情報処理学会論文誌　コンピューティングシステム　Vol. 2　No. 3　47–57 (Sep. 2009)

トポロジ情報を用いた効率的かつ漸近安定な大容量ブロードキャスト

柴田 剛志[†1]　田浦 健次朗[†1]

多拠点クラスタやグリッド環境における大規模データのブロードキャストに対して，トポロジ情報を利用し転送経路の重複を極力避け効率的に転送するさまざまな方法が研究されているが，あるノードセットで効率的に転送できても，それに少数のリンクやノードを付け加えると全体のパフォーマンスを落としてしまうことがある．そのような問題を解決するため，高橋らはブロードキャストの安定性という概念を考え，いくつかの条件の下で安定なブロードキャストアルゴリズムを提案している．その方法ではバンド幅が付与されたトポロジを用いるが，マルチクラスタなどの広域ネットワークを使う場合，実際に使うことができるバンド幅をトポロジに付与することは難しい．そこで，バンド幅が付与されていないトポロジを用い，徐々に接続数を増やしながら漸近的に安定となるブロードキャストアルゴリズムを新たに提案する．

Topology-aware and Bandwidth-oblivious Broadcast for Large Data

Takeshi Shibata[†1] and Kenjiro Taura[†1]

In multi-cluster or grid computing, as commonly happens, few nodes or few links make a bottleneck of whole system. when one broadcasts large data to every node in the cluster and make some calcution, it is important to circumvent the effect of bottleneck nodes or links. Takahashi et. al. has been introduced the notion of stability of blodcasting and the stable broadcast algorithm. Althogh their algorithm uses bandwidth-aware topology, the assumption that the bandwidth of each link is known is a little strong. The available bandwidth are varied at every time due to the effection of cross traffic in multi-cluster or grid computing. In this paper, we introduce a new aproach to broadcasting under only topology aware network.

1. はじめに

多拠点に配置されているクラスタを協調させて計算を行う場合，広域ネットワークを経由しての通信が必要となるため，シングルクラスタとは異なる集合通信の最適化が必要となる．P2P ファイル共有システムなどの場合のように，広域ネットワーク上の任意の位置にばらばらに参加ノードが配置されているわけではないため，ある程度の対称性を持った塊に分かれて分散されているネットワークを想定しなければならない．そこで，本論文では，そのようなネットワークのモデルとして，k-heterogenous という性質を定義し，ネットワークの不均一性を表す．ここで k はネットワークが均一なクラスタの数である．

近年，大容量ブロードキャストとして，Takahashi ら[6] は，ブロードキャストの安定性という概念を提案しており，バンド幅つきのネットワーク情報を用いて，特に参加するノードのルーティングが木と見なせるときに，その性質をみたすアルゴリズムを提案している．マルチクラスタなどの広域ネットワークを使う場合，実際に使うことができるバンド幅をデータに付与することは難しい．そこで本論文では，さらなる改良として，バンド幅の付与されていないトポロジ情報からでも，漸近的に安定なオーバレイネットワークが構築されるブロードキャストアルゴリズムを提案する．

2. 用語・記号に関する約束

以降で使用する用語と記号をまとめて書いておく．本論文で用いるネットワークトポロジは，次を満たすグラフのこととする．(1) ノードはエンドノードとスイッチの 2 種類からなる．(2) 各ノードを結ぶエッジは双方向であり，それぞれの方向にバンド幅を表す値がラベル付けされている．

特に断りがないかぎり，N，D，E はそれぞれトポロジ，エンドノードの集合，エッジの集合を表し，$B(e)$ はネットワークトポロジにおけるエッジ e のバンド幅を表すものとする．ルーティングは，2 つのエンドノードからパスへの写像で表されるとする．エンドノード $a \in D$ からエンドノード $b \in D$ への，N 上のルーティングから決まるパスを $a \to b$ で表すこととする．また，$B(a \to b)$ は $a \to b$ 上のエッジのバンド幅の最小値であるとする．

以降，アルゴリズムに関する記述において，単に接続と書いたときは，グラフのエッジの

†1 東京大学
The University of Tokyo

ことではなく，ルーティングから決まるエンドノード間のパスのこととする．また，N には，データの転送に参加しないエンドノードおよび関係しないスイッチは含めないものとする．また，s はブロードキャストのソースノードを表すものとする．

2.1 k-heterogenous

グリッド環境は，定義上，すべての計算機が広域上に個別に分散配備され，不均一なネットワークで結ばれていてもよいが，多くの場合，多拠点にまたがるクラスタの集合体であることが多い．そのような場合，クラスタ内ではネットワークのバンド幅はある程度均一になっている．そこで，本論文では，転送に関係するノードからなる部分グラフ G に対し，その不均一性を表す量として，k-heterogenous（**図 1**）を次のように定義する．G の中に以下を満たすような交わりを持たない連結な m 個の部分グラフ $A_1, \cdots, A_m$ をとることを考える．

- $A_1 \cup \cdots \cup A_m$ は転送参加のノードをすべて含む．
- 各 A_i の任意のエッジ e に対するバンド幅 $B(e)$ は，ある実数 f_i があって，$(1+\varepsilon)^{-1} f_i \leq B(e) \leq (1+\varepsilon) f_i$ を満たす．

ここで，ε はバンド幅の違いの許容範囲を表すための値で，$0 \leq \varepsilon \leq 1$ とする．G が最小で k 個に分けることができるとき，k-heterogenous であると呼ぶこととする．また，$A_1, \cdots, A_k$ が ε-separatable とは，A_i のエッジのバンド幅の最小値を minBW_i として，任意の $a \in A_i$ に対し，$b \notin A_i$ ならば $B(a \to b) < (1+\varepsilon)^{-1}\mathrm{minBW}_i$ をみたすこととする．直感的には，各 A_i をその内部のネットワークが homogenous なクラスタを表すと考えたとき，クラスタ内の利用可能なバンド幅のほうがクラスタ間の利用可能なバンド幅よりも小さい場合，ε-separatable である．

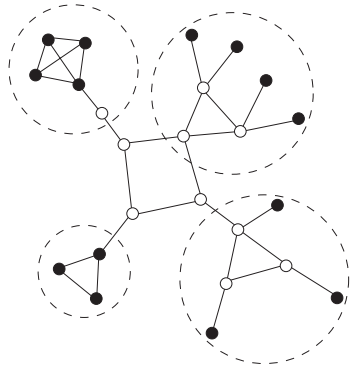

図 1　k-heterogenous
Fig. 1　k-heterogenous.

