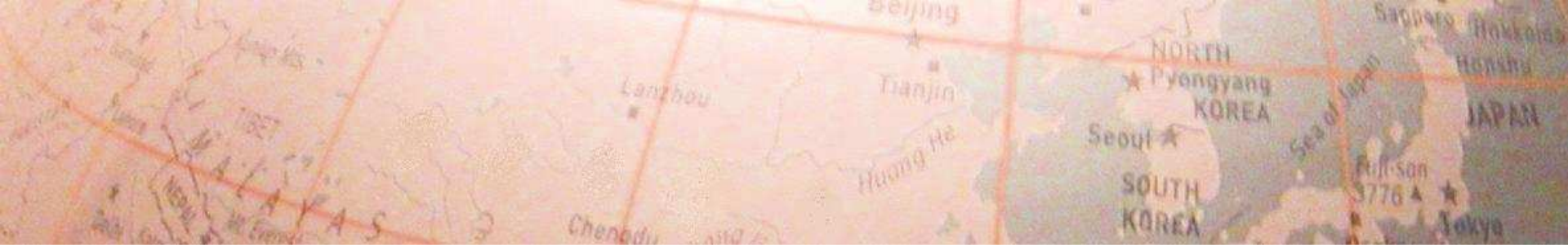


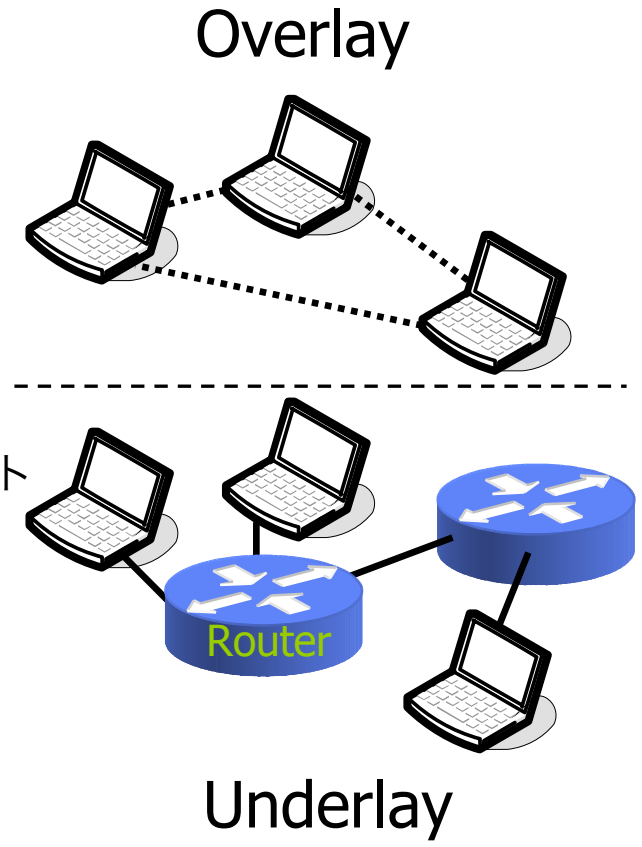
構造化オーバレイでの一括フォワーディング

首藤一幸 NICT
中尾彰宏 東京大学 / NICT

Overlay Weaver

オーバレイネットワーク

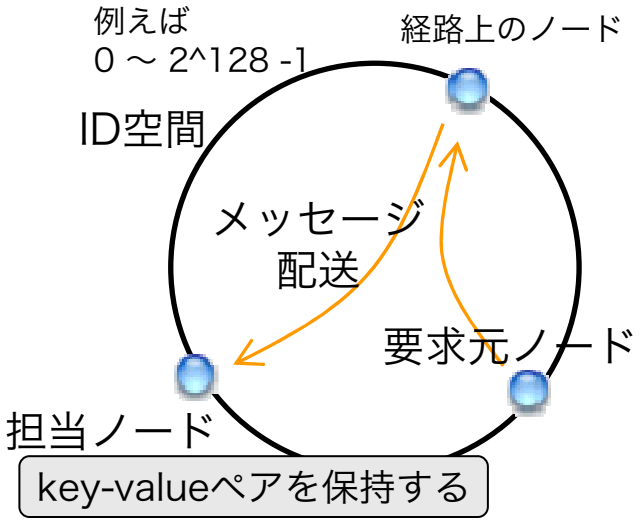
- 下位ネットワークとは独立した
  トポロジを持つ上位ネットワーク
  - 電話網上のインターネット
  - インターネット上のVPN, CDN
  - MBone, 6bone, …
  - ファイル共有ソフト
    - FastTrack, Gnutella: 100万ノード以上
  - ライブストリーミング: アプリ層マルチキャスト
    - ウタゴエ社 UG Live 等



