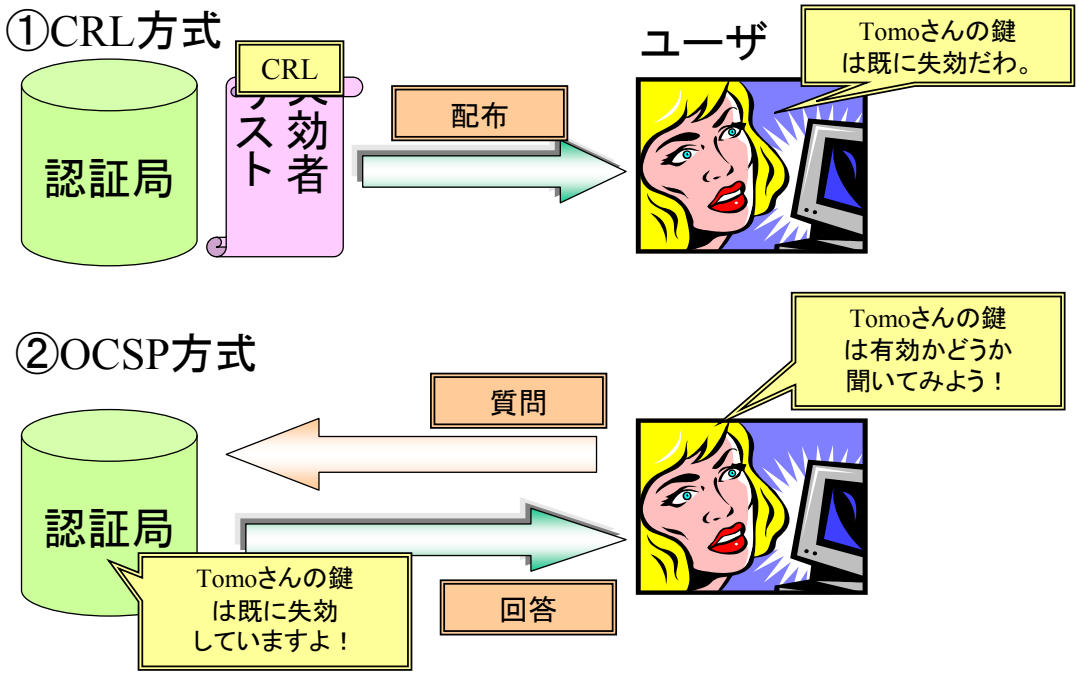
案2）他人あるいは自分自身がその人の身分を保証

⇒PGP的なアプローチ

企業ベースで運用するには案1で望ましいと考えられる。

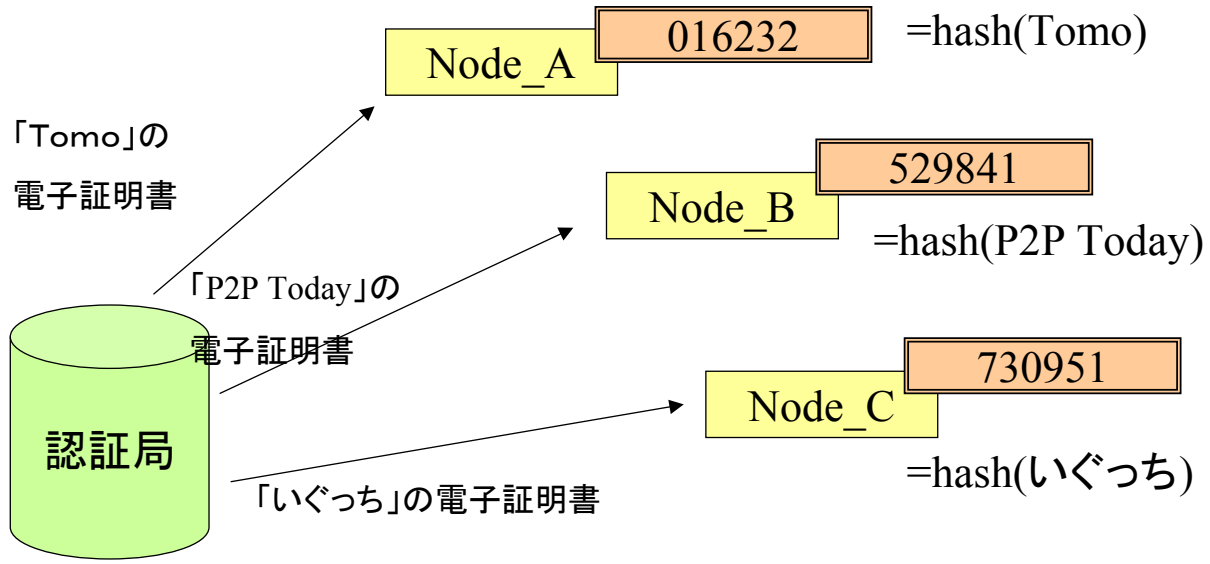
⇒非営利、個人ベースの運用であれば案2という解もある。

１－６．電子証明書の有効性確認の方法



DHTを使ってCRL、OCSP各方式の「良いところ」を融合

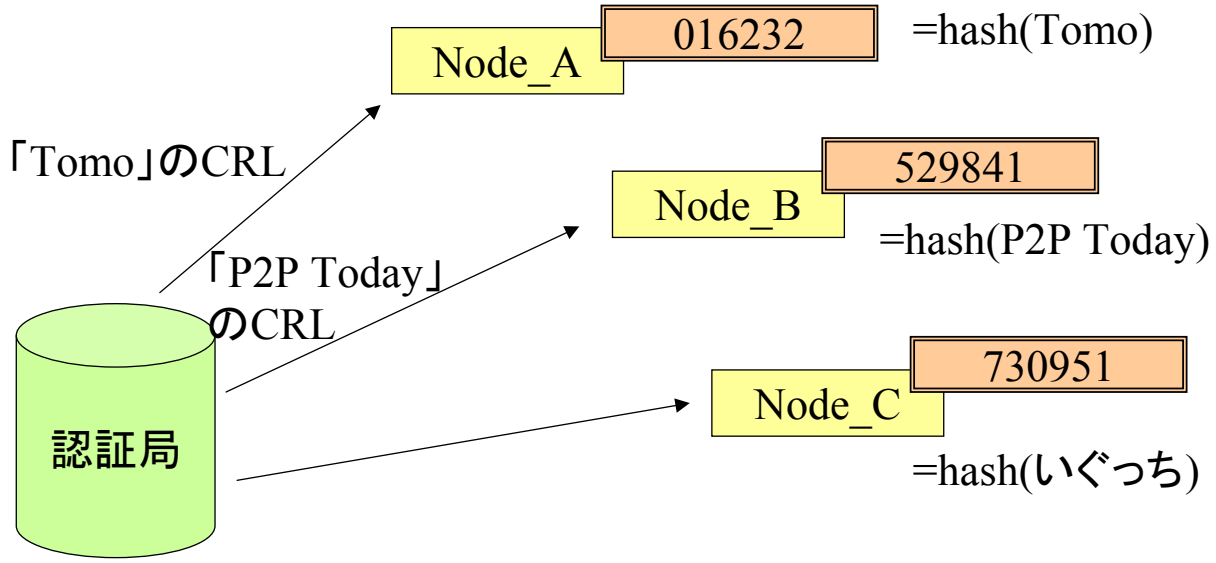
１－７．DHTによる電子証明書の管理



※電子証明書は随時発行

各ノードが電子証明書の分散的なレポジトリとなる

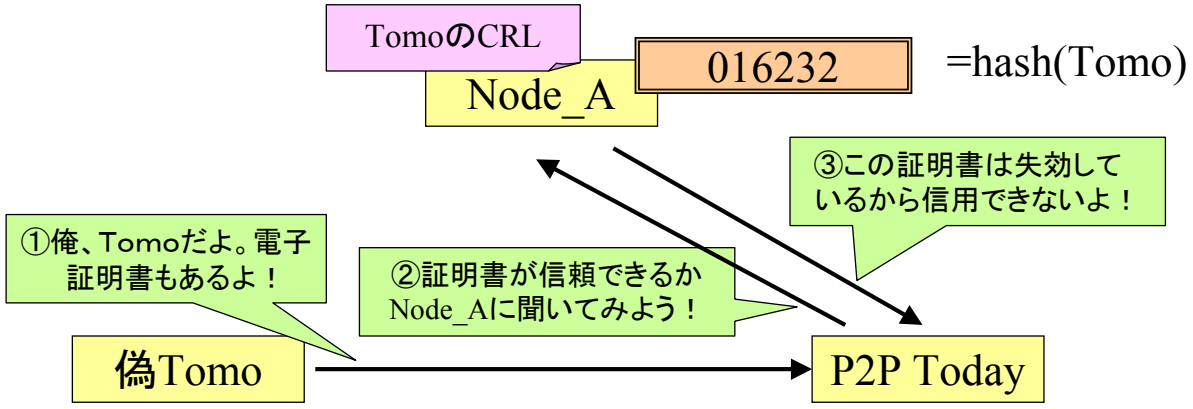
１－８．DHTによるCRLの管理



※CRLは随時発行

各ノードがCRLの分散的なレポジトリとなる

１－９．DHT＋CRLを利用した電子証明書の有効性の判別



メリット

①認証のための負荷が分散される

②即時に電子証明書が失効かどうか判断できる

１－１０．SNSに加入する時のユーザ作業

□P2P－SNSアプリをPCにインストール

□SNS運用会社から電子証明書をもらう

⇒ユーザの「身分証明書」と同様。SNS運用会社からの「お墨付き」

⇒秘密鍵はユーザのPC内に閉じる。

□電子証明書をDHT（P2P）で公開。

※証明書内容の変更、SNSからの脱退等はSNS運用会社に通知。

コミュニティからの脱退はコミュニティ管理者へ通知。

１－１１．友達を証明する仕組み「FOAF」

TomoがP2P Todayを友人と記述する場合

```
<rdf:RDF  ….>
  <foam:PersonalProfileDocument rdf:about="">……
   </foaf:PersonalProfileDocument>
   <foaf:Person rdf:nodeID="me">
    <foaf:name>Tomohiro Nishitani</foaf:name>
    <foaf:givenname>Tomohiro</foaf:givenname>
    <foaf:family_name>Nishitani</foaf:family_name>
     <foaf:nick>Tomo</foaf:nick>
     <foaf:mbox rdf:resource="mailto:aaa@bbb"/>
          <foaf:knows>
             <foaf:Person>
                    <foaf:name>P2P Today</foaf:name>
                    <foaf:mbox rdf:resource="mailto:ccc@ddd"/>
              </foaf:Person>
           </foaf:knows>