3. 関 連 研 究

本論文の対象である大規模データのブロードキャストは，主に次の 3 つの観点から研究されている．

(1)　並列分散システムの集合通信アルゴリズム．
(2)　P2P でのファイル共有システム．
(3)　CDN を用いたファイル配布やストリーミング．

それぞれ，大規模データのブロードキャストという点では同質のものであるものの，留意すべき点が異なる．たとえば，集合通信のアルゴリズムとしては，参加・脱退よりも，いかに高速にブロードキャストが終了するかという点に主眼が置かれ，アルゴリズムが考えられる．最適化のためには，ネットワークのトポロジ情報がしばしば用いられる[3),15)]．その一方で，P2P によるファイル共有システムでは，ロバスト性が必須条件であり，特定のサーバやトポロジの情報を持たず，P2P ネットワークが自律的に構築されることが必要とされる[10),11)]．また，ファイル転送は基本的にさまざまなピアを持つため，断片に分割され，それぞれが転送される形をとる．CDN では，データをデュプリケートするための中間的なノード層をいかに分散させて配置するかや，それらのノードの信頼性の確保に主眼がおかれる．加えて，ストリーミング技術では，断片化されたデータがあまりばらばらに到着してはならず，ある程度の入れ違いはあっても先頭から順に，なるべく一定の速度で到着する，という時間的な制約がある[5)]．

(2) や (3) では広域のネットワーク上の任意の位置でノードが参加，脱退することを前提としているため，陽に与えられるネットワークトポロジは基本的に仮定されない．一方で場所の情報や，RTT やバンド幅を距離として接続を切り替えることで，暗にトポロジ情報を用いることはよくある[1),13)]．本研究では，2.1 節のような k-heterogenous な環境（ネットワーク上で完全に分散されておらず，均一性を持った k 個のかたまりからなるネットワーク）を対象に考えるため，システムのロバスト性よりも終了時間の最適化を主眼においている，ネットワークトポロジを利用しているという点で，(1) の視点にたっているといってよい．しかし，後述するように，(1) では完全に終了するまでの時間が主眼におかれるが，本研究ではほかの参加ノードに依存せず，安定的にバンド幅を確保できることや，各ノードでの終了時間の平均を最小化する方向で考えているという点で，(2)，(3) の視点も取り入

れられているといってよい．

トポロジを仮定することに関しては，SNMP などによって，スイッチにおける基本情報を知るすべが整備されていることや，ネットワークトポロジの推定の手法も確立されてきており[7),8)]，広域ネットワークのさまざまな場所から参加脱退が行われない限り，十分現実的であるといえる．FPFR[4)] では，トポロジ情報を用い，転送に参加しているノードを深さ優先でたどり，パイプラインを複数構築することによって性能の向上を図っている．FPFR では，すべての参加ノードを網羅し，かつ 0 でないバンド幅を割り当てられるパイプラインを追加することができなくなると，それ以上パイプラインを構築しない．そして，事前にバンド幅に基づきデータを分割して各パイプラインに割り当てる．

Stable Broadcast[6)] は，FPFR を変形した方法であり，すべての参加ノードを網羅しなくても，0 でないバンド幅が割り当てられるようなパイプラインを作ることができるかぎり，それを追加していく．そして，転送に際しては，あらかじめデータを比較的小さいサイズのチャンクに分けておき，各ノードは持っていないチャンクを 1 つ前のノードにリクエストする．Stable Broadcast の狙いは，次章で述べるように，各計算ノードの受け取るバンド幅を最大化することである．ブロードキャスト全体の終了時間は変わらなくても，各計算ノードにおいて転送が早く終了すれば，そのノードの上では早く計算を開始することができる．このことは，たとえばパラメータスィープをする場合など，タスクのスケジューリングの効率化の観点から重要となる．

4. ブロードキャストの安定性と効率性

ブロードキャストのコンプリーションタイムの最小化は，一般的には，あて先ノードの数に対して NP 困難になる[14)]．しかし，RTT が無視できるほどデータが大きく，転送ノードの間でルーティングの経路が木になる場合に限ると，その木を深さ優先でたどって到達できる順にノードを並べてパイプライン転送を行うことによって達成することができる．ここでいうパイプライン転送とは，すべてのデータを受信し終わるのを待たずに，各ノードが受け取った分だけ次のノードに送信する方法である．この場合，ブロードキャスト自体のコンプリーションタイムは最小化される．しかし，パイプライン転送では，パイプラインの途中にボトルネックとなるようなノードを経由している場合，そのノードがない場合と比べ，そのノード以降の転送のパフォーマンスが落ちる．

一方，BitTorrent[9)] などの方法では，転送の参加脱退やロバスト性が重要視されており，経験上ノードをいくつか追加しても全体のパフォーマンスが著しく落ちることはない．しかし，実際のトポロジとは直接的には関係なく，ランダムに近い形でオーバレイネットワークが構築されるため，通信路の干渉がおこったり，同一のデータが何度も同じリンクを使って転送されることがあり，コンプリーションタイムの最小化という観点からは非効率である．

ソースノード s からエンドノード d への転送において利用可能な，未割当てのバンド幅を $\mathrm{AvailBandwidth}(s \to d)$ とする．バンド幅つきのトポロジデータからは，以下のように計算することができる．

$$\mathrm{AvailBandwidth}(s \to d) = \min_{e \in s \to d} B(e)$$

高橋らは，ブロードキャストの安定性を次のように定義している．

定義 1　宛先ノードの集合 D，ソースノード s に対し，ブロードキャストアルゴリズム A によって $d \in D$ に割り当てられるバンド幅を $\mathrm{Bandwidth}(A, s, D)(d)$ とする．任意の d について，

$$\mathrm{Bandwidth}(A, s, D)(d) = \mathrm{AvailBandwidth}(s \to d)$$

が成り立つとき，A は**安定**であるという．また，転送に使われるオーバレイが安定であるとは，その上で行うブロードキャストが安定になるような各接続に対するバンド幅の割当てが存在することとする．