- 狙いの例
  - サーバなし / pure P2P → 要オーバレイ
    - スケーラビリティ
    - 頑健さ

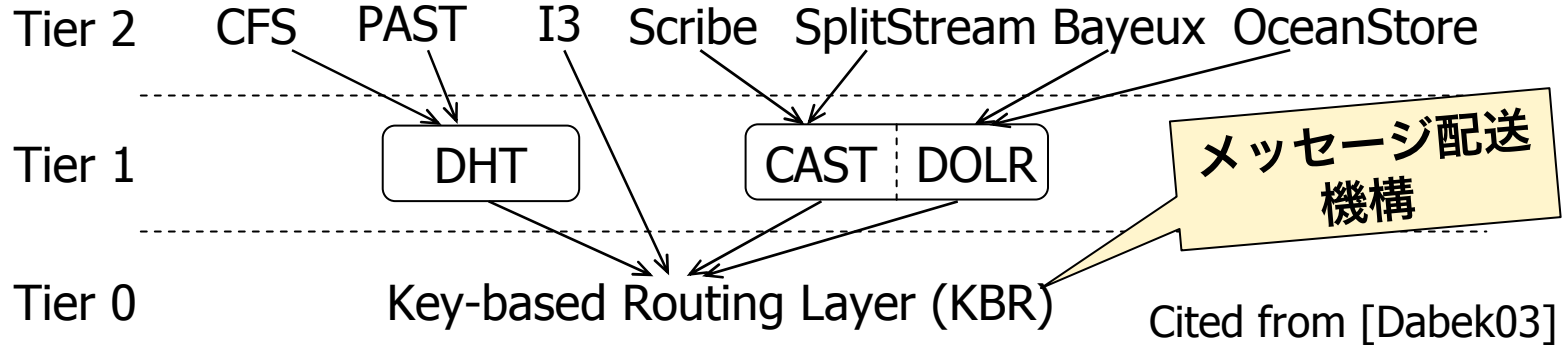
構造化オーバレイ

- 構造化: トポロジ / 隣接ノード関係にアルゴリズム上の制約 → 高効率
- 肝は メッセージ配送機構 (ルーティング)
  - ID の担当ノードに配送。
  - 配送機構の上に、分散ハッシュ表 (DHT) や マルチキャストなどを構成可能。
  - DHTへの put
    key (ハッシュ値) をあて先として配送、到達した担当ノードにkey-valueペアを保持させる。



注: アルゴリズムによっては、
ID空間を円環としてとらえることが適切とは限らない。

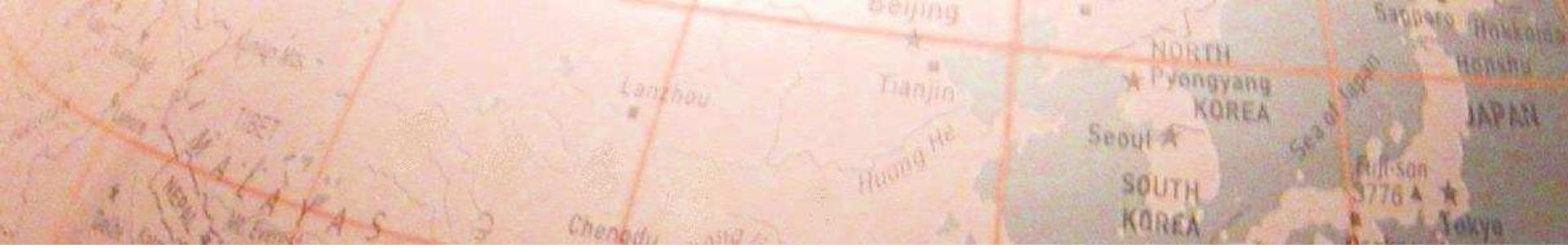
Dabek らによる 構造化オーバレイの抽象化



Cited from [Dabek03]

構造化オーバレイ: 応用

- 分散ハッシュ表
  - Tracker-less torrents
    - 集中的なサーバが不要な BitTorrent (ファイルの分散配布プロトコル/ソフト)。
  - DNS
    - “A Comparative Study of the DNS Design with DHT-Based Alternatives”, Proc. INFOCOM 2006.
  - その他、様々な名前解決、ID解決
    - 電話帳, ファイル名/氏名/曲名 → URL, …
- ID / Locator 分離
  - HIP (RFC 4423) の解決を DHT で。
    - “HIP DHT interface”, draft-ahrenholz-hiprg-dht-02, Internet-Draft, Jan 2008.
  - i3: a rendezvous-based comm. Abstraction
    - “Internet Indirection Infrastructure”, Proc. SIGCOMM 2002.
  - 構造化オーバレイのメッセージ配送で ID ベース通信
    - “オーバレイネットワークにおけるID/Locator分離機構”, 情処研究報告, 2008-DPS-134, 2008.