   </foaf:Person>
</rdf:RDF>
```

自分の情報 =Tomo

友人の情報 =P2P Today

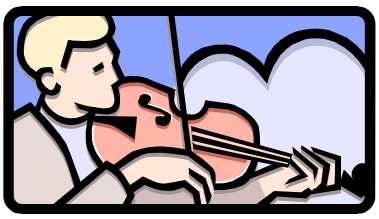
http://www.ldodds.com/foaf/foaf-a-matic.ja.htmlで生成（一部編集）

15

□電子署名をして、改ざんを防止する。

１－１２．友達の友達（2次の繋がり）を証明しよう！

友達



友達

Nao　　Tomo　　P2P Today

①NaoがP2P Todayを認証（チャレンジ・レスポンス方式）P2P Todayの電子証明書を活用]

②NaoにTomoと友達である事を通知

③NaoのFOAFにTomoが含まれている事を確認

④TomoのFOAF、電子証明書をDHTから検索し、検証。FOAFにP2P　Todayが含まれる事を確認。

⑤NaoはP2P　Todayに自分の情報を公開

N次の繋がりまでの公開も理論的には可能

6

１－１３．コミュニティに対する補足

□コミュニティメンバの管理

管理人が電子書名付きXMLで管理？（電子署名付きFOAFもあり。）

NodeID=hash(コミュニティ名）にXML等を格納

□限定コミュニティの場合、SNS内容の暗号化？

⇒特定のメンバ間で共有する内容は暗号化して共通キーを

シェアすればよい？

（キーをシェアするタイミング：管理者が参加者のコミュニティ参加を

承認した時、参加者の公開鍵を用いて暗号化メールで共通鍵を配送）

１－１４．提案方式の懸念点

□悪意のあるユーザをどうするか？

⇒情報を保持するユーザと、情報要求するユーザの結託の恐れ

（一般に2者は互いに完全独立だから結託は難しい？）

※秘密分散法等で格納情報を分散すれば良い？

□意図的に格納しているはずのデータをロストするノードは？

⇒あるノードにデータがない場合には、隣接ノードに別途

問い合わせ？

□もしPGP的な電子証明書を発行するのであれば、

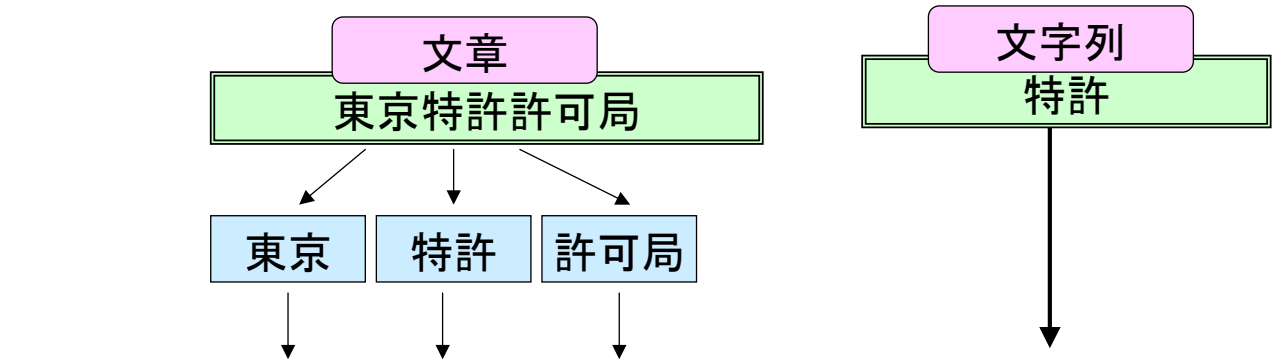
FOAFだけでなく、他の信用評価手段も必要？

例：EigenTrustの導入

第2章：DHTと部分列検索、データベース構築

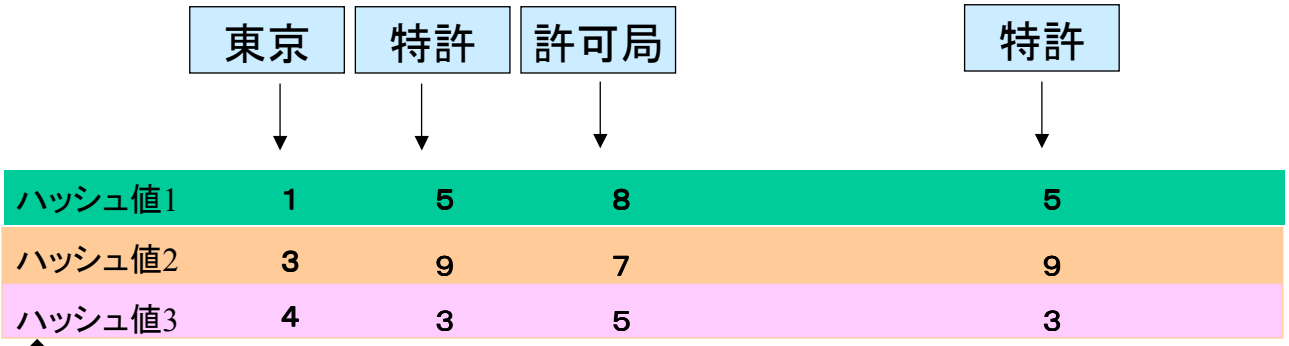
2－1．Bloom　Filterとは？～その1

□文章の中に、ある文字列が含まれている「かもしれない」事を
高速に判別する技術



ハッシュ値1	1	5	8	5
ハッシュ値2	3	9	7	9
ハッシュ値3	4	3	5	3

2－2. Bloom Filterとは？～その2



	0	1	2	3	4	5	6	7	8	9
東京	0	1	0	1	1	0	0	0	0	0
特許	0	0	0	1	0	1	0	0	0	1
許可局	0	0	0	0	0	1	0	1	1	0
東京特許許可局	0	1	0	1	1	1	0	1	1	1

東京特許許可局 ＝東京 or 特許 or 許可局 !!

