JIPDEC 情報共有（P2P）研究会
2006年 1月 13日（金）

Unstructured overlayと Structured overlay

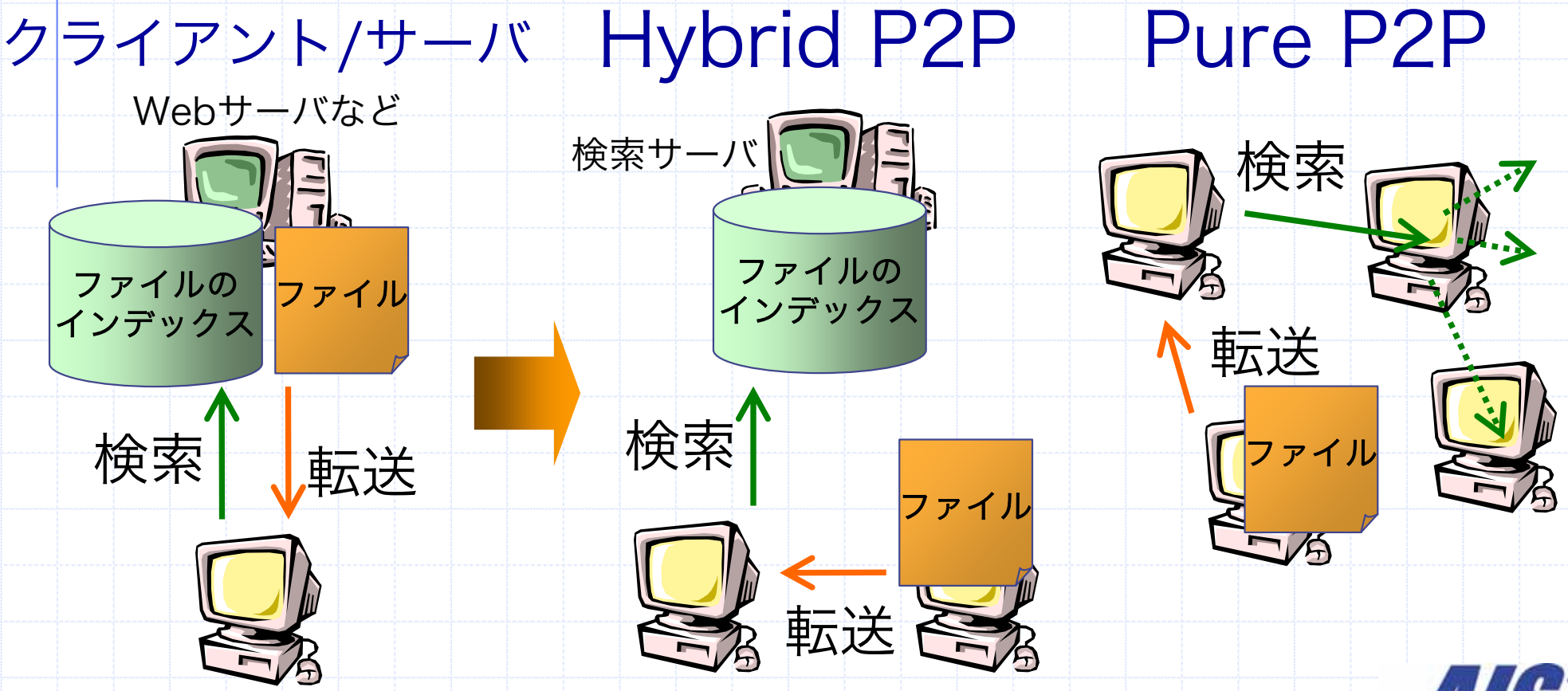
首藤 一幸

産業技術総合研究所 (AIST)

グリッド研究センター (GTRC)

前置き: hybrid / pure P2P

- P2Pファイル共有ソフトの例



オーバレイ (overlay)

- pure P2P → 非集中 (decentralized)
  - 集中サーバなしに何らかの機能を果たすため、アプリケーションレベルのネットワークを構成する。
    - 例: ファイル共有ソフトのネットワーク
      FastTrack, eDonkey2K, Gnutella, …
      ↑ ノード数 100万以上
    - 機能： 発見, マルチキャスト, メッセージ配信, …
- そのトポロジは下位ネットワーク (インターネット) の物理的トポロジとは独立
  - それゆえ、ネットワークオーバレイと呼ばれる。
  - 注）オーバレイ：ネットワークに被せられた別トポロジのネットワーク一般を指す。P2P 関係だけの用語ではない。

AIST

Unstructured と Structured

- Unstructured オーバレイ
  - 例： Gnutella ネットワーク, Winnyネットワーク
  - 誰を隣接ノードとするか、トポロジに制約がない。
  - 存在するオブジェクトは、発見できる可能性がある。
  - 一般に、効率は良くないが、柔軟な検索が可能。
- Structured オーバレイ
  - 例： DHT（分散ハッシュ表）のネットワーク
  - 誰を隣接ノードとするか、トポロジに制約がある。
  - 存在するオブジェクトは、（たいてい）発見できる。
  - 一般に、効率は良いが、柔軟な検索が苦手。

