2008－OS－109 (1)
2008／8／6

構造化オーバレイでの一括フォワーディング

首藤 一幸† 中尾 彰宏‡

構造化オーバレイにおいて，IP網などアンダーレイの負荷を軽減し，同時にメッセージ配送を効率化する手法，一括フォワーディングを提案する．オーバレイに対して多数のメッセージ配送を要求する場合，例えばDHTに対して多数のput，getを行う場合に，配送経路上のノードが複数の要求をまとめて扱うことでオーバレイでのフォワーディング回数を減らす．これによって，オーバレイでのスループットとノードの処理負担が改善され，アンダーレイでの通信回数も減る．1,000ノードからなるDHTに対してget要求を10ずつ束ねて行うことで，通信回数は34%～12%まで，所要時間は13.0%～9.7%まで減った．

Collective Forwarding on Structured Overlays

Kazuyuki Shudo† Akihiro Nakao‡

This paper presents *collective forwarding*, a technique to alleviate an underlay network such as an IP network and improve efficiency of message delivery on structured overlays. The technique improves communication throughput, reduces loads of nodes on an overlay, and reduces the number of communications on an underlay by bundling a number of delivery requests on an overlay. It shows its performance in case that an overlay delivers a large number of messages, for example we put or get many items on a DHT. It reduced the number of transmissions to 34% ～ 12% and the time to get items to 13.0% ～ 9.7% by bundling each 10 get requests on a DHT with 1,000 nodes.

1 はじめに

構造化オーバレイ (structured overlay) は，中心となるサーバ的なノードのない，全ノードが対等な非集中かつ自律的な分散環境において，IDに基づいたメッセージ配送を行う仕掛けである．メッセージ配送機構 (key-based routing) の上に，分散ハッシュ表 (DHT) やマルチキャスト，エニーキャストといった機能を構成できる [7]．スケーラビリティや耐故障性の高さから基盤的な分散システムの基礎技術として期待され，たとえばDNSへの適用検討 [13] やコンテンツ配信への実際の応用 [1, 2] が始まっている．

N ノードからなる構造化オーバレイでのメッセージ配送には，一般に，$O(logN)$ ホップのフォワーディングを要する．その各ホップがアンダーレイ，たとえばIP網でのパケット配送であり，どうしても，アンダーレイでの一対一直接通信よりも，一回あたりの所要時間やアンダーレイの負担は大きくなる．

ここで，複数の宛先IDに向けてメッセージを配送する状況を考える．もし，それぞれの経路に重なりがあれば，次ホップへフォワードするメッセージを1つにまとめることで，総フォワーディング回数を削減できる．これにより，オーバレイでのスループットが改善され，ノードの処理が軽減される．また，アンダーレイでの通信回数が減り，ルータなどの処理が軽減される．加えて，経路が重なるか否か，つまり，次ホップが同一か否かを判断するために複数のメッセージ配送要求を一括して扱うため，総所要時間も短縮される．

経路が重複するような複数のメッセージ配送要求を扱う機会としては，例えば，DHTに対する大量データのput/getがある．DNS [13] にせよRFIDタグにせよ，格納するデータの数は数百万やそれ以上のオーダとなり，初期稼働時やバックアップ時には，その全体もしくはある部分を一度にput/getする必要が生じる．

経路の重なりは，偶然の発生を期待するのではなく，積極的に起こす．つまり，重なりそうな宛先IDの組を探して，それらをひとまとめにし，一括してオーバレイに対するメッセージ配送要求を行う．

本論文で我々は，こうして，次ホップが同一となる複数のメッセージを一括してフォワーディングする手法，一括フォワーディング *(collective forwarding)* を

†ウタゴエ（株） / 情報通信研究機構
Utagoe Inc. / NICT
‡東京大学 / 情報通信研究機構
University of Tokyo / NICT

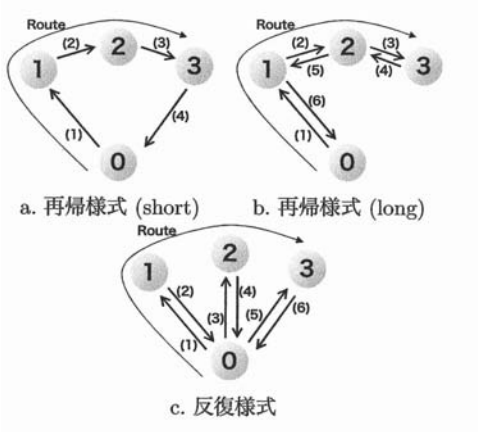


図 1: フォワーディング様式

提案する．提案手法は，オーバレイ構築ツールキット Overlay Weaver [15, 3] に実装し，その上で評価した．第 4 章で，その，通信回数と配送所要時間についての評価結果を示す．第 5 章では関連研究を示し，第 6 章で今後の展望を述べる．