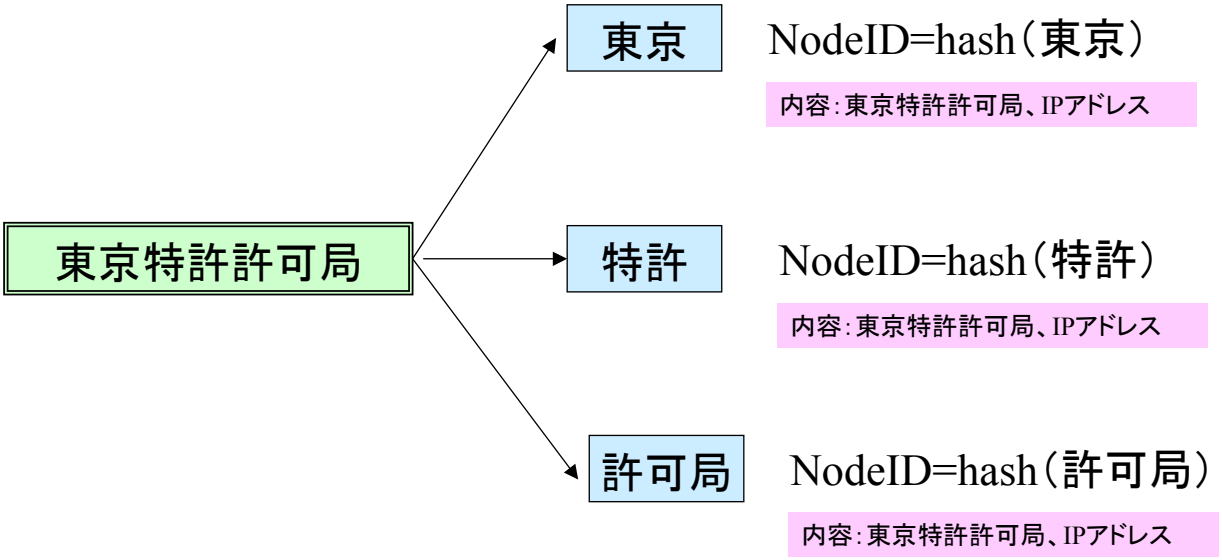
2－3. Bloom Filterとは？～その3

	0	1	2	3	4	5	6	7	8	9
東京	0	1	0	1	1	0	0	0	0	0
特許	0	0	0	1	0	1	0	0	0	1
許可局	0	0	0	0	0	1	0	1	1	0
東京特許許可局	0	1	0	1	1	1	0	1	1	1

特許	0	0	0	1	0	1	0	0	0	1

Bloom Filterを見れば、東京特許許可局に特許が含まれていることがわかる！

２－４．DHTにおける部分列検索の方法の提案～その１



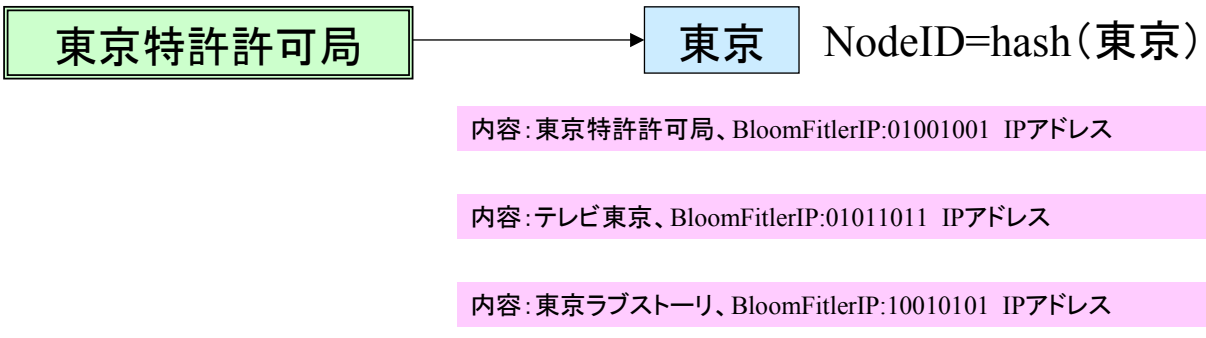
節を単語に分解してその内容を各ノードに格納することが考えられる。

問題点：「東京」に東京に関するたくさんの数の内容が格納される！！

さらに絞り込むのが大変！！

２－５．DHTにおける部分列検索の方法の提案～その２

高速に絞り込むにはBloom Filterを使えばいいのでは？



絞込み単語　特許、BloomFitlerIP:01000001 IPアドレス

特許のBloomFilterと各内容のBloomFilterをOR計算すると

「東京特許許可局」だけが「特許」を含む可能性がある！

※「特許」かつ「許可局」を含む内容絞込みも同様な方法で可能

２－６．DHTにおける部分列検索の方法の提案～その３

BloomFilter⇒ハッシュ値を使用
DHT⇒ハッシュ値を使用
} うまく融合できないか？

例えば、

NodeID= 上位X bit　ハッシュ関数（タイトル名等）
　　下位Y bit　BloomFilter（タイトル名等）

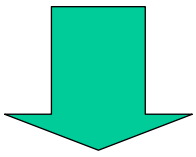
なかなか検索効率が上がらない。。

詳しくは、

http://toremoro.tea-nifty.com/tomos_hotline/2005/04/dhtdht_3755.html を参照

２－７.データベースとDHT～その１

氏名	出身地
Tomo	埼玉
P2P　Today	群馬（高崎）
Aki	東京
Yoshizawa	東京



主キーのハッシュ値で昇順

氏名	ハッシュ値（氏名）	出身地
Tomo	1039285821	埼玉
P2P　Today	2039584192	群馬（高崎）
Yoshizawa	3396516902	東京
Aki	5029182321	東京

2－8. データベースとDHT～その2

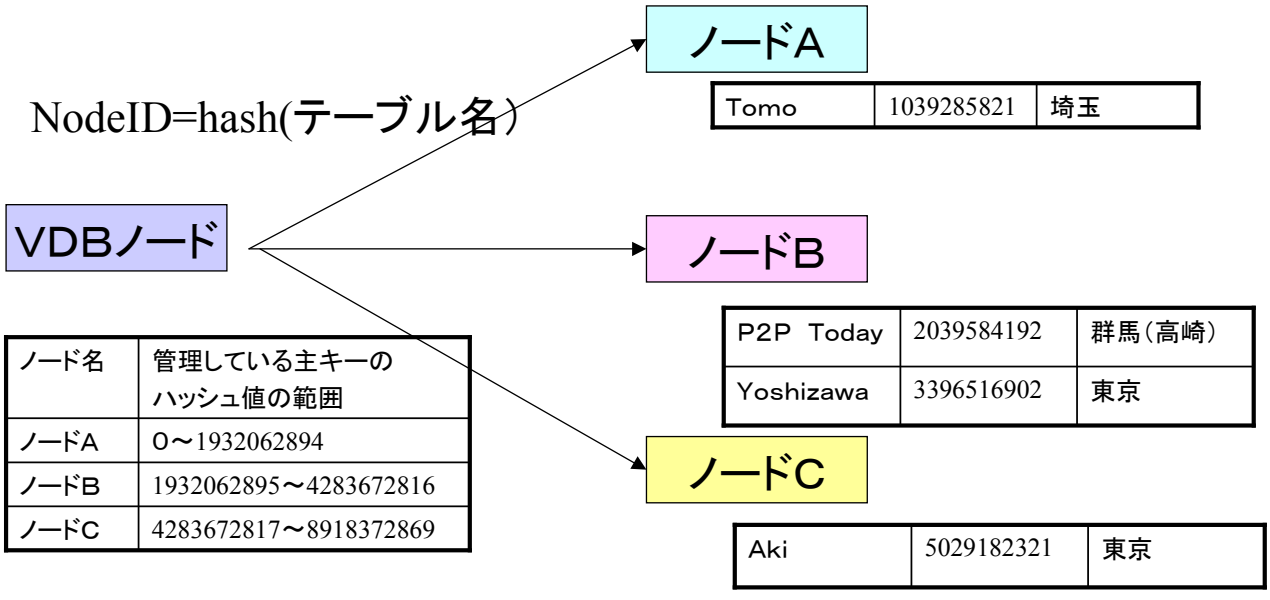
氏名	ハッシュ値（氏名）	出身地	管理ノード
Tomo	1039285821	埼玉	ノードA
P2P　Today	2039584192	群馬（高崎）	ノードB
Yoshizawa	3396516902	東京	
Aki	5029182321	東京	ノードC