すなわち，ブロードキャストが安定であるということは，ブロードキャストの宛先ノードが 1 つしかないときのバンド幅と同じバンド幅を，複数ノード宛になっても割り当てることができるということである．前述のパイプライン転送はコンプリーションタイムは最小化するものの，この意味で安定ではないといえる．

一般に，ワークフローなどにおいて，データが到着した順に各ノードでタスクを実行するような状況では，効率的にスケジューリングを行えば，タスクの粒度が細かければ，タスク全体が終了する時間に影響を与える主なファクターはコンプリーションタイム[⋆1]ではなく各ノードの転送終了時間の総和である．

$$\sum_{d \in D} \mathrm{Datasize/Bandwidth}(A, s, D)(d)$$

したがって，効率性を図る指標としては，ブロードキャストのコンプリーションタイムだけでは不十分であり各ノードの転送終了時間の総和も必要となる．

⋆1 $\max_{d \in D} \mathrm{Datasize/Bandwidth}(A, s, D)(d)$.

トポロジが木である場合，ソースから参加ノードへのパスは 1 通りしかないため，どのようなアルゴリズムを用いても，宛先ノードが 1 つしかないときのバンド幅を上回ることはできない．すなわち，任意の A と d について，

$$\mathrm{Bandwidth}(A, s, D)(d) \leq \mathrm{AvailBandwidth}(s \to d)$$

が成り立つ．したがって，トポロジが木である場合は，定義 1 の意味で安定ならば，コンプリーションタイムを最小にし，かつ各ノードの転送終了時間の総和も最小にすることが分かる．逆に，各ノードの転送終了時間の総和を最小にすれば，トポロジが木の場合，安定であることも分かるから，本論文では各ノードの転送終了時間の総和を最小にすることを最適化と呼ぶことにする．

4.1 Stable Broadcast

Takahashi ら[6] は，FPFR[4] の改良として，Stable Broadcast というブロードキャストのオーバレイネットワーク構築手法を提案している．これは，特に次の条件をみたすとき，前節の意味で安定であることが保証されるアルゴリズムである．

- 転送にかかわるエンドノードのルーティングから作られるグラフが木である．
- トポロジにおいて，各エッジの双方向のバンド幅が等しい．

なお，トポロジが木構造であるかにかかわらず，FPFR の改良であるので，前節の意味での安定性は保証されなくても，より効率的な方法となっている．

```
1: T = ∅
2: B_o = B
3: while TRUE do
4:     トポロジに基づいて，s から B_o(e) = 0 でないエッジ e
       を深さ優先順にたどったエンドノードを順に d_1, ··· , d_k
       とする．
5:     t = {s → d_1, ··· , d_{k−1} → d_k}
6:     if t が空 then
7:         break
8:     end if
9:     u = min{B_o(e)|e は t の要素に含まれるエッジ }
10:    T = T ∪ {⟨t, u⟩}
11:    for t の各要素に含まれる各エッジ e do
12:        B_o(e) = B_o(e) − u
13:    end for
14: end while
15: return T
```

図 2　Stable algorithm
Fig. 2　Stable algorithm.

バンド幅つきのトポロジ情報をもとにして，あらかじめ転送のセットと各転送に割り振るバンド幅が計算される（**図 2**）．はじめに，バンド幅に余裕があるリンクのみをトポロジを使って深さ優先順にたどることでパイプラインを作り，そのパイプラインに使われるエッジのうちバンド幅の最小のものを，そのパイプラインのバンド幅として割り当てる．パイプライン に割り当てたバンド幅を，トポロジのエッジのバンド幅から引く．残りのバンド幅が 0 となってしまったエッジを除いて，残ったエッジとノードのみでパイプラインを構築する．これを繰り返し，複数のパイプラインを構築する．最終的に，ソースノードからどのエンドノードへもパイプラインを作ることができなくなったときに終了する．

図 3 はアルゴリズムの挙動の例である．はじめのパイプラインが構築され，バンド幅 10

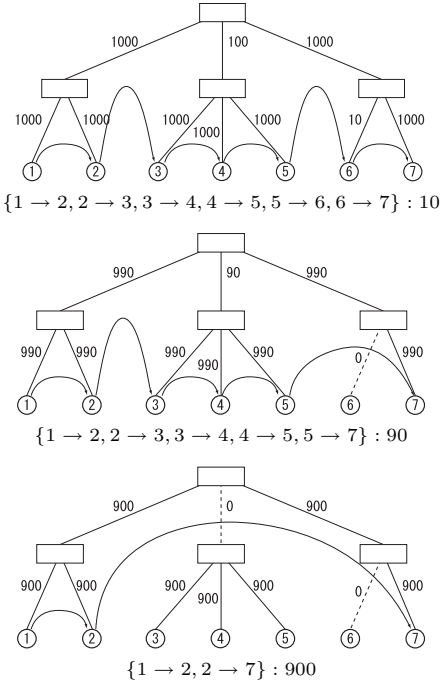

図 3　Stable Broadcast：転送路の構築の例
Fig. 3　An example for the overlay network constructed by Stable Broadcast.

が割り当てられる（図 3 上）．パイプラインに使われたエッジ上のバンド幅から 10 を引き，残ったバンド幅が 0 でないエッジをたどって 2 本目のパイプラインが作られる（図 3 中）．同様にして 3 本目のパイプライン（図 3 下）が作られた後，それ以上はどのエンドノードへの転送も 0 でないバンド幅を割り当てることができないため終了する．

5. 提 案 手 法

本章では，バンド幅の情報が未知で，ルーティングのトポロジ情報のみ与えられている場合でも，前章と同じオーバレイを構築するための方法を述べる．そのうえで，前節と同じ条件下で，安定なオーバレイが構築されることを証明する．また，断片化されたデータ（チャンク）をより効率良く送る方法も提案する．

5.1 オーバレイの構築

使用可能なバンド幅が未知な状況で，正しくボトルネックとなるリンクを知るためには，なんらかの形でそれを測定する必要がある．事前にバンド幅を測定する方法ではネットワークリソース，転送時間ともに無駄が生じるため，転送を行いながらそのバンド幅を測定し，徐々に接続を追加しながら転送を行ったほうがよい．