2 一括フォワーディング

提案手法，一括フォワーディングを説明する．

構造化オーバレイのメッセージ配送では，宛先は ID で表現され，その宛先 ID を担当するノード (responsible node) にメッセージが配送される．経路上の各ノードは，経路表に基づいて宛先 ID に応じた次ホップを決定し，次ホップにメッセージをフォワードする．具体的には，IP 網といったアンダーレイに対して次ホップへの配送を要求する．フォワードと言っても，反復様式 (iterative forwarding/routing/lookup)（図 1）[20, 14] の場合，次ホップに対して直接通信を行うのは配送要求元ノード，つまり図 1 のノード 0 だが，本稿では，図 1c のノード 1 から 2 へ，2 から 3 へのリダイレクト，これも含めてフォワーディングと呼ぶ．

ここで，複数のメッセージ配送要求をまとめて取り扱うことを考える．それぞれの宛先 ID は異なっていてよい．あるノード上で各宛先 ID について次ホップを決定した結果，次ホップが同一ノードとなるメッセージの組があれば，それらはひとまとめにして次ホップにフォワードできる．これによって総フォワーディング回数を減らせる．メッセージがまとめられ送受信の単位が大きくなることで，スループットの向上が見込める．同時に，フォワーディングごとに必要となるメッセージ処理の回数が減るため，それに由来するノードの処理が軽減される．また，アンダーレイでの通信回

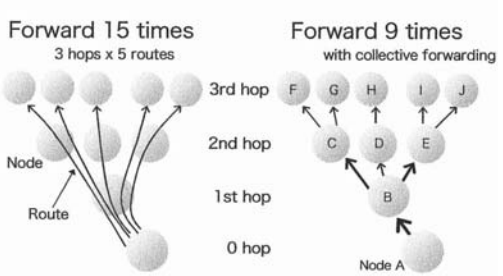


図 2: フォワーディング回数の削減

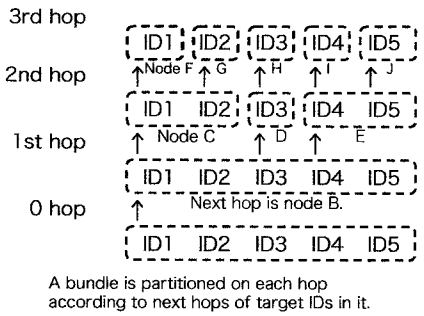


図 3: 次ホップに応じた bundle の分割

数，ひいては負担が減る．加えて，複数の配送要求をまとめて，並行して扱うため，配送の総所要時間も短縮される．こうして，複数のメッセージをまとめて一度に次ホップへフォワードする手法を，一括フォワーディングと呼ぶ．

図 2 に一括フォワーディングの具体例を示す．左は宛先 ID ごとに個別に配送を行った場合，右が一括フォワーディングを適用した場合である．ノード間の結線は 1 回のフォワーディングを表す．5 つの宛先 ID に向けて配送を行い，それぞれ経路長が 3 となっている．この場合，通常の配送（左図）では $3 \times 5 = 15$ 回のフォワーディングが必要であるところ，一括フォワーディングによって 9 回にまで削減される．

以下に一括フォワーディングの手順を示す．本稿では，複数のメッセージ，もしくは宛先 ID 群をまとめたものを bundle と呼ぶ．

1. 配送要求元ノードは，全宛先 ID を単一の bundle に含める．
2. bundle 内の全宛先 ID について次ホップを決定する．
3. 次ホップに応じて bundle を分割する．つまり，次ホップが同一ノードである宛先 ID 群を 1 つの bun-

dle にまとめる．

4. bundle ごとに，次ホップにフォワードする．
5. 2. に戻る．

図3は，図2の状況で，各ホップにおいてbundleがどう分割されるかを示している．配送要求元，つまりノードAでは，まず，全宛先IDを単一のbundleに収める．続いてノードAは，全宛先IDについて次ホップを決定し，その結果すべてノードBとなるため，分割せず，単一bundleのままとめてノードBにフォワードする．ノードBでは次ホップがノードC，D，Eと分かれるため，bundleは3つに分割され，各ノードにフォワードされる．以下同様である．