データベースが「データベースの大きさ」、管理ノードの処理能力等によって、「自律的に」分割される

2－9.データベースとDHT～その3

バーチャルDBノード(VDBノード)の導入

分割した各DBを保持しているノードの管理



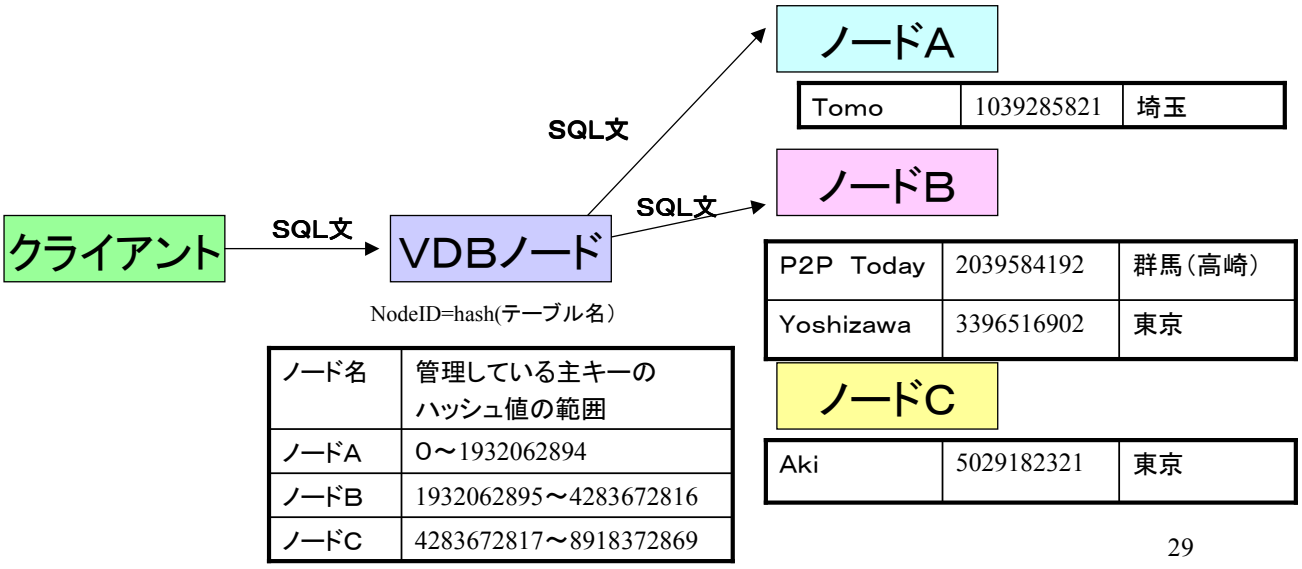
ノード名	管理している主キーのハッシュ値の範囲
ノードA	0～1932062894
ノードB	1932062895～4283672816
ノードC	4283672817～8918372869

2－10. データベースとDHT～その4

あるノードがSQL文をVDBノードに渡す。

VDBノードは各ノードにSQL文を渡し、処理を委託。

※VDBノードはSQL文を解読し、SQL文を渡すノードを最適化



ノード名	管理している主キーのハッシュ値の範囲
ノードA	0～1932062894
ノードB	1932062895～4283672816
ノードC	4283672817～8918372869