接続の集合 C とそれに対するバンド幅の割当て $h : C \to \mathbb{R}$ がすでにあったときに，新たな接続 $d \to d'$ に対して，使用可能なバンド幅を問うクエリを $\mathrm{avail}(C, h, d \to d')$ で表すこととする．これは実際には接続に対してそのバンド幅の測定値を返す関数である．この関数の値は，バンド幅つきのトポロジデータからは，現在使用しているバンド幅を引いた後に残った，使用可能なバンド幅として計算することができる．

$$\mathrm{avail}(C, h, d \to d') = \min\{\mathrm{in}(C, h, d) - \mathrm{out}(C, h, d), \min_{e \in d \to d'} \{B(e) - \sum_{e \in c \in C} h(c)\}\}$$

となる．ここで，in，out は，各エンドノードの転送の流出，流入を表し，

$$\mathrm{in}(C, h, d) = \sum_{d_p \to d \in C} h(d_p \to d)$$

$$\mathrm{out}(C, h, d) = \sum_{d \to d_n \in C} h(d \to d_n)$$

である．ただし $\mathrm{in}(C, h, s) = \infty$ とする．

提案手法である Bottleneck Skipping Algorithm では，シンプルなパイプライン転送から，バンド幅の測定結果とトポロジ情報を用い，効率的に接続を追加することによって，前節の Stable Broadcast で得られるものと同じオーバレイを構築して出力する．ここでいうオーバレイとは，接続の集合（C）と各接続へのバンド幅の割当て（h）のことである．

大筋としては，次のようなアルゴリズムである．まず，流入バンド幅に対して流出バンド幅が小さいエンドノードがあると，そのエンドノードから次のエンドノードを飛ばすように新たに接続を作成する．その結果，流出バンド幅が増えたらその接続を採用する．増えなければ，その接続は採用せず，そのかわり，飛ばしたエンドノードとその先のエンドノードを含む部分木に属するエンドノードはまとめて飛ばしてよいと判断する．

以下，**図 4** にそってその挙動を説明する．まず，すべてのエンドノードをソースノードから順に深さ優先順にリスト L に登録しておき，L で隣り合っているノードに対し接続を作成する．そして，**図 5** の MeasureForChain で，実際に転送を行い，利用可能なバンド幅を測定する．なお，実際の転送の方法については次節で述べる．

次に，図 5 中の TryAndSkip を繰り返す．TryAndSkip の中では，まず，L に含まれるエンドノードのうち，流出のバンド幅が流入のバンド幅よりも ε（$0 \le \varepsilon < 1$）以上の割合で小さくなるものを列挙する．そして，L において一番後ろのエンドノードを d_i とする．なお，$\mathrm{in}(C, h, s) = \infty$ としているため，必ずそのようなエンドノードが存在する．

その後，L および新たに測定した各ノードでの流入出のバンド幅に応じて，次の 3 つに場合分けされる．

(case 1) d_i が L の最後尾または最後尾の 1 つ前であった場合，d_{i+2} が存在しないので，L から最後尾のエンドノードを除いて終了する．

(case 2) d_i から d_{i+2} へ接続を追加し転送を開始する．d_{i+1} から d_{i+2} への転送を一時的に中断したうえで，d_i からの流出バンド幅を測定し，今までと比べ ε 以上の割合で増やすことができているかチェックする．できている場合，$d_i \to d_{i+2}$ を C に追加し，L から d_{i+1} を除く．また，L 中の d_i 以降のエンドノードについて，利用可能なバンド幅を再度測定して，割り当てる．

(case 3) case 2 の条件を満たさない場合，$d_i \to d_{i+1}$ を切断し，さらに L から除くエンドノードを次のようにして決める．まず，トポロジ情報を用いて，パス $d_i \to d_{i+1}$ とパス $d_i \to d_{i+2}$ の分岐ノードが切断点となっているかどうかチェック[⋆1]する．切断点となっている場合は d_i から分離される d_{i+2} 以降のエンドノードをすべて L から除き，なっていない場合は d_{i+2} のみを除く．

⋆1 一般に，ノードが連結グラフの切断点であるとは，そのノードを除くとグラフが連結でなくなるような点のことである．また $a \to b$ と $a \to c$ の分岐ノードとは，a からの共通のパスの終点のこととする．一般に，トポロジが木の場合は任意のノードが切断点となっているので，この条件はつねに満たされる．

Procdure　MakeOverlay()
1: $L = \langle d_0, d_1, \ldots, d_n \rangle$ を，s（$= d_0$）からトポロジに基づいて深さ優先順にたどったエンドノードのリストとする．
2: $C = \{d_0 \to d_1, \cdots, d_{n-1} \to d_n\}$
3: $h = \text{MeasureForChain}(\varnothing, [\,], C)$
4: **while** L が空でない **do**
5:　　$C, h, L = \text{TryAndSkip}(C, h, L)$
6: **end while**
7: **return**　C, h

Procdure　TryAndSkip(C, h, L)
1: L の要素を順に $d_0, \ldots, d_m$ とする
2: $i = \max\{j \in [1, m) \mid \frac{\text{in}(C,h,d_j)}{1+\varepsilon} > \text{out}(C, h, d_j)\}$
3: /* case 1 */
4: **if** $i + 2 > m$ **then**
5:　　L から最後尾の要素を取り除く
6:　　**return**　C, h, L
7: **end if**
8: $x = \text{out}(C, h, d_j)$
9: $y = \text{avail}(C - \{d_{i+1} \to d_{i+2}\}, h, d_i \to d_{i+2})$
10: /* case 2 */
11: **if**　$y > \varepsilon x$　**then**
12:　　$C_n = \{d_{i+2} \to d_{i+3}, \cdots, d_{m-1} \to d_m\}$
13:　　$C_o = C - C_n$
14:　　$h =$MeasureForChain$(C_o, h, \{d_i \to d_{i+2}\} \cup C_n)$
15:　　$C = C \cup \{d_i \to d_{i+2}\}$
16:　　**return**　$C, h, L - \{d_{i+1}\}$
17: **end if**
18: /* case 3 */
19: **if** $d_i \to d_{i+1}$ と $d_i \to d_{i+2}$ の分岐ノード v が切断点となっている **then**
20:　　$L = L - \{v$ を除くと d_i へのパスがなくなるノードのうち d_{i+2} 以降のノードすべて $\}$
21:　　**return**　C, h, L
22: **else**
23:　　**return**　$C, h, L - \{d_{i+2}\}$
24: **end if**

Procdure　MeasureForChain(C_o, h, C_n)
Require: C_n はチェーンとなっている．
1: C_n を $\{d_1 \to d_2, \cdots, d_{N-1} \to d_N\}$ とする．
2: **for** $i = 1 \cdots N - 1$ **do**
3:　　$C_o = C_o \cup \{d_i \to d_{i+1}\}$
4:　　$h[d_i \to d_{i+1}] = \text{avail}(C, h, d_i \to d_{i+1})$
5: **end for**
6: **return**　h

図 4　Bottleneck Skipping Algorithm
Fig. 4　Bottleneck Skipping Algorithm.