この提案手法によって，オーバレイでのスループットの改善，ノードの処理軽減が期待できる．一般に，小さな単位で通信したのでは高いスループットを達成することは難しい．例えばIP網では，ルータがTCP ACKの40バイトという小さなパケットでワイヤスピードを達成できる場合であっても，上位のTCPやUDPでは，小さな単位で通信したのでは高いスループットは達成できないことが知られている．提案手法はメッセージをまとめて大きくすることで，スループットを改善し得る．また，オーバレイでは，メッセージのバイト列へのエンコードとデコードが必要であり，この処理の回数はフォワーディングの回数に比例する．処理の負担と時間はオーバレイのプロトコル次第ではあり，定量的な議論には多くの前提が必要だが，一般に，注目に値する大きさとなる．提案手法はフォワーディング回数を減らすことで，ノードの処理を軽減する．

アンダーレイに対しては，提案手法は通信回数を減らし，ひいては負荷を軽減する．例えばIP網では，ルータがパケットごとにヘッダを解析して転送先ポートを決める．パケット転送のボトルネックはパケット1つごとに行われるこうした処理であり，このことは，ルータの処理能力がpps (packets per second) で表されることに表れている．提案手法は通信回数，つまりパケット転送の回数を減らすことで，ルータの処理を軽減する．

提案手法によって減るのは通信回数であって，トラフィックではない点に注意されたい．複数のメッセージをひとまとめにするという手法であるため，各メッセージのアンダーレイ上での経路と通信量は変わらない．メッセージヘッダ中の宛先ID以外の部分はbundle化によって総量が減るが，宛先IDとボディのサイズに比べれば影響は微々たるものだろう．

ただし，提案手法で（オーバレイ）マルチキャストを行う場合，状況は異なる．すなわち，配送要求元が同一のメッセージボディに対して複数の宛先IDを指定し，提案手法が，そのメッセージを全宛先IDに対して配送する場合である．その場合，複数の宛先IDに対してメッセージボディは1つとなるので，全宛先IDに対して個別にユニキャストを行う場合よりも ボディサイズ $\times (N-1)$ (Nは受信ノード数) だけトラフィックを削減できる．一般的なマルチキャスト手法では，受信メンバの管理，配送木の構築などによって，送信前に事前に受信者を確定しておく．それに対して提案手法には，(基盤となる構造化オーバレイさえ構築されていれば）そういった事前準備が不要であり，送信のたびに宛先ID群を指定できるという特徴がある．

3 宛先IDのbundle化

bundleは，いったん一括フォワーディングが始まれば，分割されて小さくなる一方である（図3）．一方で，配送すべきメッセージ群，つまり宛先ID群を与えられた状況で，最初にそれらをどうbundle化するか，という問題は残されている．

アンダーレイに配送を依頼するメッセージは，あまり大きくするべきではない．なぜなら，アンダーレイによっては，送受信できるパケットのサイズに制限があったり，また，そうでなくとも大きいことで信頼性の問題が生じるからである．例えばIPの場合，MTU (最大パケット長）より大きいデータグラムは分割され，分割数が多いほどデータグラムが失われる可能性が高くなる．

bundleは1つのメッセージとして次ホップにフォワードされるため，メッセージのサイズを抑えるためには，bundleに含める宛先IDの数を制限する必要がある．反復様式の場合，配送途中ではメッセージボディこそ送受信されないが，それでも，bundle内の宛先ID群はやりとりされる．構造化オーバレイではID長は128ビット以上であることが多く，例えば100のIDをbundleに含めると，それだけで1600バイト以上となってしまう．

では，bundleのサイズ制限はいつ行うべきか? アンダーレイ上での送受信サイズを抑えるためには，配送要求元による最初のフォワーディングまでには制限する必要がある．この，最も遅いタイミングで制限する場合，配送要求元が第1ホップを決定した後で制限，つまりbundle分割を行うことができ，bundle数，つまりフォワーディング回数を最小にできる．

しかし今回は，オーバレイに対するメッセージ配送

要求の手前で，bundle のサイズが上限を超えないように調整することとした．これは，DHT の場合，数千数万のキーを一度に put/get するのではなく，一定数，例えば 10 や 20 ずつ put/get する，ということにあたる．この設計は，bundle 分割の処理はオーバレイのノードに行わせるには高度過ぎる，という著者の判断に基づく．