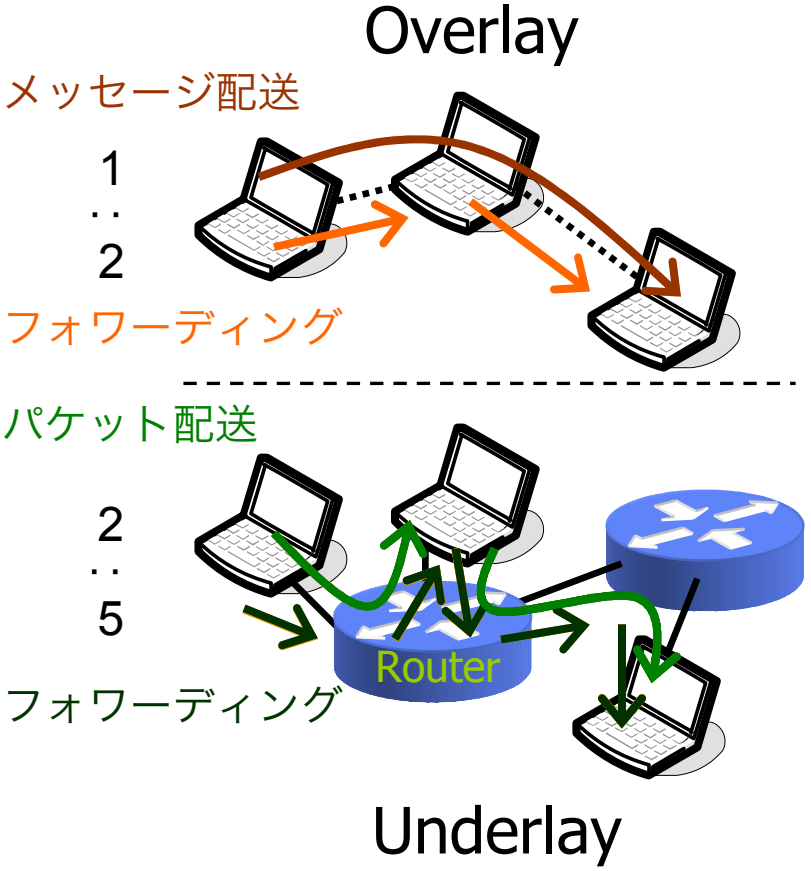


一括フォワーディング

collective forwarding

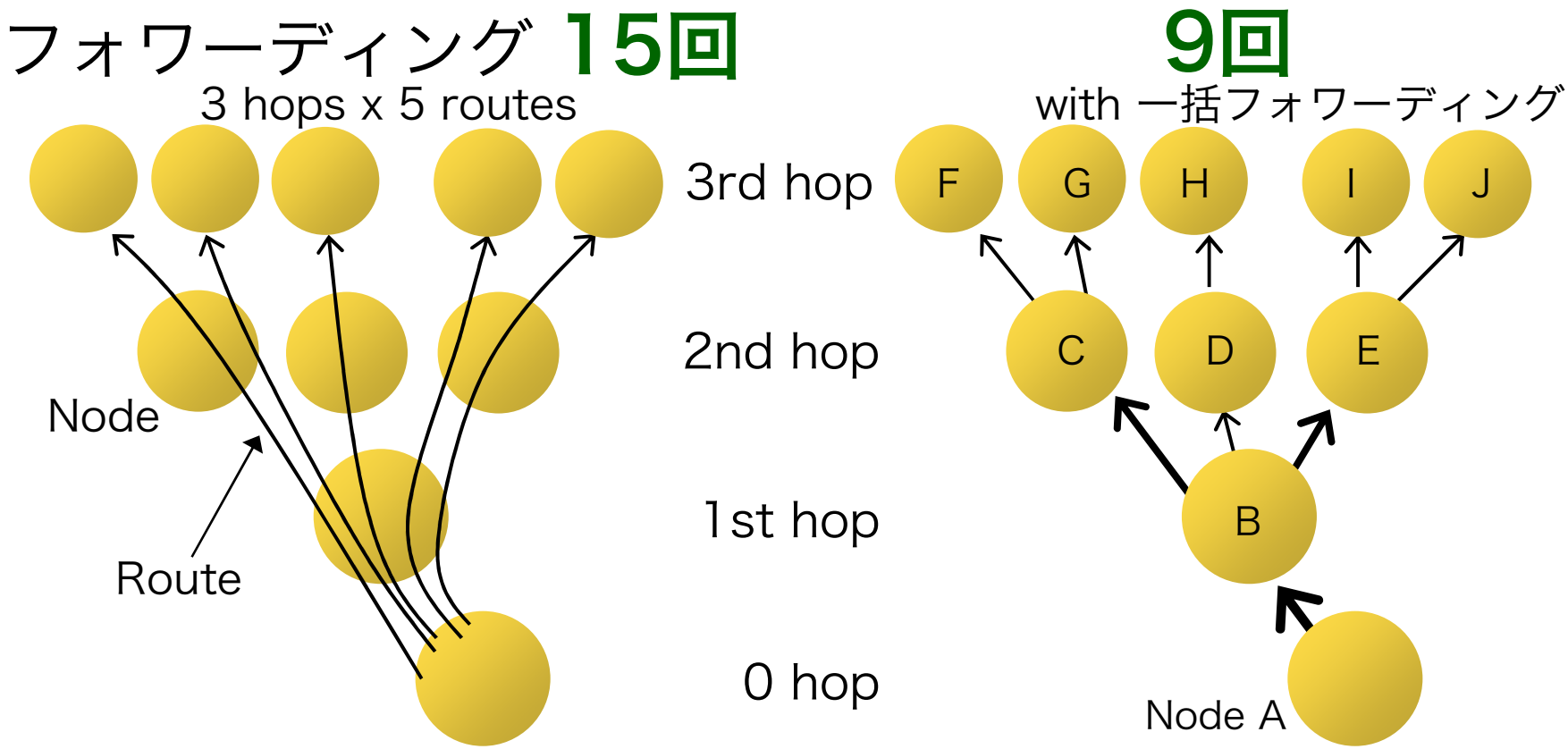
対象とするオーバレイの問題

- オーバレイでのメッセージ配送は、時間がかかる。
  - DHTに対する10,000回のgetに40〜700秒 (論文4.3節, 図6)
- オーバレイは、アンダーレイに負担をかける。
  - メッセージ配送1回が…



一括フォワーディング

- 複数のメッセージ配送要求をまとめて扱い、
  次ホップが同一であるメッセージ群を
  まとめてフォワードする。
  - 効果: 10ずつまとめて、フォワーディングの回数を ～12% まで削減



効果

- オーバレイに対して
  - **スループット向上**
    - メッセージをまとめて扱う
      → 通信の単位が大きくなる
      → 複数メッセージの並行処理
      (注: 提案手法以外でも同様の効果は得られる)
  - ノードの**処理軽減**
    - フォワーディング回数削減 → フォワードごとの処理、例えばメッセージのエンコード / デコードが減る。
      - 例: Java のobject serializationは負荷/時間が大きい
- アンダーレイに対して
  - **通信回数の削減 → 負荷軽減**
    - cf. ルータの性能はpps (packets per second)で表される。