L の中から除かれたノードを飛ばされたノードと呼ぶことにする．トポロジ情報を使うことによって効率的になっている箇所は case 3 であり，ここで飛ばしてもよいと考えられる複数のノードを 1 度に飛ばしている．1 度に飛ばしてもよいとする直感的な理由は，トポロジが木の場合，切断点によって d_i から分離されるエンドノードにはこれ以上接続を追加してもそのエンドノードの流入バンド幅を増やすことができないためである．

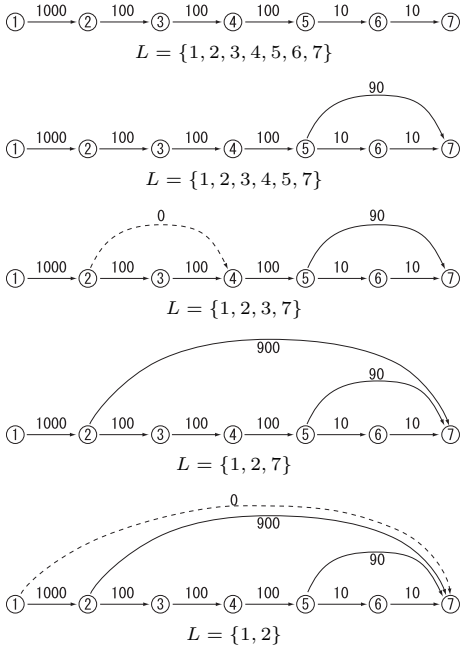

図 5　Bottleneck Skipping Algorithm：接続の追加の例
Fig. 5　Bottleneck Skipping Algorithm: An example for the overlay network construction.

図 5 は図 3 のトポロジ（バンド幅は除く）が与えられたときの，構築の様子を表している．まず，Stable Broadcast の場合と同様に，パイプラインが構成される（図 5-1）．また，パイプラインにそって，利用可能なバンド幅が転送と同時に計測され，割り当てられる．次に，TryAndSkip が実行され，流入が 100，流出が 10 であることから $d_i = 5$ となり，5 から 7 へ接続が張られ，利用可能なバンド幅が計測される（図 5-2）．その結果，$5 \to 7$ に 90 のバンド幅が割り当てられ，case 2 に従って，L から 6 が消去される．その次の TryAndSkip の呼び出しでは，$d_i = 2$ となり，2 から 4 へ接続が張られるが，計測の結果利用可能なバンド幅が 0 となるため，その接続は採用されない（図 5-3）．このとき，case 3 に従って，L から 4，5 が消去される．このような繰返しにより，接続が追加され，L からエンドノードが単調に減っていき，図 5-5 のように L の要素がソースノードを含め残り 2 つになると，以

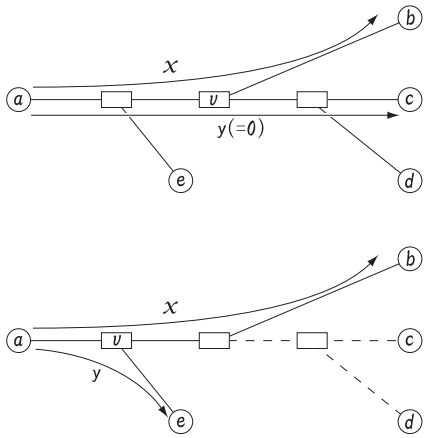

図 6　ボトルネックリンクの位置の推定
Fig. 6　Identifying bottleneck links from measurement of bandwidth.

降は case 1 のみが該当するようになり，最後に $L = \varnothing$ となり MakeOverlay が終了する．

4.1 節の条件をみたすとき，$\varepsilon = 0$ ならば MakeOverlay で構築されるオーバレイが安定となる．Stable Algorithm では，バンド幅の最も小さいエッジから順に取り除かれ，それによってパイプラインから除かれるエンドノードが決まり，それらを飛び越えるように新たに接続が追加される．以下では提案手法でも同じことが起こることを示す．

TryAndSkip 内において，$(a, b, c) = (d_i, d_{i+1}, d_{i+2})$ とおく．また，$a \to b$ と $a \to c$ の分岐ノードを v とする（**図 6**）．パス $a \to b$ 上のエッジのうちで，バンド幅が最小であるエッジを e とする．このとき，d_i の決め方から，s から L の各要素へのパス上のエッジのバンド幅のうちで，e のバンド幅が最小であることが分かる．

ここで，case 2 の条件を満たす場合，$v \to b$ 上に e があることが分かるから，$a \to c$ は Stable Broadcast で e を削除したときに新たに付け加えられる接続に等しい．

一方，case 2 の条件を満たさなかった場合，トポロジが木であることから case 3 が必ず成り立つ．$b \to c$ へのバンド幅の割当てを 0 としたうえで $a \to c$ の利用可能バンド幅を測定しているので，$a \to v$ 上に e があることが分かる．e が $a \to v$ のどこにあるかは分からないが，少なくとも v を除くと a からのパスがなくなるようなエンドノードに関しては，Stable Broadcast で e を除くことにより新たに a から追加される接続でも飛ばされることになる．したがって，それらのエンドノードは L から削除してよい．この場合は新たな接続は追加されず，TryAndSkip が繰り返される．

また，最終的に L は空となるが，これは Stable Broadcast の終了条件，つまり，エッジを取り除いていった結果 s からどのエンドノードへのパスもなくなることと等しい．