AIST

Unstructured と Structured

- 当初、pure P2P のファイル共有ネットワークは unstructured オーバレイだった。
  - Gnutellaネットワーク, Winnyネットワーク
- 最近は、structured オーバレイの応用も始まっている。
  - eDonkey2k, eMule の Kadネットワーク
  - BitTorrent、Azureus のトラッカーなし動作

Unstructured と Structured

- 豆知識
  - Structuredオーバレイ上でも柔軟な検索を達成しよう、という研究がある。
  - 両者のアルゴリズムを組み合わせて、いいとこ取りをしようという研究がある。
    - 例：Structuredオーバレイの構造を利用したflooding / random walkで検索を行う。
    - 例：Structuredオーバレイの構造を利用して、高効率 / 高伝達率のブロードキャストを行う。
  - JXTA 2.x での（広告の）検索は、DHT的な検索 + それでダメならクエリの転送。
    - Structured 的方法 + Unstructured 的方法

AIST

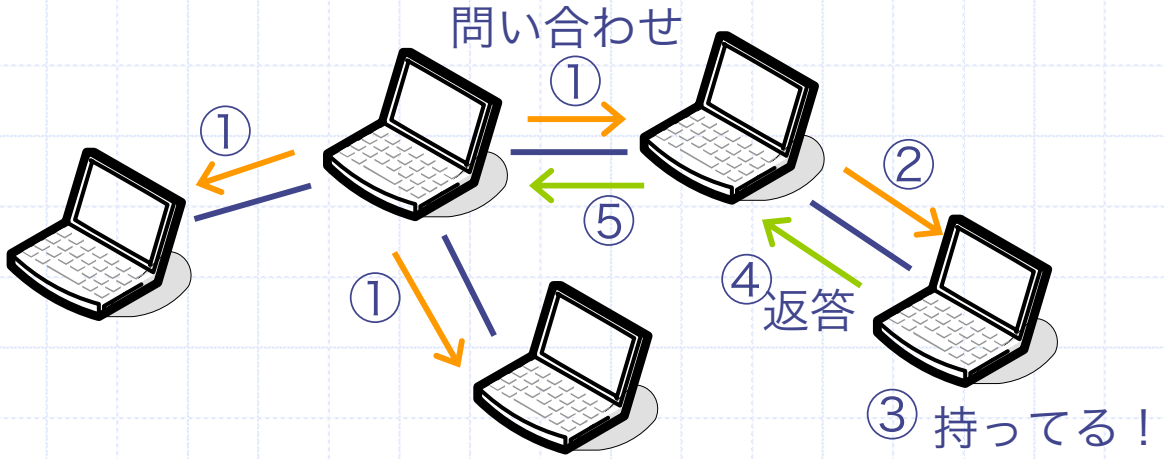
具体例

- Unstructured オーバレイ
  - Gnutella プロトコル (0.4)
- Structured オーバレイ（アルゴリズム）
  - Chord
  - Pastry
  - Kademlia

Unstructuredオーバレイの例:

Gnutella プロトコル (0.4)

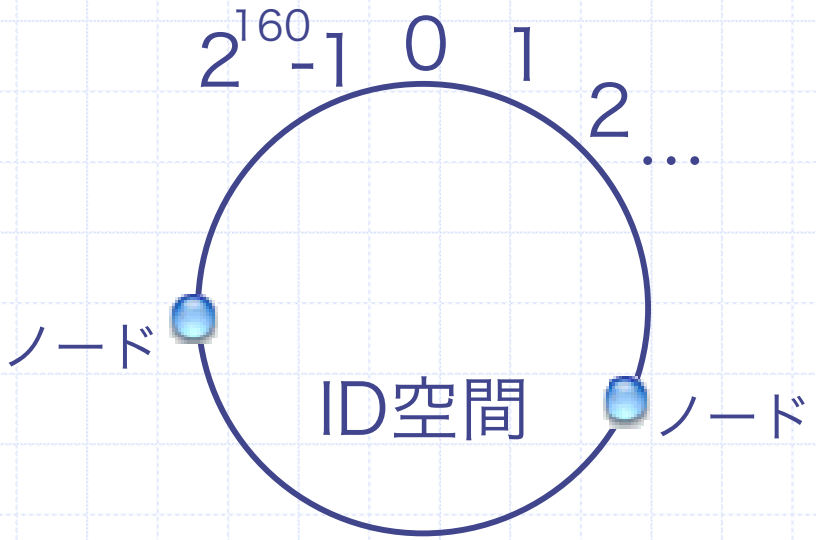
- 各ノードは、一定数の隣接ノード (neighbor) を持つ。
  - どのノードを隣接ノードとするか、および、隣接ノードの数はアプリ依存。例えば 4。
- 検索時、各ノードは問い合わせを全隣接ノードに転送する → flooding
  - TTL（time-to-live, 最大ホップ数）は 7。
  - 返答は、転送経路に沿って返される。