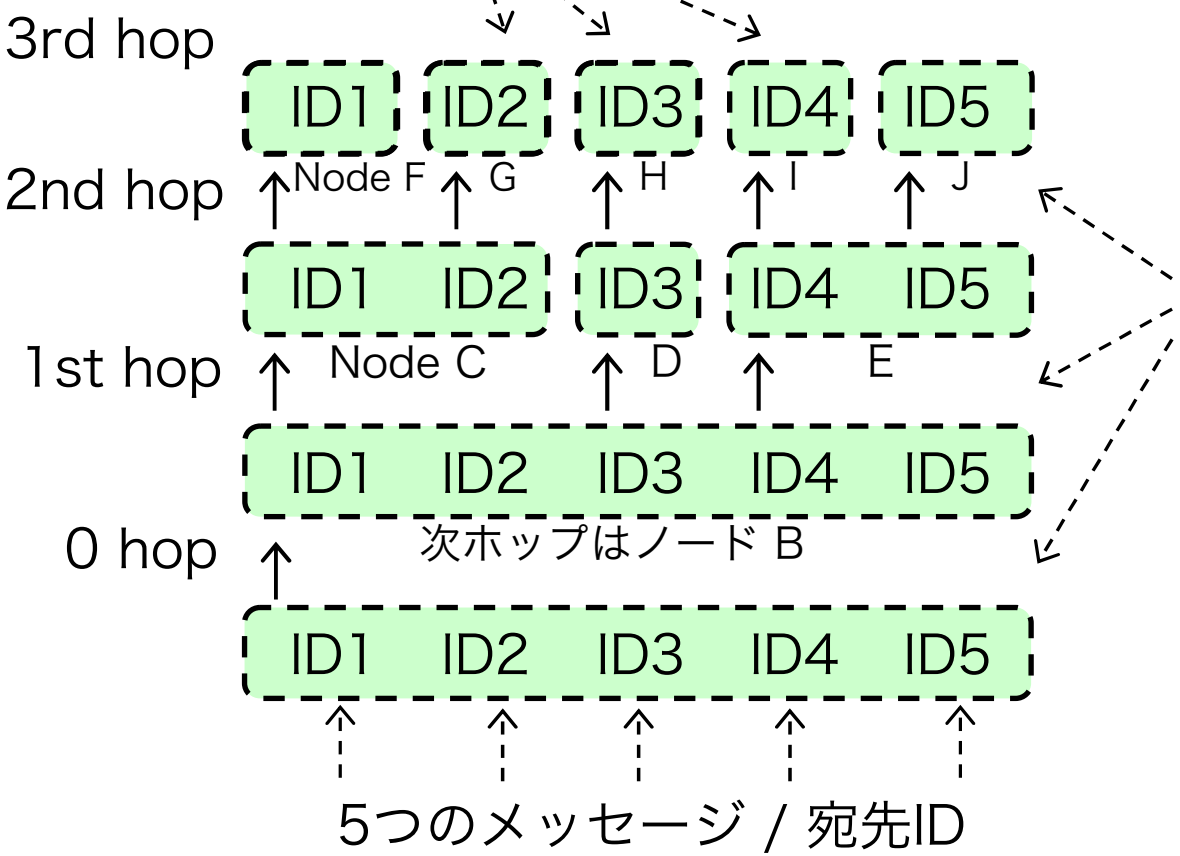
測定対象

効果を得られる状況 の例

- DHT に対する大量の put / get
  - DNS
    - “A Comparative Study of the DNS Design with DHT-Based Alternatives”, Proc. INFOCOM 2006.
  - RFID 管理システム
  - 数百万かそれ以上のデータアイテム数。
  - 少なくとも、
    初期稼動時やバックアップ時には、全体 or ある部分を一度に put / get する必要が生じる。

各ノードでの フォワーディング処理

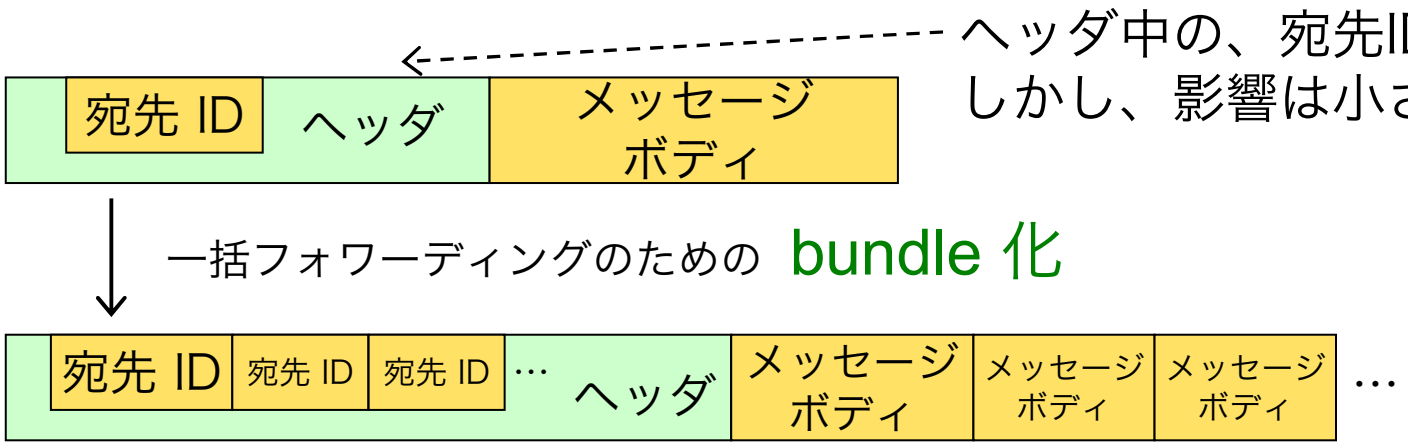
- 複数のメッセージ / 宛先ID をまとめて扱う。
  → bundle



1. 次ホップを決定
2. 次ホップに応じて bundleを分割
3. bundleごとに 次ホップに フォワード

効果についての議論

- オーバレイでのフォワーディング回数、アンダーレイでの通信回数は減るが、トラフィックが減るわけではない。
- 提案手法で small group マルチキャストを行う場合は、話は異なる。
  - 事前準備不要、送信のたびに宛先ID 群を指定、というマルチキャスト。