一括フォワーディングの過程での bundle 分割は，次ホップに応じたグループ化であり，工夫の余地もないごく軽い処理である．しかし，サイズ制限を目的とした最初の bundle 構成は，そのやり方によって一括フォワーディングの効果が大きく変わってくる．経路の重複が少なく bundle サイズがすぐに 1 になってしまったのでは，通信回数削減効果は得られない．効果を得るべく，重複がなるべく起こるように bundle 群を構成する処理はすなわちクラスタリングである．ということは，処理が重いというだけでなく，さまざまな手法の間にトレードオフがあり，やり方が自明でないということである．

この種のエキセントリックな機能を基盤システムにどうサポートさせるかは魅力的な研究課題である．しかし，知られた経験則に従うなら，基盤部分に含めることは避けるべきである．一方で，オーバレイ上のノードだからこそ可能な最適化，例えば，存在するノード群をある程度把握した上での，それに応じたクラスタリングも考えられる．これは残された課題である．

4 評価

メッセージ配送に要する通信回数と所要時間について提案手法を評価した．本章でその結果を示す．

提案手法は，Overlay Weaver [15, 3] （以下，OW）に実装し，その上で評価した．OW は構造化オーバレイ構築ツールキットであり，ID に基づくメッセージ配送機構と，その上の DHT およびアプリケーション層マルチキャスト機能を提供する．ライブラリとしての利用に加えて，DHT シェル，マルチキャストシェルを通じて，開発なしに各機能を利用することもできる．

OW の場合，提案手法は routing driver 部に実装することになる．他のコンポーネント，例えばルーティングアルゴリズムに手を入れる必要はない．そのため，OW が提供するアルゴリズム，Chord，Kademlia，Koorde，Pastry，Tapestry と反復/再帰フォワーディングのすべての組み合わせで，提案手法を利用できる．

実験は，OW が提供する分散環境エミュレータの上で複数の DHT シェルを動作させて行った．このエミュレータは，計算機（Java 仮想マシン）1 台上で多数のノードを動作させ，それらノードを与えられたシナリオに従って制御できる．ここで動作するコードは，通信部分を除き，実環境で動作するものと同一のものである．

実験には，2.8 GHz Pentium D プロセッサ，x86-64 用 Linux 2.6.25，x86 用 Java 2 SE 5.0 Update 15 の HostSpot Server VM を用いた．OW の版は 0.8.7 である．すべての実験は 5 回行い，最良値と最悪値を捨て，残り 3 回分の値を平均したものを結果として採用した．

4.1 宛先 ID のクラスタリング

提案手法からより高い効果を得るためには，含む宛先 ID 群の経路がなるべく重なるように bundle を構成する必要がある（3 章）．

今回は，提案手法の可能性を調べるために，クラスタリング処理自体の効率よりも，なるべく良い結果を得ることを優先した．そのため，クラスタリングに要する時間は度外視し，次に示す単純な手法を採用した．

1. 空の bundle を用意する．また，最後に直前のクラスタに加えた ID を目標 ID とする．1 つ目の bundle に対しては，0x00...0 を目標 ID とする．
2. 未処理 ID 群から，目標 ID に対して最も近い ID を選び，bundle に移す．
3. 未処理 ID 群から，bundle 内の全 ID に対する距離の平均が最も小さい ID を選び，bundle に移す．
4. 3. を，bundle のサイズ（ID の数）が上限値に達するまで繰り返す．
5. サイズが上限値に達した bundle は，そこで確定する．
6. 1. に戻る．

ここでの距離とは，ルーティングアルゴリズムによって算出方法が異なる，ID 間の数値的な距離を指す．例えば，ID 3 から 4 への距離は，Chord であれば 1 であり，XOR 距離を使う Kademlia であれば $3 \oplus 4 = 7$ となる．つまり，提案手法およびその OW への実装はルーティングアルゴリズムを選ばないが，適切な初期 bundle 構成はアルゴリズムごとに異なる．

構造化オーバレイにおける ID 間距離には方向があり，そのため，知られた多くのクラスタリング手法は適用できない点に注意されたい．例えば Chord において，ID 3 から 4 への距離は 1 であるが，4 から 3 への距離は $2^{160} - 1$（ID が 160 ビットの場合）となる．