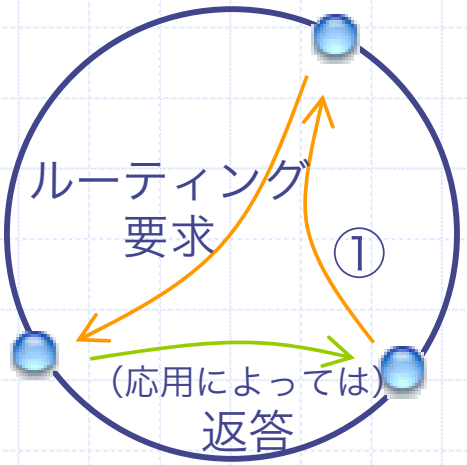


- Gnutella プロトコル 0.6 は、super-peers の概念を採り入れた。

Structured オーバレイの基本

- ノード（計算機）とオブジェクトの両方にIDが振られる。
  - IDはたいてい整数値。160ビットだったり 128ビットだったり。
  - オブジェクト：任意の文字列だったり、ファイルだったり、プロセスだったり。
- ノードは、ID空間中のある範囲を受け持つ。
  - だいたい、ノードのIDと数値的に近い範囲を受け持つ。
- IDを宛先としてルーティングが行われ、その行き着く先は受け持ちノードとなる。

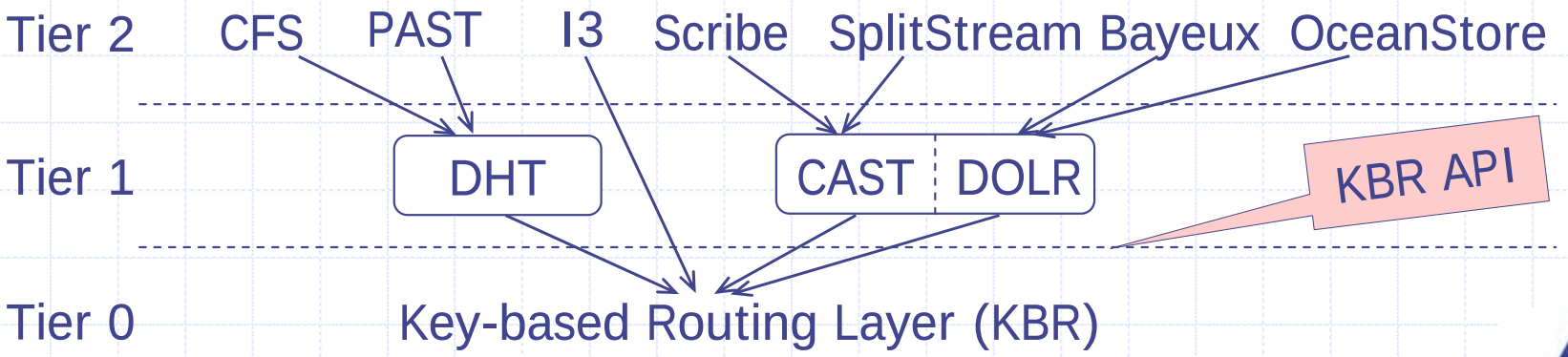




AIST

Structured オーバレイの基本

- 肝はルーティング（アルゴリズム）
  - 資源にかかる負担が、ノード数 n として O(log n)。
    - ルーティング時のメッセージ数など。
    - cf. unstructured オーバレイ上の flooding
  - 様々なアルゴリズムが提案されてきた。
    - CAN, Chord, Tapestry, Pastry, Kademlia, Koorde, Accordion, ORDI, DKS, D2B, Symphony, Viceroy, …
  - Structured オーバレイ上に、いろいろなサービスが載る。
    - 分散ハッシュ表 (DHT), マルチキャスト, メッセージ配送, …



Cited from [Dabek03]

AIST

分散ハッシュ表 (DHT)

- Structured オーバレイ上に載るひとつのサービス。ハッシュ表。
  - put (key, value)
  - get (key)
  - キー・値ペアの削除

- 処理
  - put: keyをキーとしてルーティングを行い、目的ノードにキー・値ペアを保持させる。
  - get: keyをキーとしてルーティングを行い、目的ノードが保持している key に対応する値をもらう。

  - 注) key が ID になっていない場合、ハッシュ値を求めて ID にしておく。例えば（暗号学的ハッシュ関数） SHA1 を通すと、160ビットの ID を得られる。

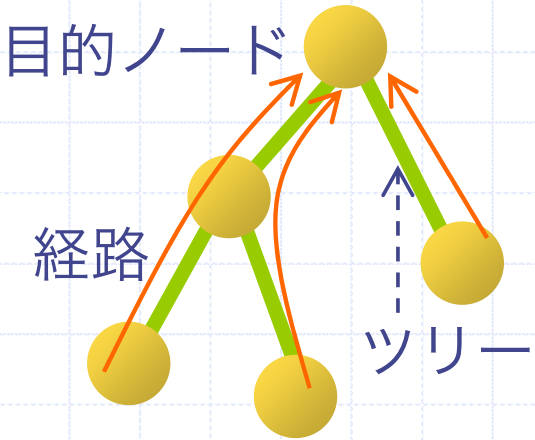
AIST

DHT の応用

- もろもろの名前解決や位置解決
  - ホスト名 → IPアドレス, 名前 → 電話番号, 曲名 → 楽曲ファイルやそのURL などなど。
  - DNS を作ってみました、という研究もある。
  - IP電話ソフト Skype のコンタクトリスト管理に DHT のようなアルゴリズムが使われているとのこと。
    「DHTである」という証拠はない？