5.2　ネットワークの不均一性とクエリの回数

上述の方法において，クエリの実行は，実際には，新たな接続に対するバンド幅の測定を意味する．十分安定したバンド幅を得るためには，ある程度の計測時間を必要とするから，アルゴリズムが終了するまでのクエリの回数を考察する必要がある．MeasureForChain で実行するクエリは，実際には接続のチェーンに沿って転送を行い同時に計測するので，1 回と数えることとする．完全に不均一なネットワーク上では，最悪，転送に参加しているノードの数だけクエリを必要とする．

k-heterogenous なトポロジに対し，提案アルゴリズムにおけるクエリの回数は，ε-separatable ならば，グラフの直径を L として，Lk で抑えられる．また，前節で述べたように，4.1 節の条件の下で，各転送先ノードにおける最適値 maxv からのずれは，最悪 εmaxv である．一方，定義から，ε を小さくすると k が増加することが分かる．したがって，クエリの回数と，最適値からのずれとの間のトレードオフがあって，ε の値がそれらの間をとりもつことが分かる．また，現実には，バンド幅のゆれや測定誤差があるので，ε の値はある程度大きくとる必要がある．

5.3　データの断片化と転送の方法

データが複数の送信者から転送される場合，データをチャンクと呼ばれる断片に分割して，重複がないように転送する必要がある．たとえば，BitTorrent の場合，256 KB 程度のチャンクに分割される．

Kostic らはチャンクの転送の方法を大きく分けて pull 型と push 型の 2 つに分類している[2)]．push 型では，計画的にチャンクを転送経路に割り振ることで，受信側の持っている情報を定期的に取得することなく，チャンクが送信される．一方，pull 型では，受信者側が送信者側に何らかの形でチャンクをリクエストしたり，定期的に受信側の持っている情報を送信者に伝えたりし，その情報をもとにチャンクが送信される．P2P 型のブロードキャストの場合，前もってチャンクを転送経路に割り振ることは難しいため pull 型が用いられるが，リクエストの通信コストや遅延と，送信されるチャンクの重複率との間にトレードオフがあるため，リクエストの時間間隔や正確性を工夫し，調整する必要がある．本手法の場合，構築されるオーバレイネットワークの形が次のような特徴を持つことが分かる．まず，すべてを含む 1 本のパイプライン転送が含まれているため，ソースノードを起点とした全

順関係に対応したグラフとなっている．エンドノードはその順に $d_0, d_1, \cdots, d_N$ と書ける．また，トポロジの部分木を削除してゆく形になっているため，任意の 2 つのエンドノード間の接続 $d_i \to d_j$，$d_k \to d_l$（$i < k$）に対し，$i < j < k < l$ または $i < k < l < j$ が成り立つことが分かる．これは，平面上でエッジどうしが交わらないように描けることを意味する．

この性質に基づくと，次のような方法で重複なく効率的にチャンクを転送することができる．まず，転送において送られるデータは，チャンクとそれに対応した整数値（以降 bound と呼ぶ）である．また，オーバレイネットワークが全順序グラフであるため，ソースノードから 1 から順に番号をつけることができる．以降エンドノード A に対する id_A で表す．

次に，送り方を述べる．初めにソースノード s において，各チャンク c に対する bound を次のように初期化しておく．

$\mathrm{bound}_s(c) = \infty$.

エンドノード A がチャンク c と bound x を受信したとき，bound を次のように更新する．

$\mathrm{bound}_A(c) = x$

A が別のエンドノード B に送信するとき，

$\mathrm{bound}_A(c) > \mathrm{id}(B)$

をみたすようなチャンク c を送信する．また，c を送信したとき，A は bound を次のように更新する．

$\mathrm{bound}_A(c) = \mathrm{id}(B)$.

各チャンクごとの転送経路は自然にツリーとなるため，重複は起きない．また，あらかじめ転送経路を割り振っているわけではないため，実際の転送時のバンド幅に適合的に転送経路が構築され，効率的である．

5.4 Far First

BitTorrent では，送信可能なデータチャンクが複数あるとき，Rarest First というポリシで順序を決めている．送信者が持っているチャンクの中で，その隣接ノードが持っている数が最も少ないチャンクが送信される．これは，チャンクの送信者と受信者の間で，持っているチャンクの種類をなるべく異なるものすることで，全体のスループットを向上させる効果がある．Bottleneck Skipping Algorithm で作られるオーバレイネットワークの場合でも，同様のことが求められるが，前節のような方法でチャンクを転送するため，bound の値に基づいて優先順位をつけることができる．

bound を用いた転送方法に従うと，A を送信者，B を受信者として，次の性質が満たされる．

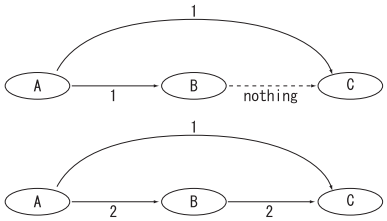

図 7　Far First
Fig. 7　Far First.

- $\mathrm{bound}_A(c) \le \mathrm{id}(B) \iff$ チャンク c は A と B の両方がすでに持っている．

したがって，A は，$\mathrm{bound}_A(c)$ が最も大きいチャンクを送信すれば，受信者と送信者の両方がすでに持っている，という状況をなるべく後ろのエンドノードでしか起きないようにすることができる．これは，受信者に送るチャンクを送信者が持っていない，という状況をなるべく少なくすることにつながる．このことを Far First と呼ぶことする．

たとえば，**図 7** のようなオーバレイネットワークを考える．はじめに A のみがチャンク $\{1, 2\}$ を持っているとする．A から C にチャンク 1 を送ったあと，A から B に 1 を送ってしまうと，B と C で同じ種類のチャンク 1 を持つことになるため，B から C に送信できるチャンクが一時的になくなる（図 7 上）．一方，far first にすると，A から C に 1 を送ったあと $\mathrm{bound}_A(1) < \mathrm{bound}_A(2)$ となるので，A から B へは 2 を送ることになり，送信できるチャンクがなくなるということはない．

5.5 エンドゲームモード

転送の終了が近くなると，送信者が持っているにもかかわらず bound の値が受信者の id よりも少ないため送ることができないケースが起こりうる．これは，安定なオーバレイネットワーク上で転送した場合は，理論的には起こりえない．実際には，バンド幅のゆれやトポロジが木でないことや，Bottleneck Skipping Algorithm では途中で逐次的に接続を追加するため，そのようなことが起こってしまうので，適切に対処する必要がある．そこで，実装では，送信者がすべてのデータを受け終わり，受信者がその送信者からどのチャンクも bound の値のために受け取ることができなくなったとき，送信者に bound の値にかかわらずリクエストすることによってこの問題に対処している．

Push 型，Pull 型の観点から本提案手法におけるチャンクのデータ転送をまとめると，次のように，Push と Pull の 2 つをもちいたハイブリッド型であるといえる．

(1)　Chunk ごとに Bound をもうけた Push 型転送.
(2)　end-mode：データを完全に持っていて 1 の方法では送るものがなくなっている相手からは持っていないものをリクエストする Pull 型転送.