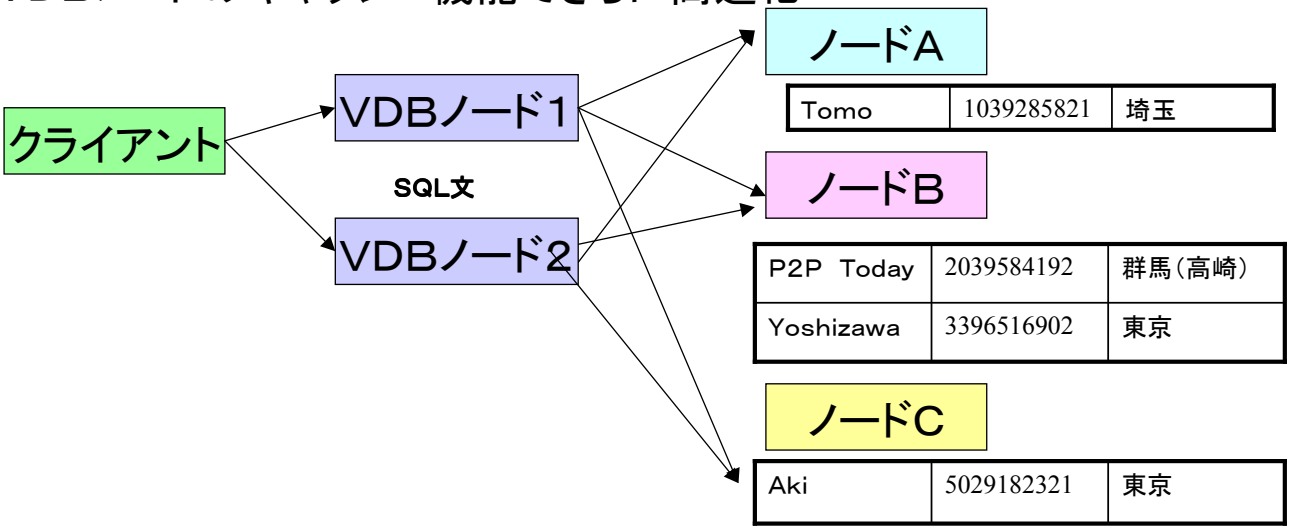
2－11. データベースとDHT～その5

VDBノードの負荷分散

※DBのロック機能などを司るノードが必要？

DBの参照だけなら問題なし。

VDBノードのキャッシュ機能でさらに高速化

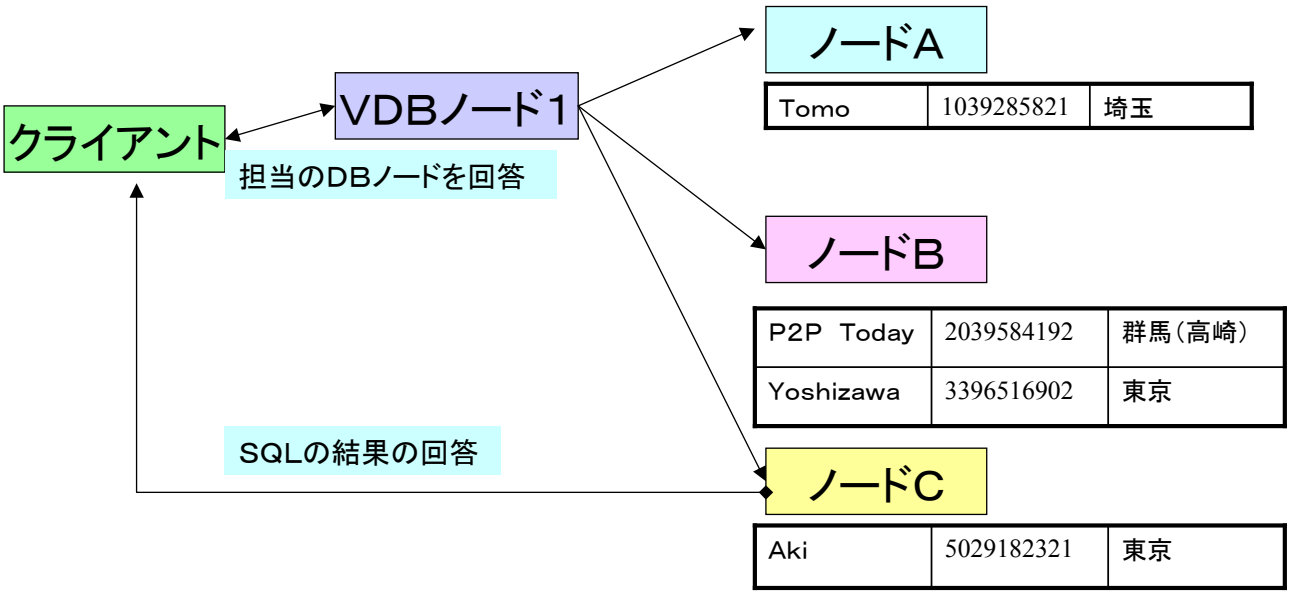


2－12. データベースとDHT～その6

VDBノードの負荷軽減及びDB参照の高速化

ロードバランサのDSR機能と類似

※SQL文によっては参照以外も応用可能



2－13. DHT型分散データベースのターゲット

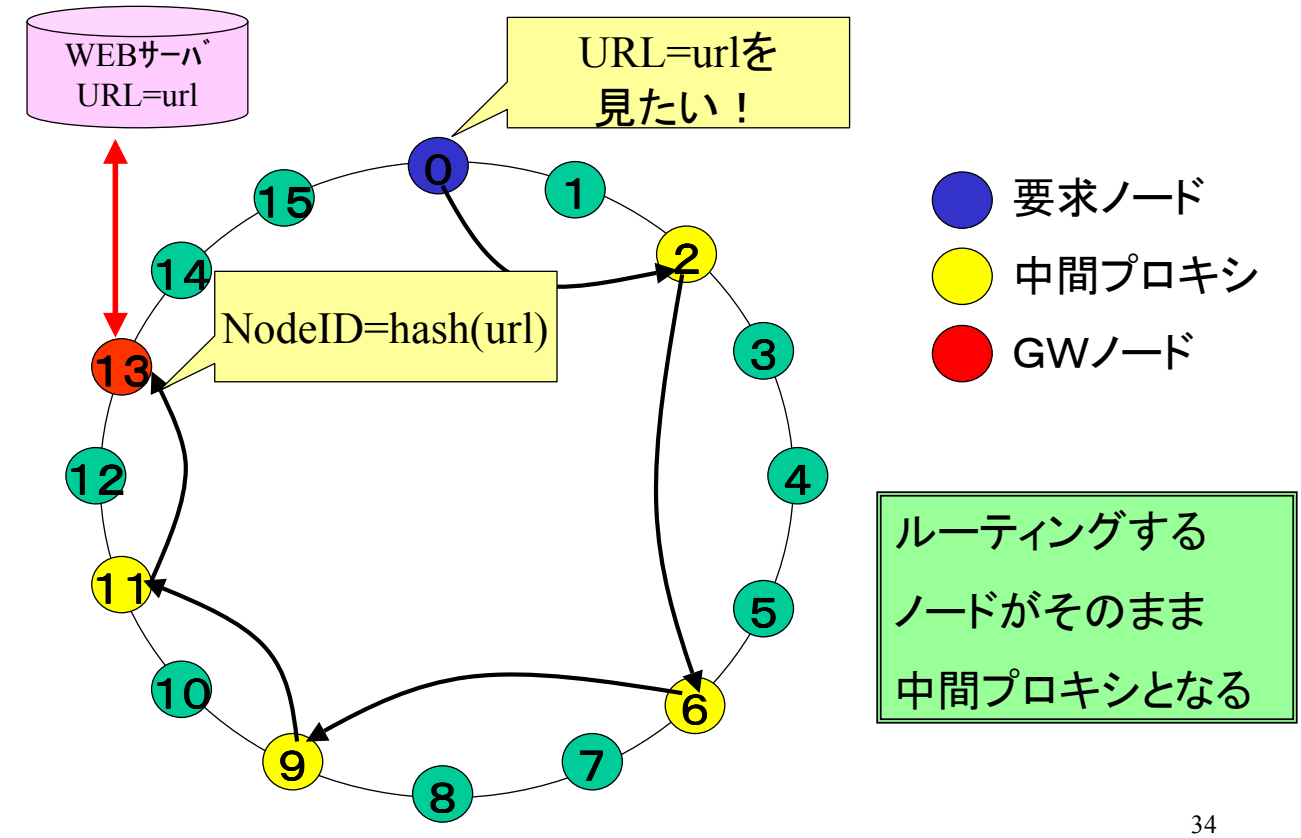
□高速・高頻度の情報参照が可能

（単なるDHTによる情報格納による参照より一般的には高速）

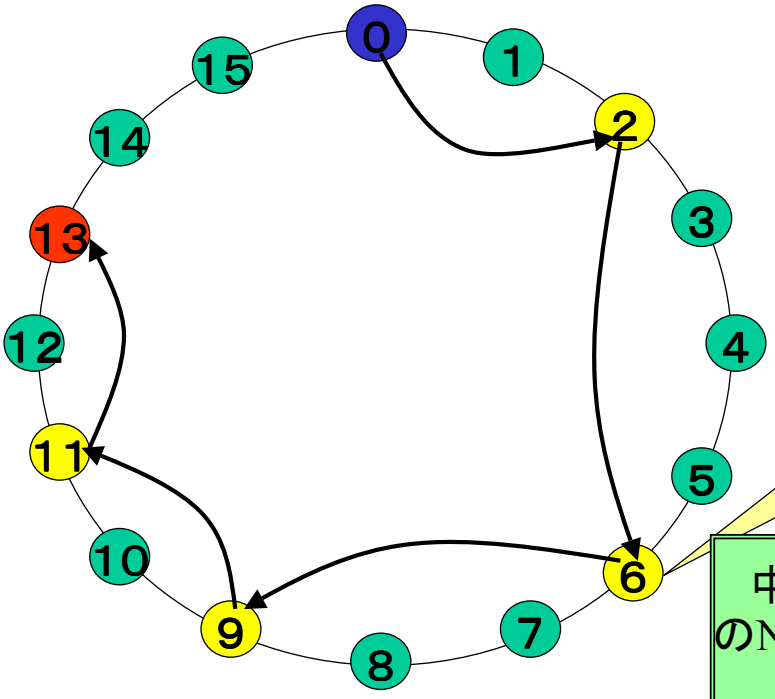
□巨大なデータベースの構築が可能

第3章：DHTと分散プロキシ

3－1．DHTにおける分散プロキシの基本的原理



３－２．要求ノードの推測の可能性



要求URLのハッシュ値は13だし、NodeID=2のノードから要求が転送されているから、要求ノードはNodeIdが0,1,2,14,15に違いない！！

中間プロキシが要求ノードのNodeIDの範囲を特定できるかもしれない。。。