最初の bundle 化

- いったん一括フォワーディングが始まれば、bundle は 機械的に 分割される一方。

- では、最初の bundle をどう構成するか？
  - 1万の get 要求があったとして、
    1万メッセージをくっつけてしまうのか？
    - ID 128 ビットとして、1万 IDで 156 KiB 以上。
    - それは問題が多い。遅延, ロス率, …

  → 最初から適当なサイズに bundle化しておく。

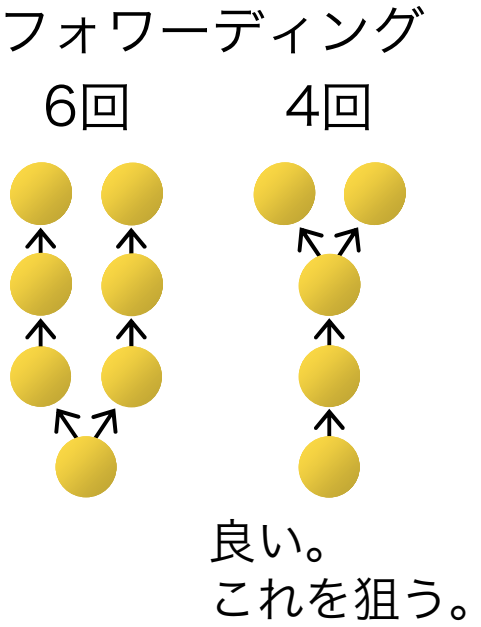
- 決めるべきこと
  - サイズは？
    - 注: サイズとは bundleに含める宛先 ID の数
  - どういう基準で組にする？
  - いつ？

最初の bundle 化: サイズ

- サイズは？
  - 効率のためには、大きくしたい。
    - 大きい方が、フォワーディングの回数を減らせる。
  - 抑えざるを得ない。
    - アンダーレイに信頼性がない場合、メッセージ長を MTU 未満に抑えた方がよいだろう。
  - **今回は 10** として、効果を測定した。
    - 今回の実験条件では、
      bundle 長は数百バイト < 実験環境の MTU
    - どう決めるべきかは、open problem。

最初の bundle 化: 基準

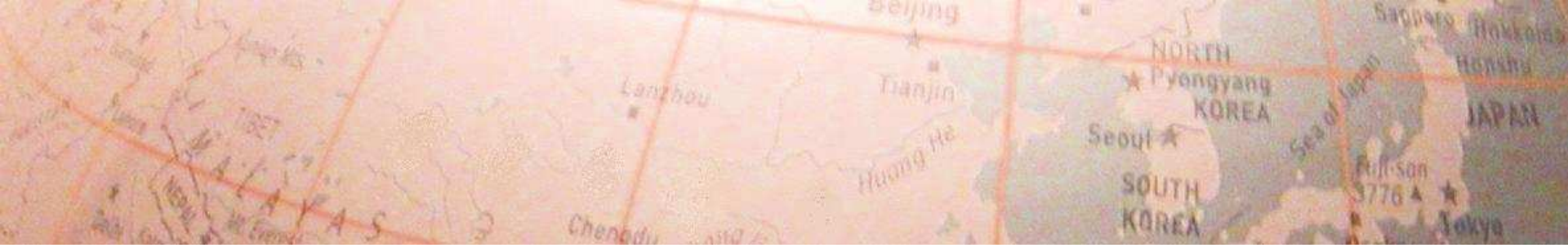
- どういう基準で組にする？
  - なるべく経路が重複するような宛先ID どうしを組にする。
    → 「近い」ものどうしを組にする。
    - ID間の数値的な距離
      例: ID 3から4への距離と、4から3への距離
      - Chord　1　2^160 – 1
      - Kademlia　7　7　← 3 XOR 4



  - すなわち、クラスタリング。
    - ただし距離が非対称なので、知られた手法は使えない (?)。
    - 今回採用した手法は、
      単純、計算量大 (度外視)、良い結果を期待できる手法。
      - 一括フォワーディングの効果を見ることが主眼。

最初の bundle 化: タイミング

- いつ？
  - 効率上、最善のタイミング: 最遅
    - 要求元での最初のフォワーディングまでに。
    - 次ホップを決めた後で bundle 化できる → 最善
  - 今回
    - オーバレイへのメッセージ配送要求の手前で、bundle サイズが上限を超えないように調整。
      - DHT の場合、数千数万のキーを一度に put/get するのではなく、例えば 10や20ずつ put/get する、ということ。
  - オーバレイ (のノード) にやらせるべきではない、と工学的に判断した。
    - クラスタリング → 処理が重い, 手法間にトレードオフがありやり方が自明でない
    - 効率はほとんど変わらないだろう。