上述の手順は，クラスタ内の全 ID に対する距離が最短，つまりその時点では最良に見える ID をクラスタに追加していくというものであり，良い結果が期待できる．その代わり計算量のオーダは高く，実験に用いた PC で，5 万の ID を 10 ずつクラスタリングする処理に 36 時間半を要した．とはいえ，クラスタリング結果の良さをあまり損ねずに計算時間を短縮することは難しくない．

4.2 通信回数

DHT に対する put および get を行うシナリオを作成，実行し，提案手法による通信回数の削減効果を測った．1000 ノードで DHT を構成し，そこに 5 万の key-value ペアを put，続いて get した．一括フォワーディングを行う際の bundle サイズ（ID の数）は 10 とし，クラスタリングは 4.1 節で述べた方法で行った．最適な bundle サイズは，ID 長やメッセージ長に依存し，これをどう定めるかは今後の課題である．

シナリオの内容は次のとおりである．所要時間は，加入開始から 1040 秒となる．

1. 1000 ノードを起動する．
2. 1000 ノードを 20 ミリ秒ごとに DHT に加入させ，オーバレイを構築する．
3. 10 秒間，制御を休止．
4. 10 ミリ秒に 1 つのペースで，5 万の key-value ペアを put．
5. 10 秒間，制御を休止．
6. 10 ミリ秒に 1 つのペースで，5 万の key-value ペアを get．

put，get を行うノードは，シナリオ生成時に乱数で選んだ．一括フォワーディングを行う場合は，100 ミリ秒ごとに 1 つの bundle を put，get した．

図 4 に，1 秒あたりの通信回数を示す．図 4a，b，c はそれぞれ，ルーティングアルゴリズムとして Chord，Pastry，Kademlia を用いた場合の結果である．ここでのフォワーディング様式（図 1）は反復様式であるが，再帰様式も反復様式と同様の傾向を示した．また，Koorde は Chord と，Tapestry は Pastry とアルゴリズムの類似点が多く，傾向が同様であったため，ここでは代表して Chord，Pastry，Kademlia の結果を示す．

図 5 には，put と get に要した通信回数を示す．オーバレイの構築に要する通信回数は一括フォワーディングの影響を受けないので，ここでは算入しない．値は，

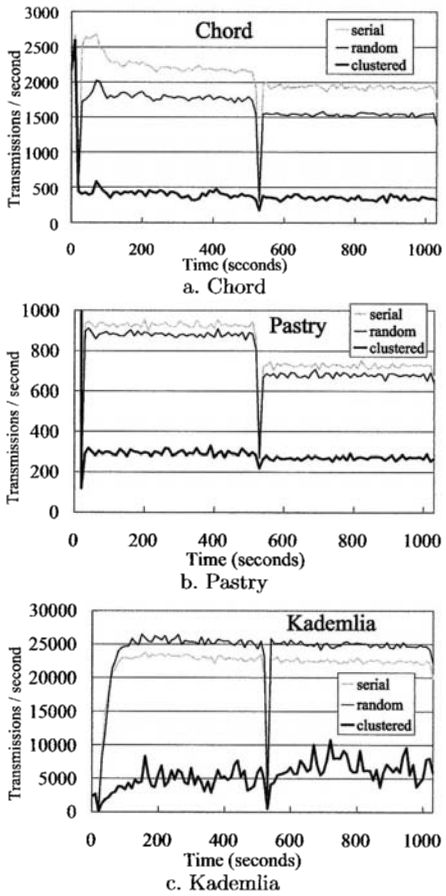


図 4: オーバレイでのメッセージ配送に要したアンダーレイでの通信回数

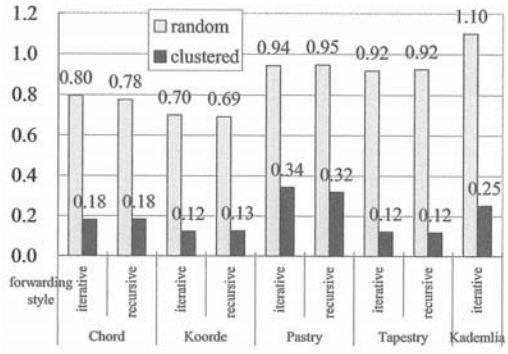
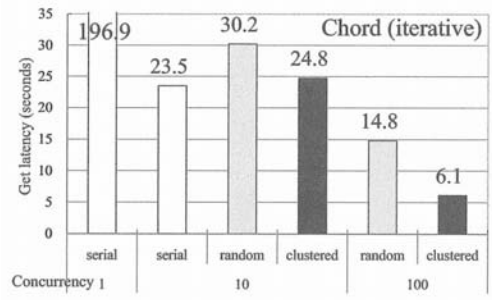
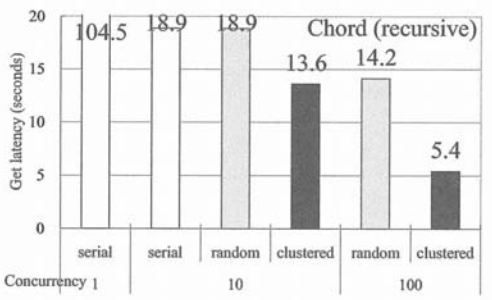