３－３．ChordとSymphony

ルーティングするノードとの距離

□Chord

2^0、2^1、2^2、2^3　……　2^n

⇒完全に決まっている！

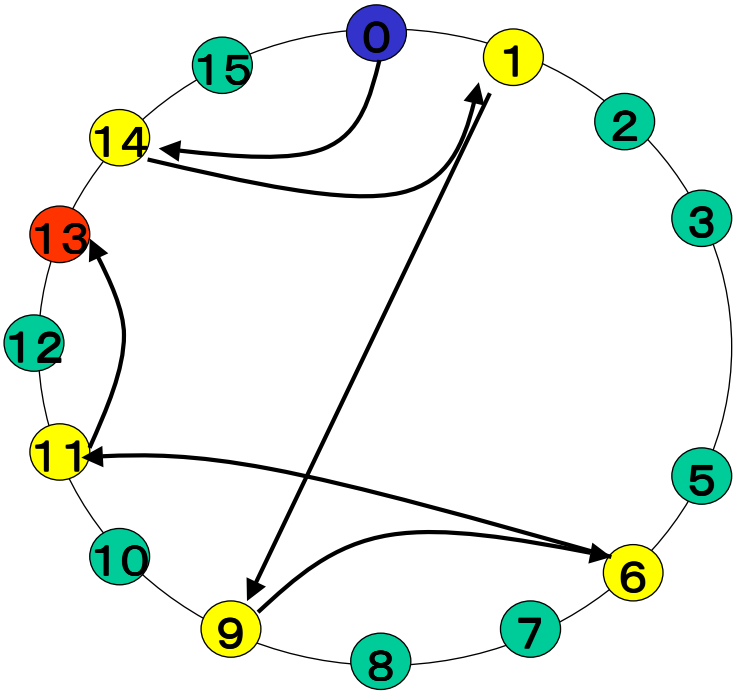
□Symphony

Chordのルーティングのノードとの距離の概念を

連続化、さらにランダムネスの要素！

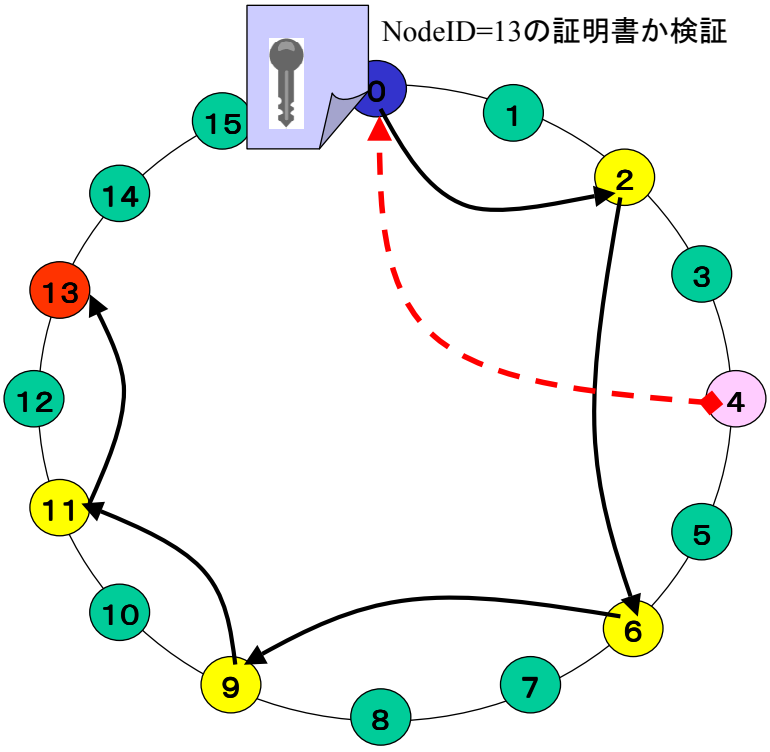
⇒Symphonyの方が送信元を特定しにくい

3－4. ルーティングのランダムネスの導入



完全にDHTのルーティングに従うのではなく、ある確率で逆方向へのルーティング、あるいは最適化にならないネクストホップへのルーティングなどの概念を導入

3－5. 通信の暗号化

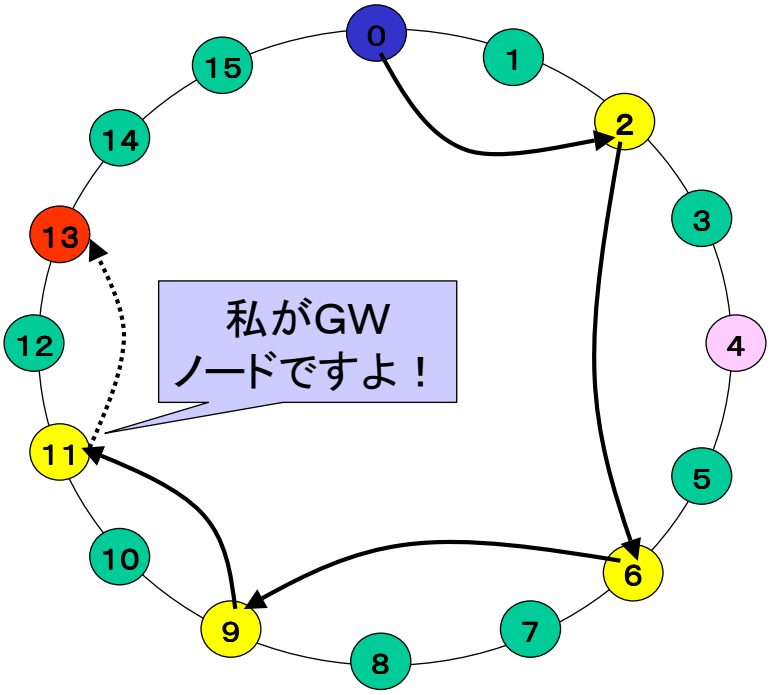


NodeID=13の電子証明書を持っているノード

電子証明書

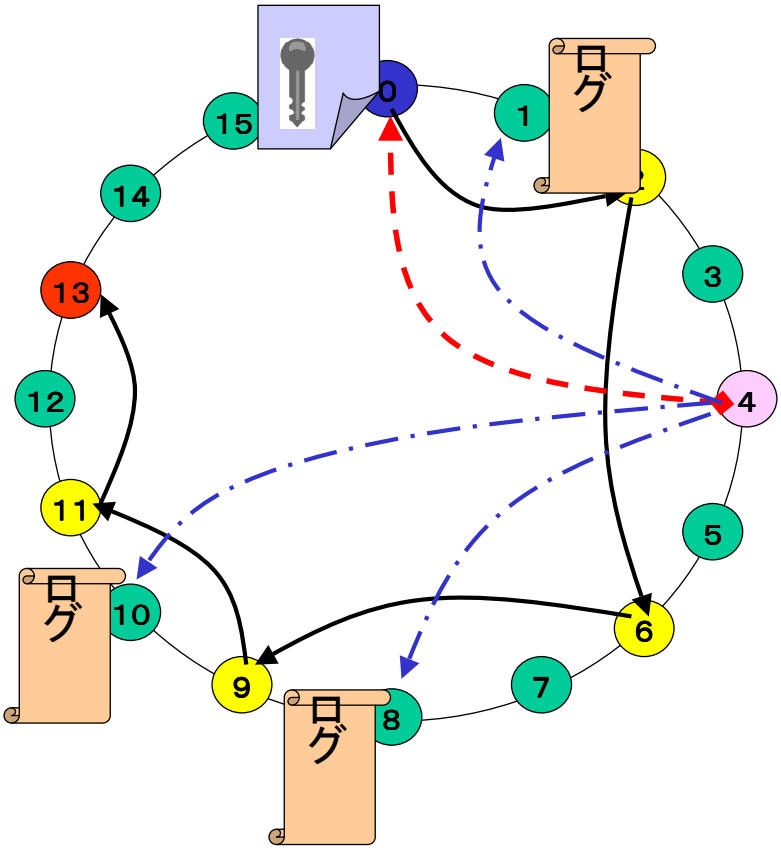
電子証明書を使って、GWノードに気づかれずに暗号化可能

3－6. 提案方式の暗号化アタックの可能性



GWノードと充分距離が小さい中間プロキシがGWノードになる済ましする可能性あり。
⇒悪意のあるノードが中間プロキシになるかどうかわからないし、送信元も要求URLも分からないのにアタックしても意味ある？