効果

通信回数の削減
メッセージ配送の所要時間 短縮

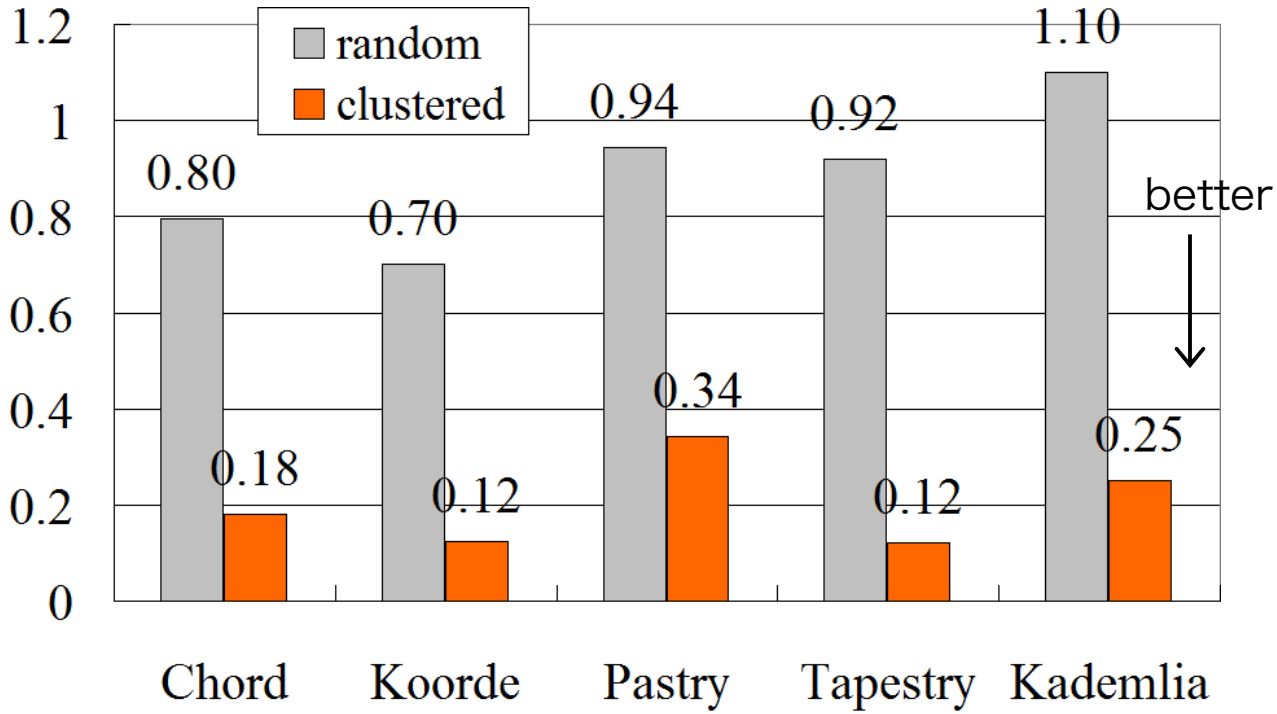
効果の計測

Overlay Weaver

- 実験
  - Overlay Weaver の分散環境エミュレータを用いて
  - PC 1台の上で
  - 1,000 ノードを動作させた。
- 測定対象
  - **通信回数**
    アンダーレイでのパケット配送の回数
  - **メッセージ配送の所要時間**
    大量 get に要した時間
- 条件
  - 一括フォワーディングを行う場合、初期 bundle サイズは 10。
  - put, getを行うノードはシナリオ生成時に乱数で決定。
  - ルーティングアルゴリズム Chord, Koorde, Pastry, Tapestry, Kademlia と、反復/再帰フォワーディングの全組み合わせに対して、計測。
- 実験環境
  - Overlay Weaver 0.8.7
  - x86 用 Java SE 5.0 Update 15, HotSpot Server VM
  - x86-64 用 Linux 2.6.25
  - 2.8 GHz Pentium D

通信回数
一括フォワーディングなしに対する通信回数の比

- 1,000 ノードに対して
  5万の key-value ペアを put、その後 get。
- アンダーレイでのパケット配送回数 → 通信回数を計測。
  - put, get 期間中の通信回数。オーバレイ構築中は除く。

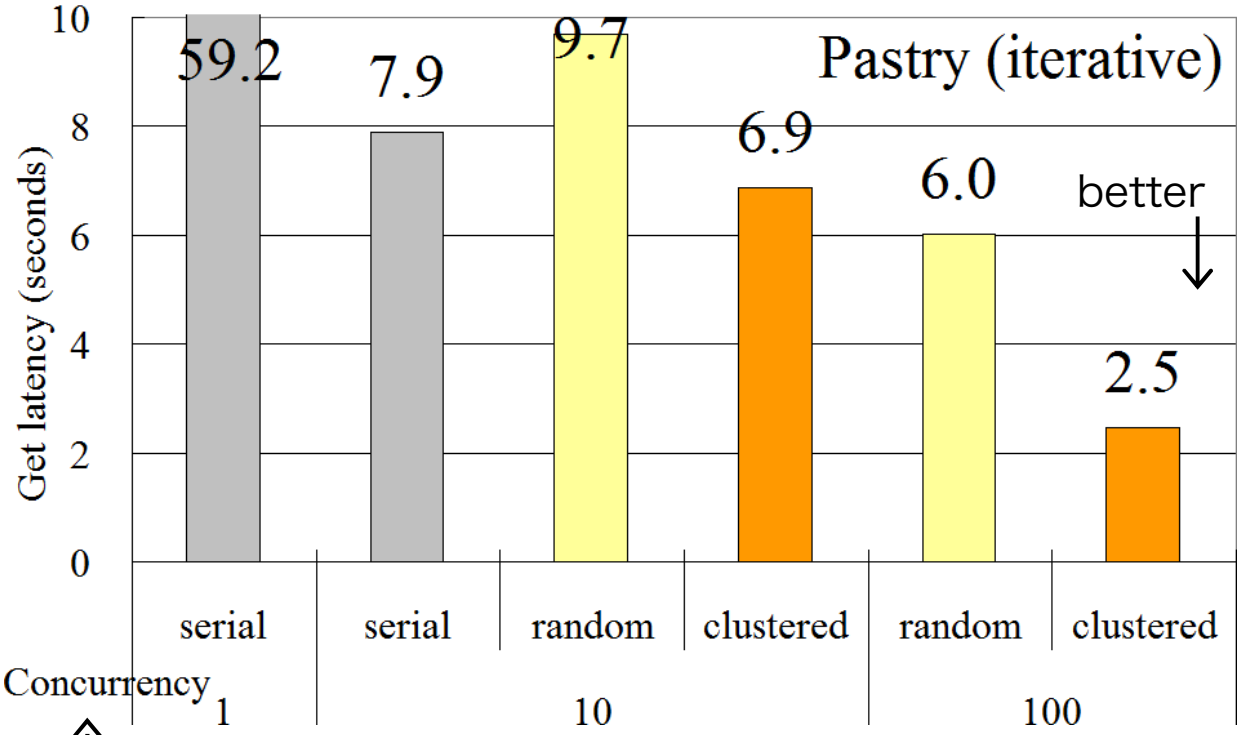


注: グラフは反復様式の場合のもの。

- 最初のbundle化
  - serial: 一括…なし
  - random: 無作為
  - clustered: クラスタリング
- 考察
  - 定常的な通信があるため、0.1 まで下がり切らない。
  - **理論的な限界 0.1 に近い 0.12 まで減っている。**
  - Kademlia は k-bucket満杯が起こり易くなった。