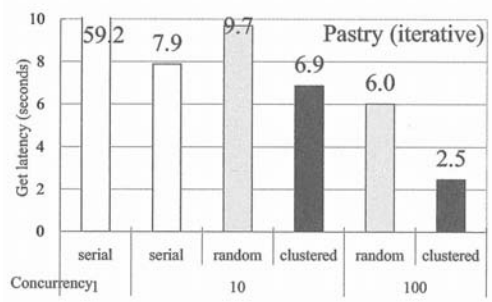
図 5: 一括フォワーディングを行わない場合に対する通信回数の比



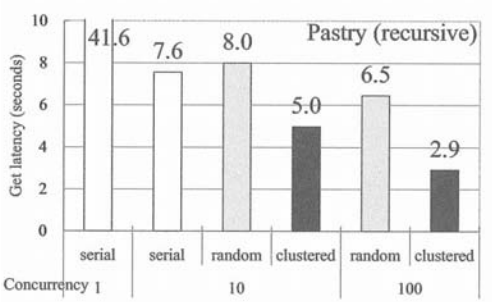
a. Chord (反復フォワーディング)



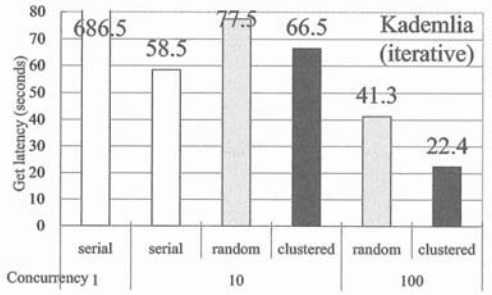
b. Chord (再帰フォワーディング)



c. Pastry (反復フォワーディング)



d. Pastry (再帰フォワーディング)



e. Kademlia (反復フォワーディング)

図 6: 10000 メッセージの配送に要した時間

一括フォワーディングを行わない場合を 1 とした比である.

図中の “serial” は一括フォワーディングを行わない場合, “random” と”clustered” は行う場合を表す. bundle サイズ制限 (3 章) のための put, get 要求のグループ化を, ”random” は無作為に行った場合, “clustered” はキーに基づいてクラスタリング (4.1 節) した場合を表す.

“clustered” では, 一括フォワーディングによって, 配送に要する通信回数が Pastry で 34%, Koorde や Tapestry で 12% まで減った (図 5).

Pastry での削減効果が他より低めなのは, Pastry, 少なくともその OW 実装では, メッセージ配信に要する通信に対して, 定常的なノード間通信の割合が他よりも高いからである. このことは, 図 4b の 500 秒過ぎ, put 完了後の通信回数落ち込み具合いが少なめであることにも表れている.

(Kademlia の “clustered” を除き) put より get のための通信回数の方が少ないことは, OW のプロトコルに起因する. get ではメッセージにキーを含めているため, 配送に対する返答をもって get が完了するのに対し, put では配送に対する返答の後, あらためて

値も含めたput要求を送る．この，putとgetの差は，経路が短いほど顕在化するので，図4ではPastryにおいて一番明確に表れている．

（Kademliaを除き）“random”であっても通信回数は若干減っている．これは，クラスタリングを行わずとも次ホップの偶然の一致はある割合で発生するから，と説明できる．

Kademliaでは，“serial”より“random”での通信回数が増えており（図5），“clustered”ではput中よりget中の通信回数が多い（図4c）．これは，通信回数の多いノードは，他ノードの生存確認のためにさらに多くの通信を行う，というKademliaの経路表管理方式に起因する．