Structured オーバレイ上の アプリケーションレベル マルチキャスト (ALM)

- ある ID に対して複数のノードからルーティングすると、それら経路の集合はツリーを成す。
  これを配送木として利用する。
- チャネルが ID で表されるので、多チャネル。
  ≠ ブロードキャスト
- 処理
  - あるチャネルにsubscribeするノードは、チャネルの識別子をキーとしてルーティングをする。
  - 経路上のノードは、1ホップ手前と1ホップ先のノードを、それぞれツリーの子と親として記憶する。
  - 各ノードが、ツリー上の親子関係に沿ってトラフィックを転送する。



AIST

マルチキャストの応用

- 同報通信を行うもろもろの応用
  - グループチャット・音声 / ビデオ会議
  - 放送
  - VPN
    - L2 のブロードキャストに使える。
    - 例??? x-kad: Kademliaというアルゴリズムを応用した VPN
  - 分散処理
    - コードの配布やプロセッサ間同期、ブロードキャスト
- 当然、エニーキャストも載る。
  - グループ中のノードどれかにメッセージ配信
  - 応用：サーバ群の負荷分散

AIST

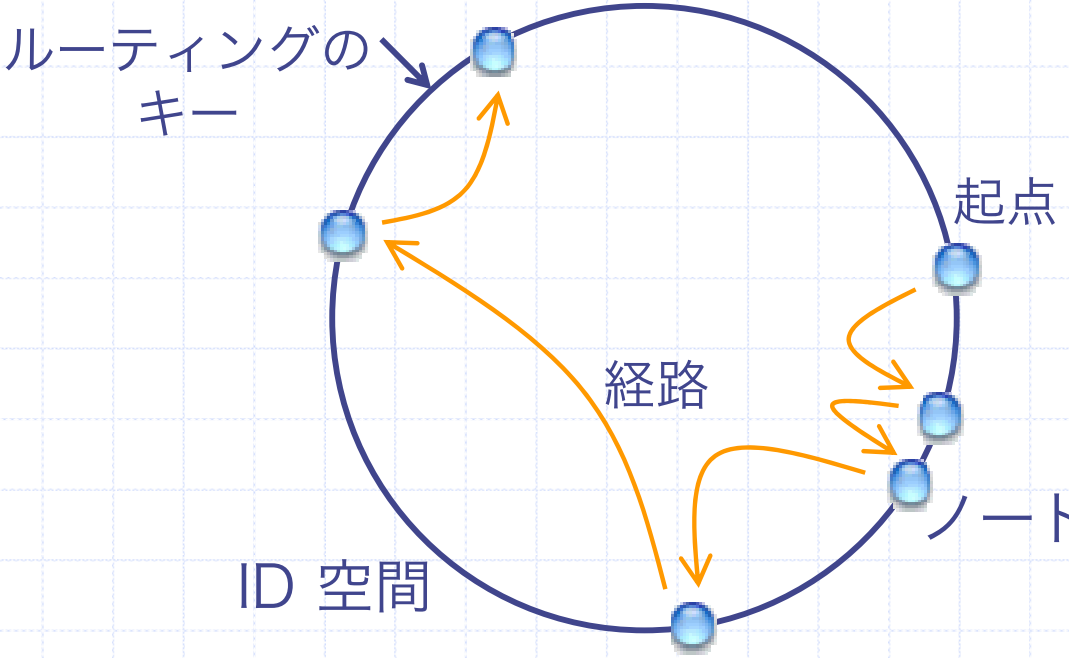
Structured オーバレイの（ルーティング）アルゴリズム

- 宛先 ID に（数値的に）より近い ID を持つノードに、要求をまわしていく。
  - いつかは最も近い ID を持つノードに辿り着く。
- 近づき方の、アルゴリズムごとの違い
  - Chord
    - ID 空間を時計まわりに近づいていく。
  - Pastry, Tapestry
    - ID を、上位桁から順に、b (例 4) ビット単位で揃えていく。
    - Pastry は、最後の 1ホップは違う方法で近づく。
  - Kademlia
    - ID を、上位桁から順に、1 ビット単位で揃えていく。

AIST

Chord

- Consistent Hashing プラス finger table
  - finger table: ショートカットリンク
- Consistent Hashing
  - ID空間を時計まわりに進む。
    - 各ノードは、次のノード (successor) へのリンクを持つ。
  - 下図の場合、5 ホップ
  - オーバレイ上のノード数を n とすると、ホップ数は O(n)。残念！