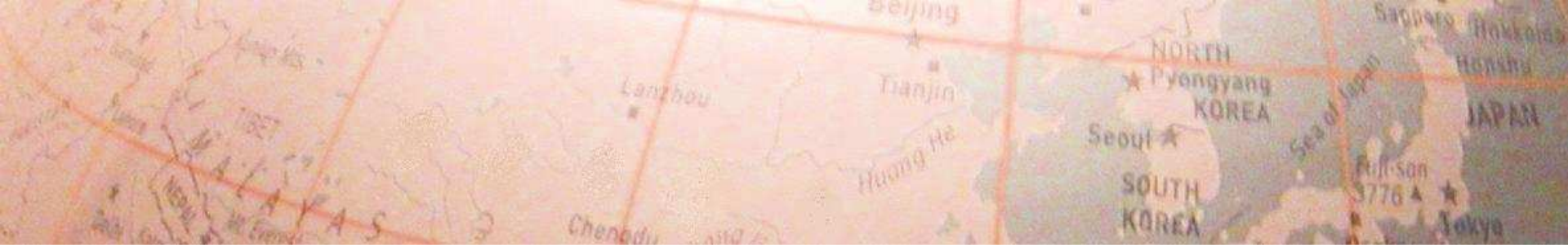
メッセージ配送の所要時間
DHT からの 大量 get の所要時間

- 1,000 ノードから 1万の key-value ペアをget。
- ノード間の通信に 1ミリ秒の遅延を付加。
- 所要時間を計測。



- 考察
  - **10 並列で 10倍近く速くなっている。**
  - 10並列: 一括…なしとclusteredはとんとん。7.9秒 vs. 6.9秒
  - 一括…で 10接続 → **100並列とすると、さらに数倍早くなる。**6.9秒 → 2.5秒

- Concurrency
  - オーバレイ上で同時に処理され得るメッセージ配送 (get) 要求の数
  - bundleサイズ × 接続数


Beijing
Lanzhou
Tianjin
Huang He
NORTH
Pyongyang
KOREA
Seoul
SOUTH
KOREA
JAPAN
Honshu
Tokyo

関連研究とまとめ

関連研究: 手法についての議論

- フォワーディング自体を省いてしまう。
  - 宛先 ID に対する担当ノードを算出して、直接、担当ノードに対してメッセージ配送 (put, get) してしまう。
  - 問題点
    - コストが下がるかどうか、定かではない。
      - フォワーディングを省くためには、オーバレイ上のノード群について ID を把握する必要あり。
        → ID 把握のコストは？ 特に、churn → コスト高。
    - 構造化オーバレイの強みを捨てることになる。
      - 強み: 各ノードが持つ情報を O(log N) 以下に抑えられる。
- ノード上で動的に bundle 化を行う。
  - メッセージの待ち合わせ → 配送の遅延が増す。

関連研究

- Xcast
  - Small Group Multicast のためのプロトコル。RFC 5058。
  - ヘッダに複数の宛先 IP アドレスを含む。
  - Xcast 対応ルータは、一括フォワーディングと同様、次ホップが同一となる複数の宛先について、ただ１つのパケットをフォワード。
  - 相違点
    - IP 層のプロトコル。マルチキャスト専用。
    - deploy の容易さや効率については、要議論。そもそも目的が違うし…
    - クラスタリングによるフォワーディング回数削減、といった最適化の余地はない。
      - 宛先が IP アドレスであるため、複数の宛先に対して経路の重複具合いを見積もることは困難。

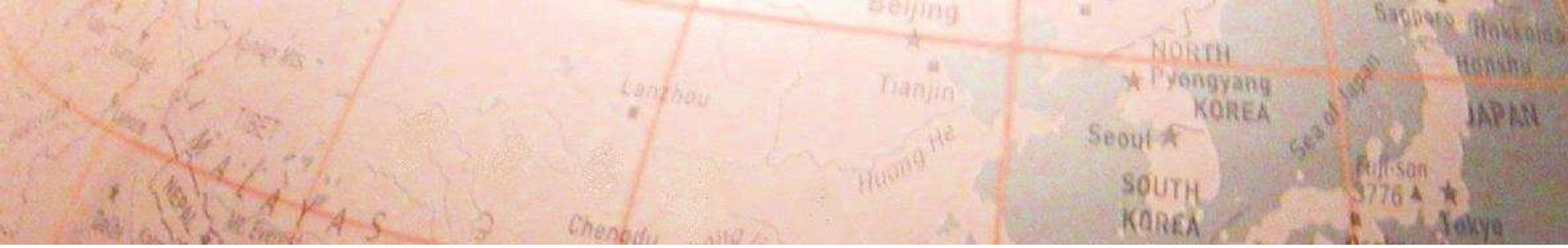
関連研究

- 単一のメッセージ配送を効率化する手法
  - 遅延を小さくする。
    - Proximity routing
      ルーティングの際、ネットワーク的な近さに配慮すること
  - 経路長を短くする。
    - 1-hop DHT
    - Amazon's Dynamo や JXTA の loosely-consistent DHT も、1ホップで済ませようという試み。
  - 相違点
    - 提案手法は、
      複数の配送要求をまとめて効率化する。