6. 実　　験

6.1 シミュレーション

次のようなネットワークシミュレーションを用いて，提案手法の評価を行った．まず，次のようにして，ランダムに k-heterogenous なツリーを作る．スイッチを表すノードを順次追加し，終端ノードとなるスイッチの数が k 個になるまで追加する．この際，ネットワークがスケールフリーとなるよう，優先的選択を行った．その後，終端ノードとなるスイッチの下に 1 つずつエンドノードを追加した後，さらに，必要な数になるまで優先的選択でランダムにエンドノードを追加する（**図 8**）.

次に，複数のエンドノード間の接続が与えられたとき，次のような方法で割り当てられるバンド幅をシミュレートする．まず，バンド幅の適当な更新幅 δ を決める．各接続に対して，バンド幅に空きがあり，かつ転送元のノードでデータの流入が流出より多いとき，δ/(接続のステップ距離) だけバンド幅の割当てを増やす．すべての接続において，これ以上バンド幅の割当てを増やすことができなくなるまでそれを繰り返す．この方法は，TCP の輻輳制御のアルゴリズムを粗く近似している．TCP では，パケットロスが起きるまで，RTT の逆数に比例した速度でウインドウサイズを増やすため，RTT が接続のステップ距離で表されるとすれば，上で述べたようなバンド幅の更新幅が得られる．

エンドノードの数を 1024，各エッジに割り当てられるバンド幅を，クラスタ外では $[10, 1000]$ 上の一様乱数，クラスタ内では 1000 とし，提案アルゴリズムのシミュレーション実験を行った．バンド幅の変化を検出するための閾値は，バンド幅の比率 $(1+\varepsilon)$ のかわりに差分を用い，その値を 1 とした．また，トポロジから得られる深さ優先のパイプライン転送（Single Pipline）を実験の比較対象とした．

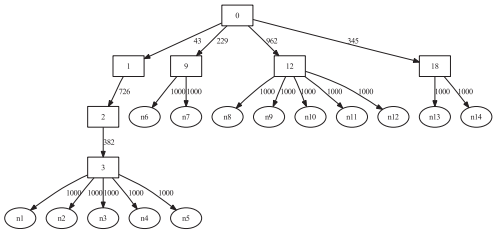

図 8　例：4-heterogenous なネットワーク
Fig. 8　A 4-heterogenous tree-shaped network.

図 9 は，接続の追加が終わるまでに必要とした試行回数（図 4 の TryAndSkip が呼び出された回数）をグラフにしたものである．k-heterogenous の k が増えるに従って，必要となる試行回数がほぼ線形に増えてゆくことが分かる．また，**図 10** は，接続の追加を行う前と行った後での各ノードに割り振られるバンド幅の，1 対 1 での転送でのバンド幅に対する割合の平均である．k が増えるに従って，得ることができるバンド幅の比率は落ちているが，Single Pipline に対しての差が広がっていることが分かる．また，256-heterogenous のとき，最適値からの割合が，Single Pipline に比べて十分良いものの，0.8 程度に落ちてい

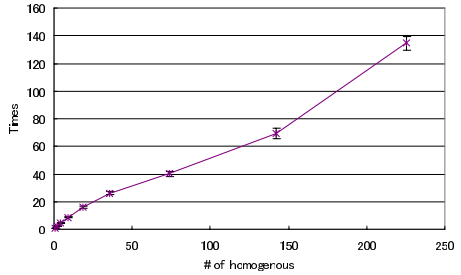

図 9　オーバレイネットワークの漸近的構築が終わるまでの時間
Fig. 9　The relationship between homogeneity and finishing time of construction for overlay network.

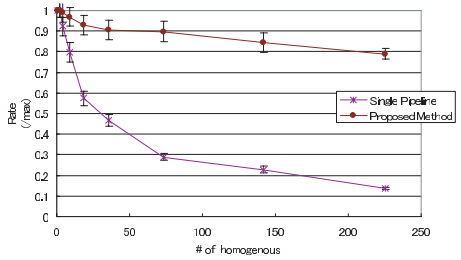

図 10　1 対 1 通信でのバンド幅の総和に対するバンド幅の総和の割合の変化
Fig. 10　The relationship between homogeneity and the rate of aggregate bandwidth to theoretical upper bound.

る．完全に安定ならばつねに 1 である必要がある．これは各接続に割り振るバンド幅を完全に決定しているわけではなく，TCP の持つロードバランスの結果にまかせているため，前提条件が厳密には成り立っていないことを意味している．

6.2　実ネットワークにおける実験

多拠点クラスタ環境として，InTrigger [12] を使って実際の実験を行った．InTrigger は日本の各所に設置された十数個のクラスタからなっている．バンド幅の変化を検出するための閾値は，前節と同様に，差分を用い，それを 10 Mbps とした．

図 11 は，クラスタ数を変えたときの平均の転送終了時間の比較である．Single Pipeline と比べると，つねにほぼ同じか優位であることが分かる．これは，提案手法では SinglePipeline からスタートし，ほとんどの場合において，バンド幅の総和が大きくなるような接続しか追加しないためである．使われたクラスタ内のネットワークの構成から，クラスタ数は heterogenous の数とほぼ同じと考えてよい．クラスタの数が増えると Single Pipline の方法よりも，提案方法のほうが効率的な転送が行えていることが分かる．**図 12** は 5 クラスタ上での 1 つの転送において，各ノードで転送が終了するまでにかかった時間を表している．Single Pipline では，転送レートは単調に低下するが，提案手法では転送レートがクラスタに依存して適応的に決まっていることが分かる．これは漸近的に安定なオーバレイネットワークに近づくためである．

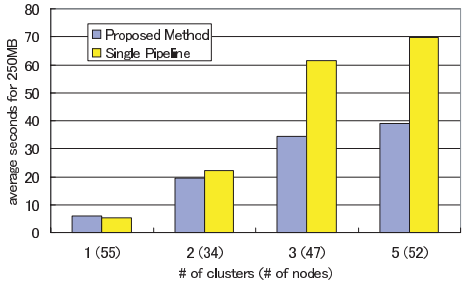

図 11　複数クラスタでの平均転送時間の比較
Fig. 11　Average transferring time for multi clusters.