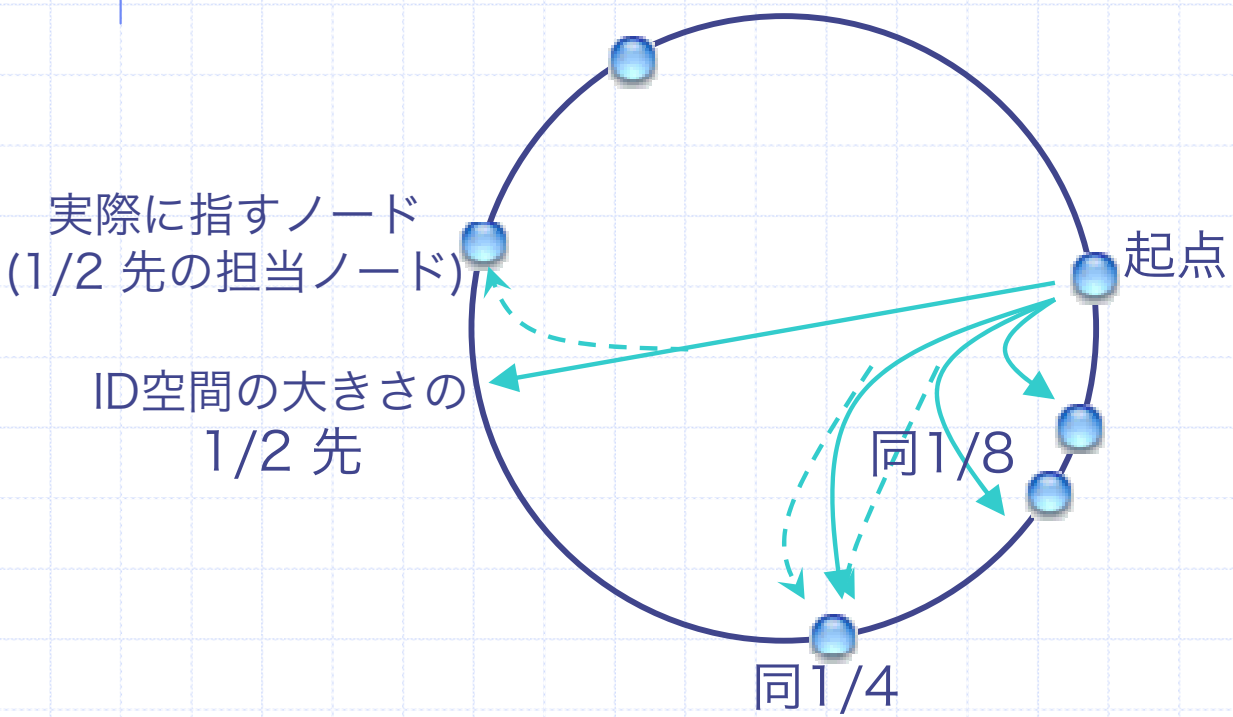
Consistent Hashingでの
ルーティング

AIST

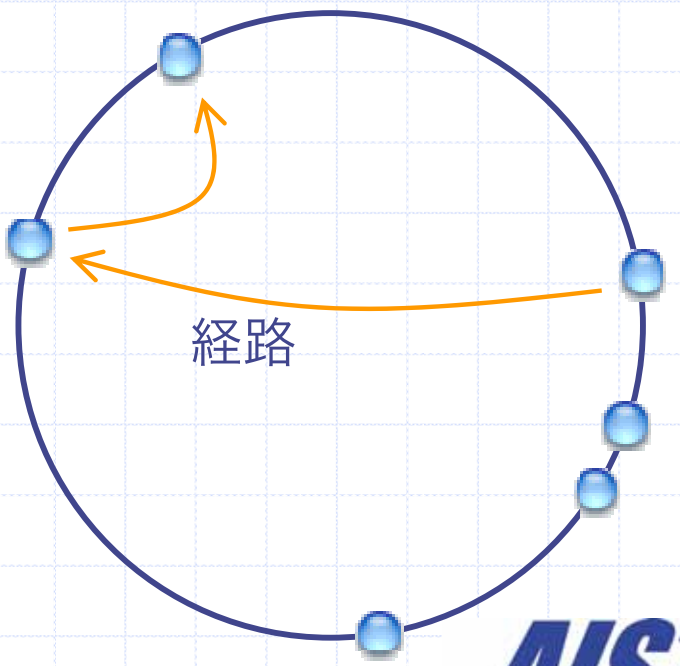
Chord

- finger table
  - ショートカットのためのリンク
  - 下図のルーティングの場合、2ホップ。
  - ホップ数 O(log n)。万歳！

finger table



finger tableを使った
ルーティング



AIST

Pastry, Tapestry

- Plaxtonらの方法 を用いる
  - 上位桁から bビット単位で揃えていく。
- 例
  - b = 4 とする → 16進数の一桁ごとに揃えていく。
  - ルーティングの宛先 ID: 437A (16進数) とする。
    - 本当は、160ビット (Tapestry) や 128ビット (Pastry)。
  - 手順
    - ID が 4XXX であるノードにまわす。
    - ID が 43XX であるノードにまわす。
    - ID が 437X であるノードにまわす。
    - …
  - そういうIDを持つノードが経路表に載っていなければ、次善のノードにまわす。
    - 43XX がなければ、44XX にまわす。