4.3 メッセージ配送の所要時間

DHTからのgetに要する時間を測った．OWのエミュレータを用いて計算機1台上に1000ノードからなるDHTを構成し，そこから1万のkey-valueペアをgetした．分散環境を模すため，ノード間の通信に1ミリ秒の遅延を付加した．これは，LANとしては大きく，広域ネットワーク（数ミリ秒〜）としては小さい，という程度の値である．

図6に所要時間を示す．図4，図5（4.2節）と同様に，“serial”は一括フォワーディングを行わない場合，“random”はput，get要求のグループ化を無作為に行った場合，“clustered”はクラスタリング（4.1節）を行った場合を表す．

並行性(concurrency)は，オーバレイ上で同時に処理され得るメッセージ配送（つまりget）要求の数を表す．“random”，“clustered”の並行性10は，サイズ(宛先ID数）10のbundleを1つずつ配送要求することを意味する．“serial”の並行性10では，1ノードに対して10の接続を張り，それらを通じて，10までのメッセージ配送を並行して要求した．並行性100とは，10の接続を通じてサイズ10のbundleを配送要求することを指す．並行性10の“serial”は，同じだけの並行性を活かせるという観点では並行性10の一括フォワーディングとの比較が適当だと言えるが，一方で，ノードに対して10の接続を張るので，並行性100の一括フォワーディングとの比較が適当だとも考えられる．

要求メッセージ群を10ずつクラスタリングした場合（“clustered”，並行性10），並行性1の“serial”と比較して，配送の所要時間は13.0%(*Chord*, 再帰) 〜 9.7%(*Kademlia*)まで短縮された．これは，一括フォワーディングなしに，複数の接続を並行に利用した場合（“serial”，並行性10）の18.2%(*Pastry*, 再帰) 〜 8.5%(*Kademlia*)に近い短縮率である．さらに，複数のbundleを並行して配送要求することで（並行性100），所要時間を7.03%(*Pastry*, 再帰) 〜 3.12%(*Chord*, 反復)まで短縮できた．

5 関連研究

提案手法と同様の効果を得るために，宛先IDに対する担当ノードを算出して，フォワーディングを行わずに，直接，担当ノードに対してput，get要求を行うという手法が考えられる．そのためには，まず，オーバレイ上のノード群についてIDを把握する必要があり，そのためのコストがフォワーディングより低くなるか否かは明らかではない．特に，ノードの離脱，churnを前提とすると，コスト高となることが予想される．そもそも，各ノードが持つ情報をO(log N)以下に抑えられることが構造化オーバレイの特徴であり，この強みを捨てることとなる．提案手法は，この強みを損ねずに，多数のメッセージ配送要求を効率化する．

提案手法のように配送要求の時点でbundle化しておくのではなく，ノード上で動的にbundle化を行うことも考えられる．しかしそのためには，ノード上でのメッセージの待ち合わせが必要となり，その待ち時間の分，配送が遅くなってしまう．

Xcast [6]対応ルータも，一括フォワーディングと同様に，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する．Xcastは，メンバ数の少ないグループ向けのマルチキャストプロトコルである．Xcastパケットはヘッダに複数の宛先IPアドレスを含み，それを，Xcast非対応ルータは宛先のうちの1つに転送，対応ルータは複数の宛先について次ホップを決定して，各次ホップに対して1回ずつフォワーディングを行う．Xcast対応ルータが提案手法と異なるのは，XcastはIP層のプロトコルである点，マルチキャストが目的であり，ユニキャストメッセージの一括配送は対象としてない点である．また，宛先がIPアドレスであり，よほど共通プリフィクス長が長い場合を除いて，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できず，よって，クラスタリング（4.1節）によるフォワーディング回数の削減といった最適化の余地はない．

IPマルチキャストも，多数の宛先に向けたメッセージ配送を効率化する技法である．どのプロトコルも，事前に配送木を構築する．それに対して提案手法は，事前に受信者の確定は行わず，送信のたびに宛先ID

群を指定できるという特徴がある（2章）．

提案手法のような複数配送要求の効率化ではなく，単一の配送を効率化するには，1ホップあたりの通信遅延を小さくするか，経路長を短くすればよい．前者の試みとしてproximity routing [10]が，後者の試みとして例えば1-hop DHTがある．必ず1ホップで済ませるkey-valueストアDynamo [8]や，なるべく1ホップで済ませようとするJXTAのloosely-consistent DHT [18]も同様の試みである．