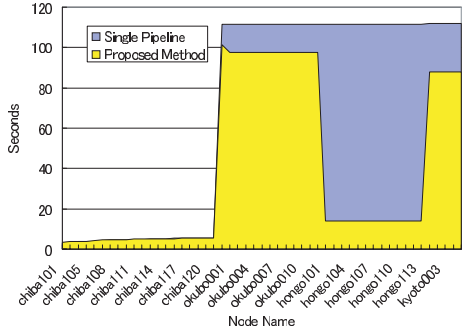

図 12　各ノードごとの転送時間の比較
Fig. 12　Transferring time for each node in 5 clusters.

7. 結　　論

バンド幅が未知であるようなネットワークトポロジ情報を用いた，大規模データのブロードキャストに関する新しい方法を提案した．提案した手法は，いくつかの条件の下で 4 章で述べた意味で漸近的に安定になる．また，ネットワークの不均一性を k-heterogenous という数で表したとき，提案した方法は，主に k がノード数に対してあまり大きくないとき特に有効であることを示した．多拠点クラスタの場合がこれに相当する．実際の多拠点クラスタを用いた実験では，広域ネットワークで実効バンド幅が未知であっても，適応的に接続を追加するため，より安定的で効率的な方法であることが確かめられた．

一方で，広域ネットワークがツリーからかけ離れている場合や，バンド幅がブロードキャスト中に大きく変動する場合や，アップリンクとダウンリンクのバンド幅の違いなどの影響で，提案手法が前提としている条件が大きく崩れる場合，安定性は保障されない．しかし，提案手法では SinglePipeline からスタートし，ほとんどの場合においてバンド幅の総和が大きくなるような接続しか追加しないため，多くの場合 Single Pipline よりも良く，最悪の場合でも Single Pipline の場合と同等の性能は保障される．

謝辞　本研究は特別推進研究「高度言語理解のための意味・知識処理の基盤技術に関する研究」の助成を受けて行われた．

参　考　文　献

1) Kostic, D., Rodriguez, A., Albrecht, J. and Vahdat, A.: Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh, *Proc. SOSP*, pp.282–297 (2003).
2) Kostic, D., Snoeren, A.C., Vahdat, A., Braud, R., Killian, C.E., Anderson, J.W., Albrecht, J.R., Rodriguez, A. and Vandekieft, E.: High-bandwidth data dissemination for large-scale distributed systems, *ACM Trans. Comput. Syst.*, Vol.26, No.1 (2008).

3) den Burger, M. and Kielmann, T.: MOB: zero-configuration high-throughput multicasting for grid applications, *Proc. HPDC*, pp.159–168 (2007).
4) Izmailov, R., Ganguly, S. and Tu, N.: Fast Parallel File Replication in Data Grid, *Future of Grid Data Environments workshop* (*GGF-10*), Berlin, Germany (2004).
5) Shah, P. and Paris, J.-F.: Peer-to-Peer Multimedia Streaming Using BitTorrent, *Proc. IPCCC*, pp.340–347 (2007).
6) Takahashi, K., Saito, H., Shibata, T. and Taura, K.: A Stable Broadcast Algorithm, *Proc. CCGrid*, pp.392–400 (2008).
7) Shirai, T., Saito, H. and Taura, K.: A Fast Topology Inference — A building block for network-aware parallel computing, *Proc. HPDC*, pp.11–21 (2007).
8) 長沼　翔，高橋　慧，斎藤秀雄，柴田剛志，田浦健次朗，近山　隆：ネットワークトポロジを考慮した効率的なバンド幅推定手法，*SACSIS 2008*, pp.359–366 (2008).
9) Cohen, B.: Incentives build robustness in BitTorrent, *Proc. Workshop on Economics of Peer-to-Peer Systems* (May 2003).
10) Androutsellis-Theotokis, S. and Spinellis, D.: A Survey of Peer-to-Peer Content Distribution Technologies, *ACM Computing Surveys*, Vol.36, No.4, pp.335–371 (2004).
11) Feldman, M., Lai, K., Stoica, I. and Chuang, J.: Robust Incentive Techniques For Peer-to-Peer Networks, *Proc. ACM Electronic Commerce*, pp.102–111 (2004).
12) InTrigger. https://www.intrigger.jp/
13) Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A. and Singh, A.: Splitstream: High-bandwidth content distribution in cooperative environments, *Proc. SOSP* (2003).
14) 蓬莱祐一郎，西田　晃，小柳義夫：木構造型ネットワークにおける最適ブロードキャスティング，情報処理学会誌：コンピューティングシステム，Vol.45, No.Sig3 (ACS 5), pp.100–108 (2004).
15) den Burger, M., Kielmann, T. and Bal, H.E.: Balanced Multicasting: High-throughput Communication for Grid Applications, *Proc. ACM/IEEE SC2005* (2005).

(平成 21 年 1 月 27 日受付)
(平成 21 年 5 月 1 日採録)

柴田　剛志（正会員）

1978 年生．2007 年東京大学大学院工学研究科電子工学専攻博士課程修了．工学博士．2007 年より同大学にて研究員．グリッドにおける大規模計算に関する研究に従事．人工知能学会会員．

田浦健次朗（正会員）

1969 年生．1997 年東京大学大学院理学博士（情報科学専攻）．1996 年より東京大学大学院理学系研究科情報科学専攻助手．2001 年より東京大学大学院情報理工学系研究科電子情報学専攻講師．2002 年より同助教授．並列・分散処理，プログラム言語に興味を持つ．ACM，IEEE，ソフトウェア科学会各会員．