AIST

Plaxtonらの方法の経路表

- 例: ID が 10DF であるノードが持つ経路表
  - この場合、サイズは 16 列 x 4 行。
  - n 行目には、n – 1 桁目まで自分と ID が一致するノードが入る。
  - ID とコンタクト先（IPアドレス等）の組が入る。空の場合もある。

	0	*1*	…	*D*	*E*	*F*
1桁目		自身		DXXX	EXXX	FXXX
2桁目	自身			1DXX	1EXX	1FXX
3桁目				自身	10EX	10FX
4桁目		10D1		10DD	10DE	自身

Pastry と Tapestry の違い

- Pastry
  - 前頁の経路表に加えて、ID が数値的に近いノードの集合 leaf set も保持する。
    - 8ノードだったり 32ノードだったり。
    - 次ホップを決める際、まずは leaf set を見る。
- 違い
  - エントリが空だった場合の、次善のノードの決め方。
    - Pastry: とにかく ID が数値的に近いノード。
    - Tapestry: 表を右に見ていく。自身にぶつかったら、下段に降りる。
  - おまけ：広域分散ストレージ OceanStore のルーティングアルゴリズムは、Tapestry + bloom filter

オーバレイへの参加・経路表の保守

- 参加の際は、自身および他ノードの経路表を更新（作成）する。
- 新ノードの参加や既存ノードの脱退に応じて、各ノードは経路表を保守する。

- ただのルーティングよりは複雑。
  - アルゴリズムごとの詳細は、ここでは割愛。

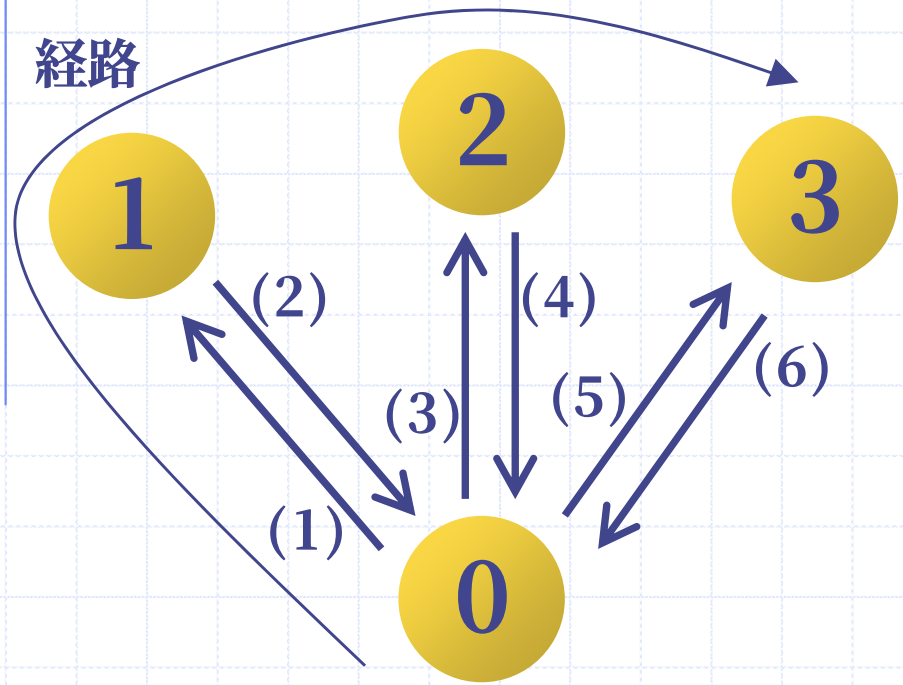
オーバレイへの参加・経路表の保守

- 各アルゴリズムが、プロトコル・手順を規定している。
  - 参加の際は、まず、自ノードの ID を宛先としてルーティングする。
  - 参加の時点で、関係する全ノードの経路表を更新し切ってしまうか否か：
    - 更新し切る
      - Chord （論文Fig.6のアルゴリズム）, Pastry, Tapestry
    - 保守によって、じょじょに更新される
      - Chord（通常のアルゴリズム）, Kademlia
  - 保守
    - 定期的に通信して行う
      - Chord, Pastry, Tapestry
    - 定期的な保守が不要
      - Kademlia: ルーティング目的の普段の通信を手がかりに保守する。