まとめ

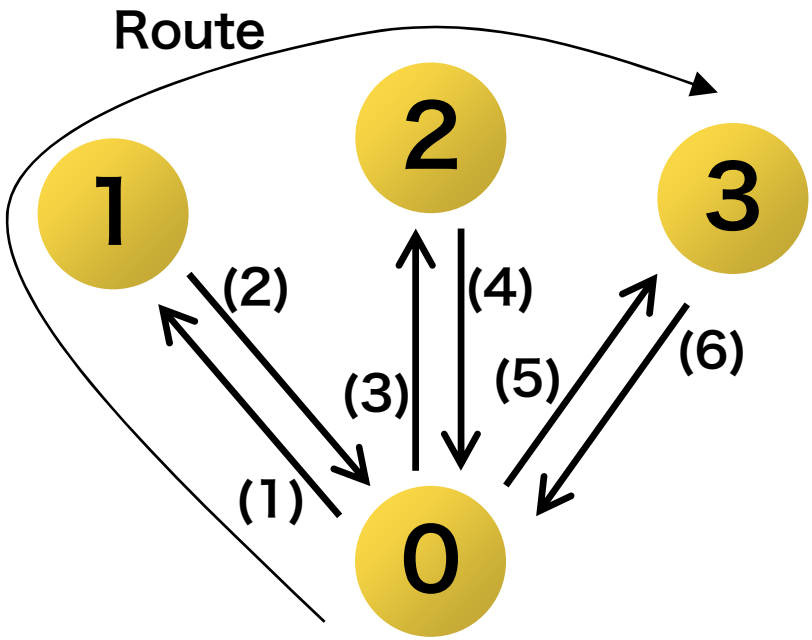
- **一括フォワーディング**を提案
  - オーバレイにて
    複数のメッセージ配送要求をまとめて扱う
    - オーバレイのスループット向上、処理軽減
    - アンダーレイの通信回数を削減 → 負荷軽減

- 効果
  - 通信回数は 34% ～ 12% まで
  - DHT での get 所要時間は 13.0% ～ 9.7% まで減った。
    - 100並列とすると、7.03% ～ 3.12% まで短縮できた。

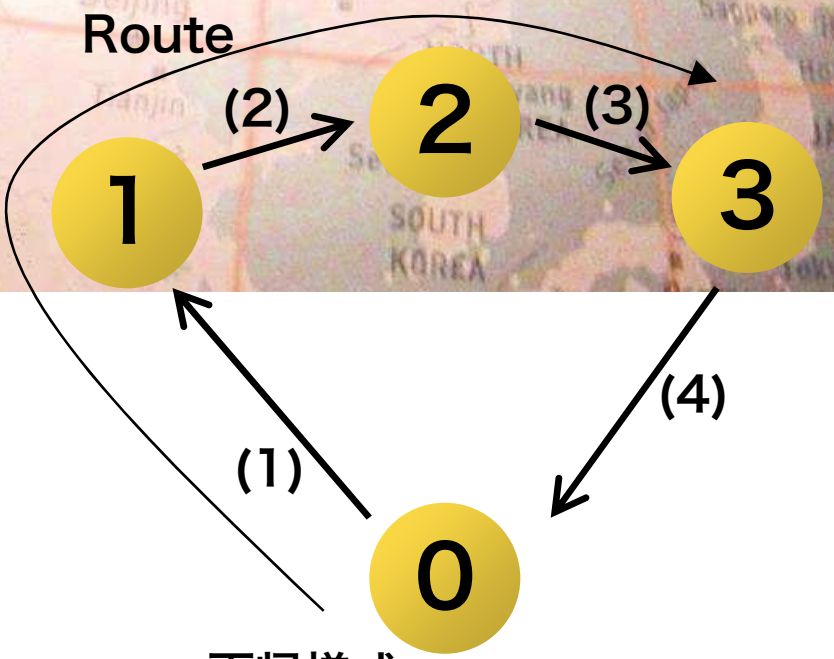


Spare slides

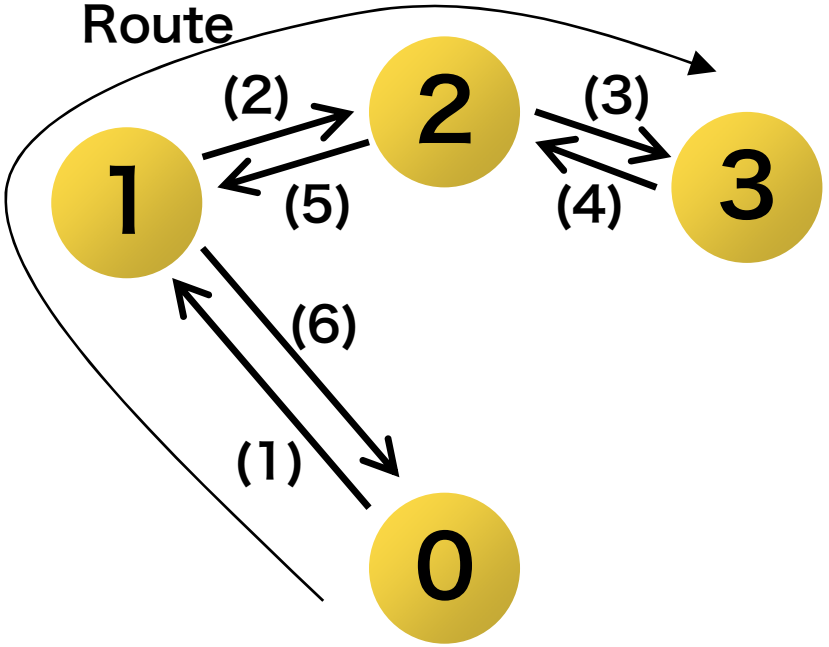
フォワーディング様式



反復様式
Iterative forwarding



再帰様式
Recursive forwarding



Overlay Weaver の構造