3－7. ログの秘密分散分散

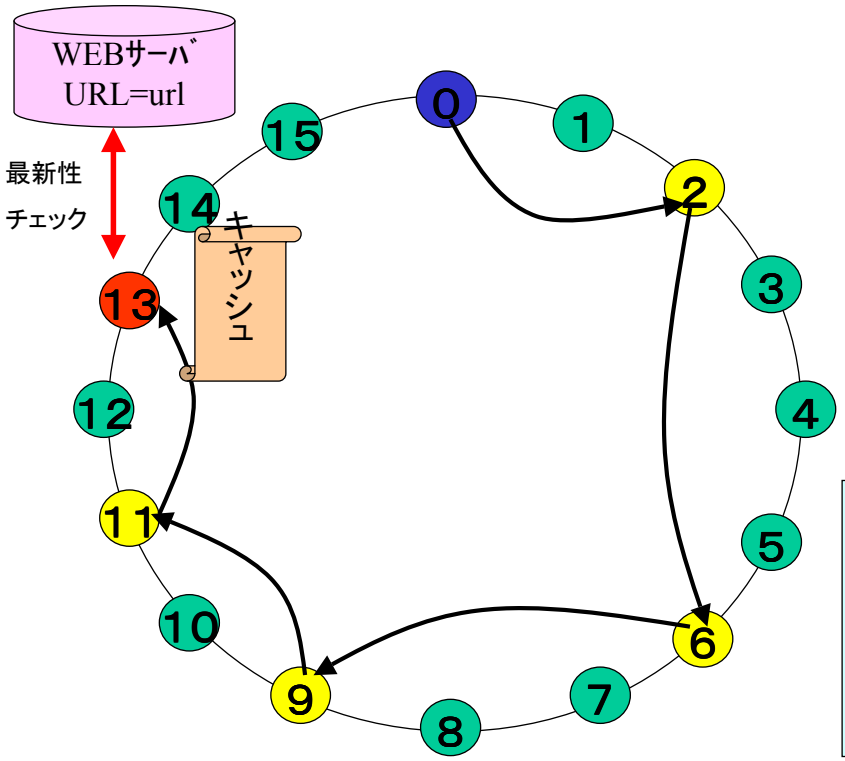


NodeID=13の電子証明書を持っているノード

電子証明書

電子証明書を取ったノードが分かれば特定GWノードにアクセス要求するノードが判明する
⇒ログは秘密分散で複数ノードへ

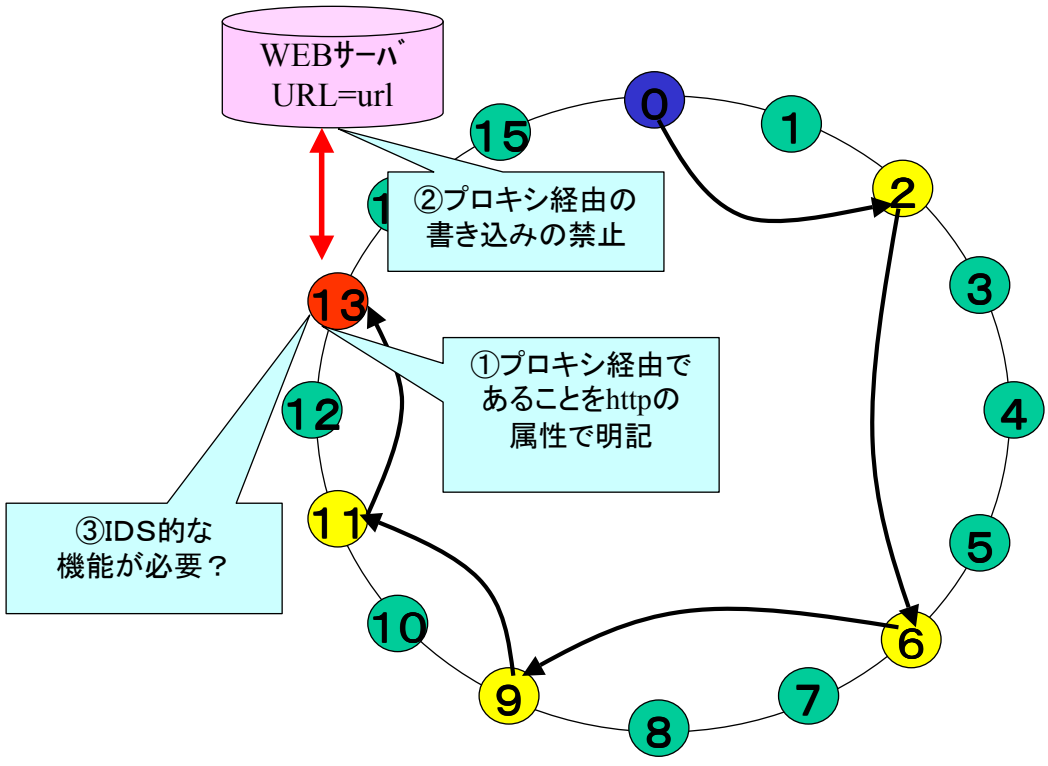
3－8. 分散プロキシとキャッシュ



GWノードは最新性のチェックのみ。
これによりWEBサーバへの
負荷低減が図れる。
⇒2chブラウザなどに応用化？

昔のWEBをスナップショット
のように残す事も。

3－9. スパム的書き込み対策



３－１０．匿名と非匿名のバランスなど

□悪意のあるユーザの特定ができるような機能追加は
開発者側にとっては必須？過度な匿名機能はあまり意味
がない。（学術的には別として。）

⇒Winny事件

□匿名プロキシというよりもWEBサーバへの負荷分散を意識
したプロキシという思想の方が良いかも。
例：2chブラウザ

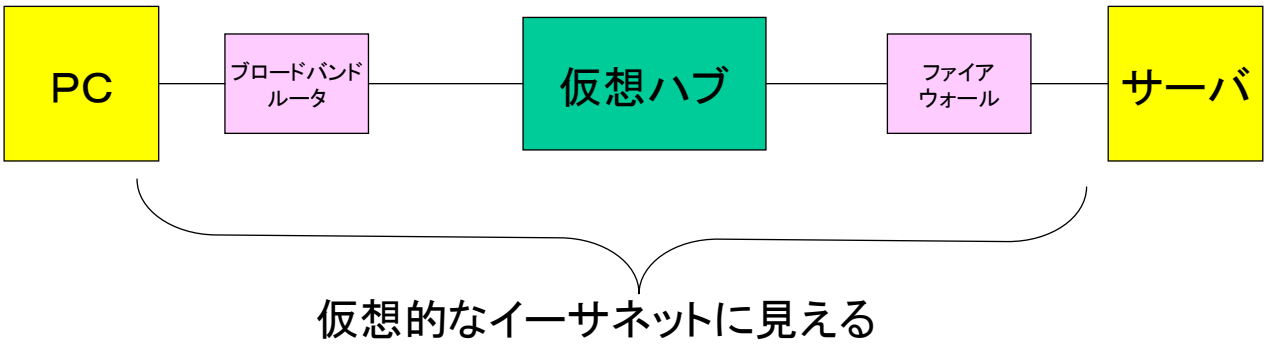
□ブラウザにDHT機能を盛り込むことによりDHTミドルウェアの
ユーザへの浸透が図れる。