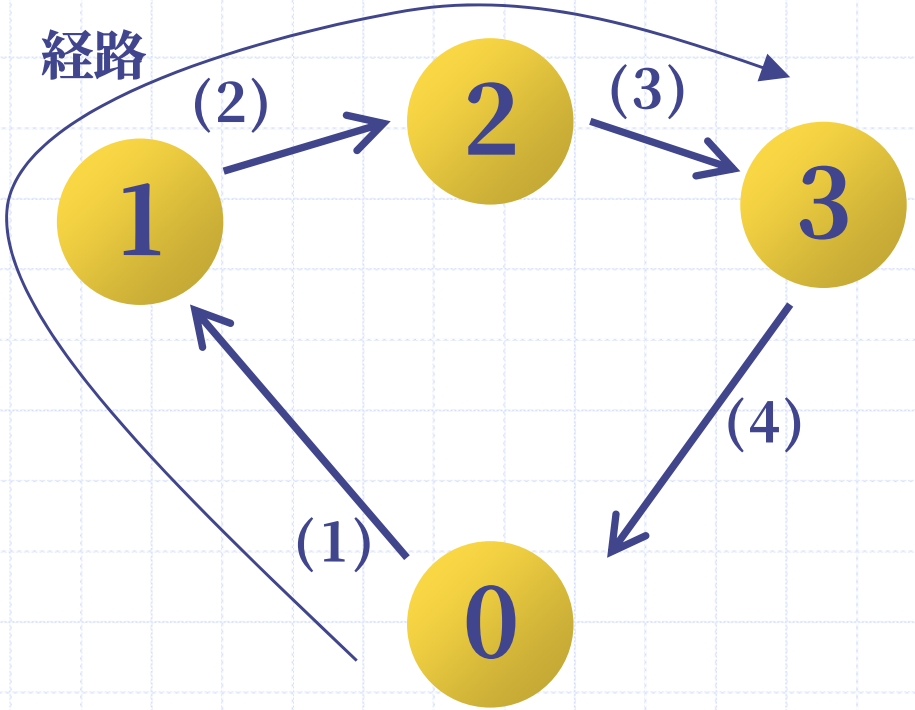
AIST

ルーティングの様式

Iterative ルーティング



Recursive ルーティング



- アルゴリズムと独立だったり、そうでなかったり。
  - Pastry, Tapestry 論文は Recursive が前提だが、どちらも可能。
  - Kademlia では Recursive ルーティングは困難。
- それぞれ、強み / 弱みがある。

AIST

オーバレイについての発展的な話題

- ネットワーク的な近接性の扱い
- 非均質性の扱い
- Complex network の工学応用？
- 研究開発ツール・ソフトウェア

AIST

ネットワーク的な近接性の扱い

- 「素早く見つけたい」「速くダウンロードしたい」「途切れなく視聴したい」「確実に届けたい」…
- ネットワーク的に近い方が
  早い / 速い / 確実。
  だから、近い相手を選びたい。
- 近いとは
  - 遅延が小さい and/or 帯域幅が広い
  - ID の数値的な近接と混同しないように注意する。

ネットワーク的な近接性の扱い

- Unstructured オーバレイの方が得意だと言われている。
  - 隣接ノードや通信相手を選ぶ際の制約がないから。
- Structured オーバレイに採り入れることも可能。
  - 採り入れ方の分類
    - Proximity Neighbor Selection (PNS)
    - Proximity Route Selection (PRS)
    - Proximity Identifier Selection (PIS)
  - アルゴリズムによって、それぞれとの親和性が異なる。
    - Pastry: 前述の表に加えて、neighborhood setを持つ。
      最初の論文では「実装してない」とのこと。
    - Tapestry: 経路表を複数枚持つことで、PNS & PRS が可能。
- Iterative と Recursive ルーティングでは、近接性の影響が異なる点にも注意。

AIST

近さを知る方法

- 基本的には、計測。
  - 遅延 & 帯域幅
- 推定する手法も提案されている。
  - 一部分の計測結果をもとに、ノード間距離を推定する。
  - Vivaldy
    - ２次元空間 + 高さにノードをマップ
  - Lighthouse

非均質性の扱い

非均質性: heterogeneity
⇔ homo-…

- オーバレイに参加するノードは様々。
  - 処理能力
  - ストレージの大きさ
  - ネットワーク帯域幅
    - 例: xDSL の上りは狭い。P2P で放送をする際に…
  - 通信の信頼性
  - ファイアウォール内 / 外
  - 継続して稼動する時間の長さ
    - 例: 常時電源 ON, たまに起動, …
- 全ノードにまったく同じ処理を求めると、最も低い / 狭い / 弱いところに全体の性能が制約されかねない。

AIST

非均質性への対処方法

- スーパーノード / ピアの導入
  - 限られたノードでオーバレイを構成する。
    他のノードはスーパーノードの提供するサービスを利用する。
  - cf.
    - ファイル共有ソフト Kazaa の Gnutella プロトコル 0.6
    - JXTA 2.x
  - いわば、0 か 1
- （全ノードの）適応 (adaptation)
  - 能力に応じて、隣接ノード数などを変える。
    - cf. HeteroPastry
  - いわば、連続量
  - Unstructured オーバレイへの導入の方が素直ではある。

AIST

Complex Network の工学応用？

- Complex Network
  - regular グラフでも random グラフでもないグラフ
  - Small World Network
    - Low diameter
      - Diameter (直径): 任意の２ノード間の距離の平均
    - High clustering
      - clustering coefficient (クラスタリング係数): あるノードの隣接ノード群がつながっている割合
    - cf. Milgram の手紙転送実験
  - Scale-free Network
    - 次数（枝数）が k であるようなノードの出現頻度が、k ^ -γ に比例。⇒ べき乗則
    - つまり、ハブが存在する。
    - cf. ウェブページのリンク数