6 まとめ

本論文では，オーバレイでのメッセージ配送を効率化し，IP網などアンダーレイの負担を軽減する手法，一括フォワーディングを提案した．次ホップが同一となるメッセージをまとめてフォワードすることで，総フォワーディング回数，ひいてはアンダーレイでの通信回数を減らす．

提案手法をOverlay Weaverに実装し，いくつかのルーティングアルゴリズムと組み合わせて動作させた結果，通信回数は34% ～ 12%まで，所要時間は13.0% ～ 9.7%まで減った．さらに，複数のbundleを並行して配送要求することで，所要時間は7.03% ～ 3.12%まで短縮できた．

今後は，DNSやセンサデータといった実データを記憶，処理するオーバレイを構築し，提案手法の適用可能性や効果を調べていく．また，構造化オーバレイに限らず，非構造化を含めたオーバレイ一般への提案手法の適用も興味深い．

一方で，将来のインターネットやその先のネットワークに必須の要件と見なされているID/Locator分離 [12, 9, 4, 17, 11]に，DHTや，さらにはオーバレイのルーティング・フォワーディング技法を適用する試みが始まっている [16, 5, 19]．そこで提案手法が活きるとすれば，ネットワークの側で，IDに基づく集約サービス(ID-based aggregation service)として提供することも考えられる．

謝辞

提案手法の有効性について議論して頂いた阿多信吾様，DHTでの大量データの取り扱いを考えるきっかけを下さった藤田昭人様，また，本研究の基盤であるOverlay Weaverを開発する機会を下さった関口智嗣様，田中良夫様に感謝致します．

参考文献

[1] Azureus: Java BitTorrent client. http://azureus.sourceforge.net/.

[2] BitTorrent. http://www.bittorrent.com/.

[3] Overlay Weaver: An overlay construction toolkit. http://overlayweaver.sf.net/.

[4] Site multihoming by IPv6 intermediation (shim6). http://www.ietf.org/html.charters/shim6-charter.html.

[5] Jeff Ahrenholz. HIP DHT interface. draft-ahrenholz-hiprg-dht-02, Internet-Draft, IETF, January 2008.

[6] Rick Boivie, Nancy Feldman, Yuji Imai, Wim Livens, and Dirk Ooms. Explicit multicast (Xcast) concepts and options. RFC 5058, IETF, November 2007.

[7] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a common API for structured peer-to-peer overlays. In *Proc. IPTPS'03*, February 2003.

[8] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. In *Proc. SOSP 2007*, October 2007.

[9] Dino Farinacci, Vince Fuller, Dave Oran, and Dave Meyer. Locator/ID separation protocol (LISP). draft-farinacci-lisp-07, Internet-Draft, IETF, April 2008.

[10] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proc. SIGCOMM 2003*, August 2003.

[11] 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*, Vol. E84-B, No. 8, pp. 2076–2086, August 2001.

[12] Robert Moskowitz and Pekka Nikander. Host identity protocol (HIP) architecture. RFC 4423, IETF, May 2006.

[13] Vasileios Pappas, Dan Massey, Andreas Terzis, and Lixia Zhang. A comparative study of the DNS design with DHT-based alternatives. In *Proc. INFOCOM 2006*, April 2006.

[14] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *Proc. USENIX '04*, June 2004.

[15] Kazuyuki Shudo, Yoshio Tanaka, and Satoshi Sekiguchi. Overlay Weaver: An overlay construction toolkit. *Computer Communicatios (Special Issue on Foundations of Peer-to-Peer Computing)*, Vol. 31, No. 2, pp. 402–412, February 2008.

[16] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *Proc. ACM SIGCOMM 2002*, pp. 73–86, August 2002.

[17] Fumio Teraoka, Masahiro Ishiyama, and Mitsunobu Kunishi. LIN6: a solution to multihoming and mobility in IPv6. Internet-Draft, IETF, December 2003.

[18] Bernard Traversat, Mohamed Abdelaziz, and Eric Pouyoul. Projext JXTA: A loosely-consistent DHT rendezvous walker, 2003.

[19] 吉田幹, 寺西裕一, 下條真司. オーバレイネットワークにおけるID/Locator分離機構. 情報処理学会研究報告, 2008-DPS-134, March 2008.

[20] 首藤一幸, 加藤大志, 門林雄基, 土井裕介. 構造化オーバレイにおける反復探索と再帰探索の比較. 情報処理学会研究報告, 2006-OS-103-2, July 2006.