⇒一般ユーザへのDHTミドルウェア導入の導線

第４章：DHTとP2P－VPN

４－１．SoftEtherとは？

仮想L2VPNの代表例

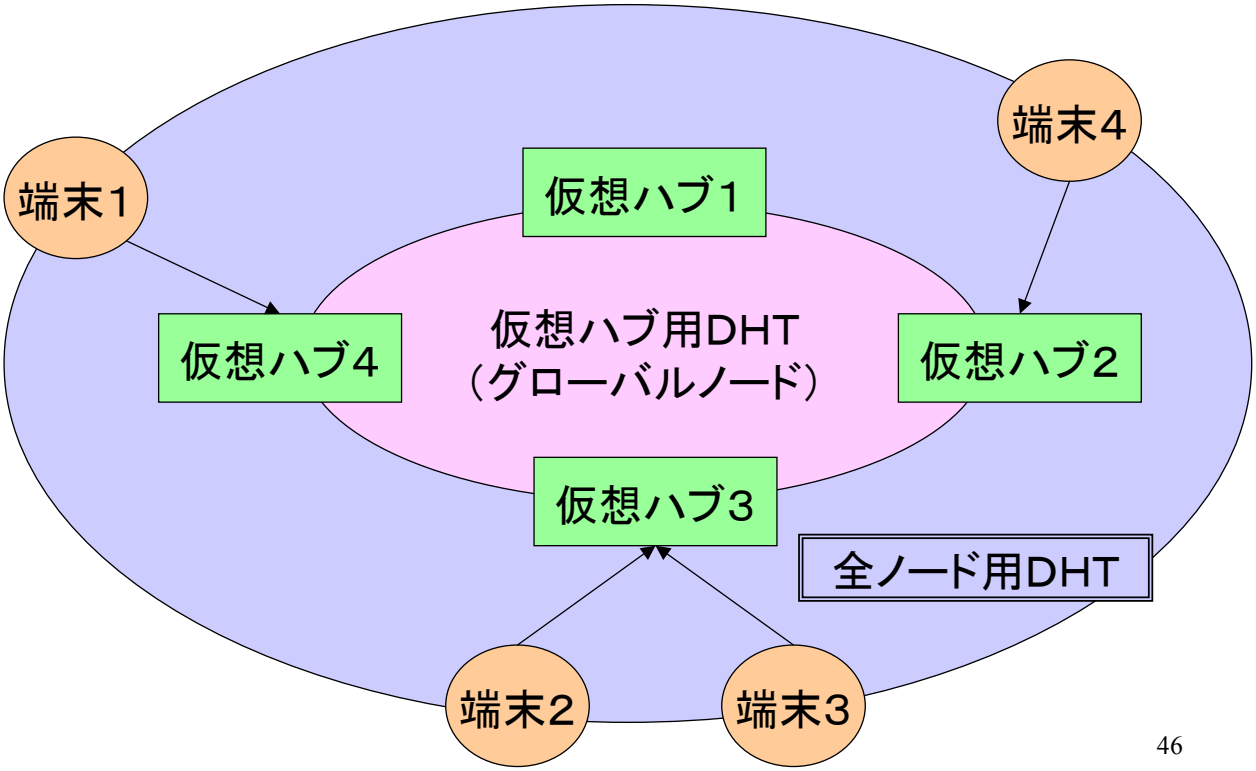


課題　[1]仮想ハブにトラフィックが集中

　　[2]仮想ハブがダウンすると全通信がストップ

⇒DHTで上記課題を解決したい！

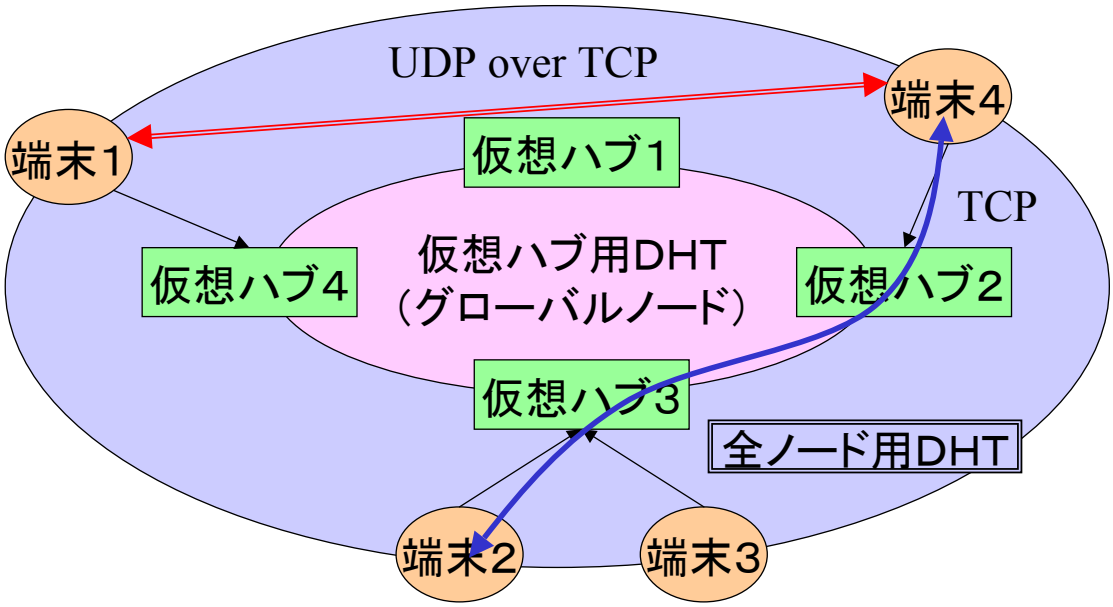
４－２．仮想ハブとDHTのイメージ



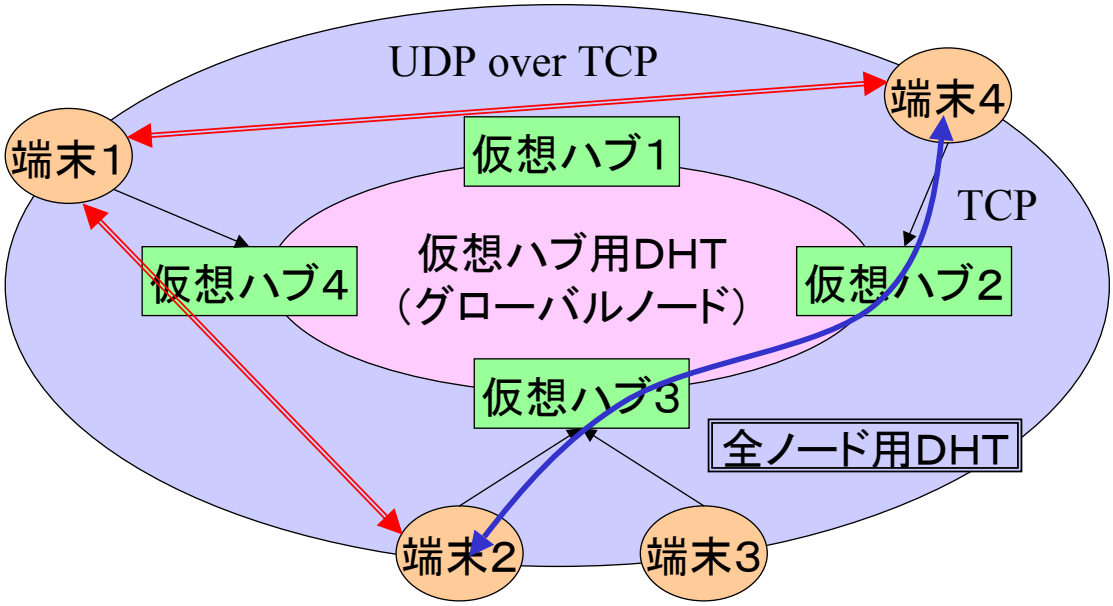
4－2. 仮想ハブとSTUNサーバ

仮想ハブはSTUNサーバを兼ねる。端末情報は仮想ハブ用DHT
で交換。（skypeのスーパーノードライクな感じで）
通常ノード間通信はUDP Hole Punchingを使ってTCP over UDP

上記が可能でない場合は仮想ハブ経由でTPC通信

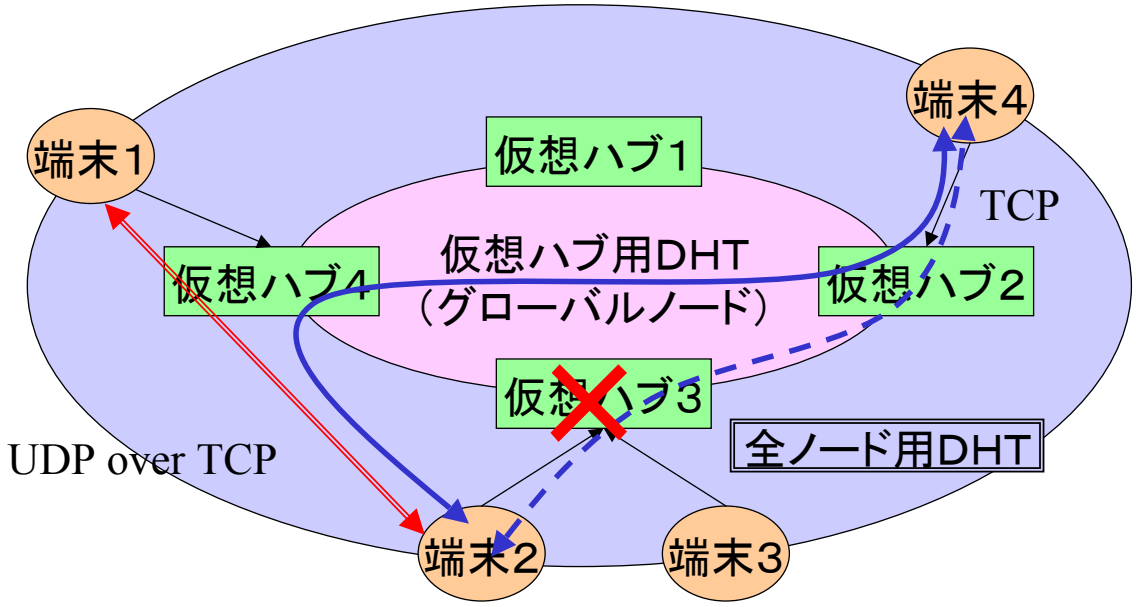


4－3. トラフィックの分散



特定の仮想ハブに通信が集中しない。

４－４．仮想ハブのダウン



既にUDP hole punchingに成功しUDP通信しているノード間は問題なし。

TCP接続は他の仮想ハブにセッションを引き継ぐ。

（分散ストレージのように仮想ハブDHTにより隣接に仮想ハブに定期的に

セッション管理情報を送る、端末も全ノードDHTにより他の仮想ハブ情報を取得しておく）

４－５．VPN認証方法

仮想ハブに盗聴、改ざんされないようにしないと。。

１．ノード間でセッション前にシェアードキー

２．電子証明書方式

⇒**Diffie**-Hellmanは仮想ハブによる中間侵入攻撃が可能！

４－６．MACアドレスとハッシュ値

ノードID（ハッシュ値）⇒ユニークな値のはず。

MACアドレス（仮想）⇒ユニークな値のはず。

ノードIDからMACアドレスが生成できそうだ。。。

電子証明書と組み合わせると、仮想MACアドレスの

偽装が防止できる？

例： Tomoの電子証明書

NodeID＝hash_1(Tomo)

(仮想) MACアドレス=hash_2(hash_1(Tomo)) mod 48bit

⇒なりすましができない！

４－７．課題

□ブロードキャストをどうする？１対Nは苦手かも。

□仮想ハブダウンによるセッション引継ぎは実装大変？

⇒でもロードバランサで実現できているし。。。

仮想ハブにセッション引継ぎ機能を持たすよりも、

むしろノードが自律的に他の仮想ハブを探してセッション

を維持するように設計？

□VLANの動的制御！⇒MACアドレスVLANライク？

複数L2VPNをどうやって仮想ハブに収容する？

L3VPNの方が設計が楽？

ご清聴ありがとうございました。