AIST

Complex Network の工学応用？

- Complex Network の性質
  - 次数が小さいのに直径（≒ ホップ数）が小さい。
- ネットワーク分析の分野で注目されているが、作る際（工学）にも活かせるだろう。
  - 例
    - 流通ネットワークの構築：ハブの設置
    - Structured オーバレイに適用：Symphony
      - ショートカットパスの数を、ID 空間サイズに依らず、一定とする。⇔ Chord
      - それでもホップ数はわりと小さいまま。

Complex Network

- Gnutella ネットワークは
  以前から Scale-free Network ？
  - 多分、super-peers 導入前。それにもかかわらず。

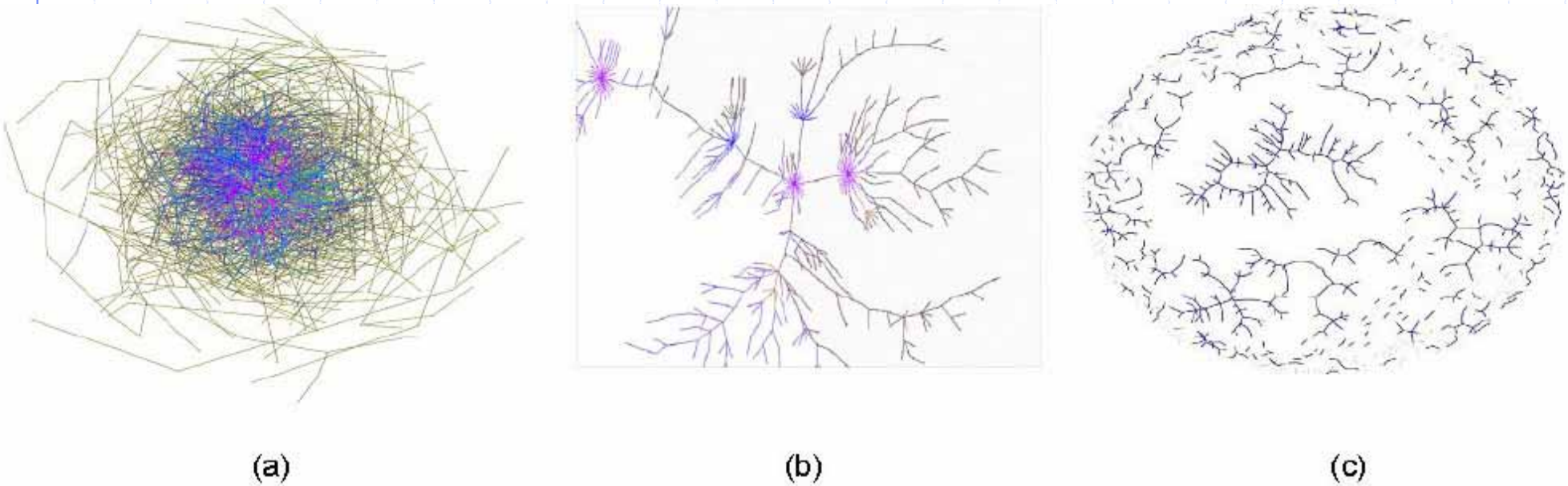


Figure 13. Left: Topology of the Gnutella network as of February 16, 2001 (1771 peers); Middle: Topology of the Gnutella network after a random 30% of the nodes are removed; Right: Topology of the Gnutella network after the highest-degree 4% of the nodes are removed.

Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”, Proc. MMCN’02, 2002. の図13

AIST

研究開発ツール・ソフトウェア

- p2psim
  - ルーティング方式のシミュレータ
  - Structured オーバレイのアルゴリズムを多数提供
    - Chord, Accordion, Koorde, Kelips, Tapestry, Kademlia
- MACEDON
  - オーバレイアルゴリズムの研究プラットフォーム
    - structured / unstructured 問わず。
  - C / C++ に似た専用言語でアルゴリズムを記述し、それをMACEDONのコンパイラで処理し、動作する C++ コードを得る。⇒ コード量が少なくて済んでいる。
  - DHT アルゴリズムとして Chord, Pastry を提供
    - 本来 160 / 128 ビットであるID が、なぜか 32 ビット。
- Mace
  - MACEDON の後継プロジェクト

AIST

研究開発ツール・ソフトウェア

- DHT のライブラリ
  - Bamboo DHT
    - PlanetLab 上で運用されている。⇒ OpenDHTプロジェクト
    - アルゴリズムは Pastry ベース。
    - Javaで書かれている。
  - Chimera, Tapestry
    - Tapestry の研究グループ自身による実装。
    - Tapestry は Java で書かれている。
    - Chimera は、Tapestry のコンパクトな C 実装。
  - FreePastry
    - Java で書かれている。
  - Khashmir
    - Python で書かれた Kademlia 実装。
    - BitTorrent （本家クライアント）が使っている。
  - Kenosis, SharkyPy, DKS, OPeN, …