Vol.2009-HPC-121 No.29
2009/8/6

大規模ネットワークにおける バンド幅測定アルゴリズム

長沼　翔[†1]　田浦　健次朗[†1]

広域分散システム上で分散アプリケーションを実行する場合，データインテンシブなタスクのようにデータ転送が全体の処理のボトルネックとなることがある．通常，リンクのバンド幅は場所によって大きく異なるので，データ転送を非効率にならないよう計画的に行う必要がある．この為にはネットワークトポロジーとバンド幅を結びつけた，バンド幅マップの情報が欠かせない．既存の手法では測定に時間がかかるうえ，バンド幅マップのような詳細な情報は得られない．本論文では，まず求めるべきリンクのバンド幅を定義し，バンド幅マップを構築するアルゴリズムを示す．提案するアルゴリズムではネットワークトポロジーを利用してバンド幅測定を進めることで高速性と正確性を追求している．

Bandwidth Measurement Algorithm for Large-Scale Networks

Sho Naganuma[†1] and Kenjiro Taura[†1]

Data transfer is a bottleneck to excute data intensive applications in distributed environments. The bandwidth of each link there varies from place to place, however, it is necessary to perform the data transfer efficiently and systematically. To do this, the Bandwidth Map is required, which is information of the network topology combined with the values of bandwidth. Some existing methods to measure bandwidth take a long time to be accomplished, and what is worse, we cannot know details such as the Bandwidth Map. In this paper, we start with to define what is the bandwidth, and then we propose an efficient and accurate algorithm of measuring and building the Bandwidth Map taking a network topology into account.

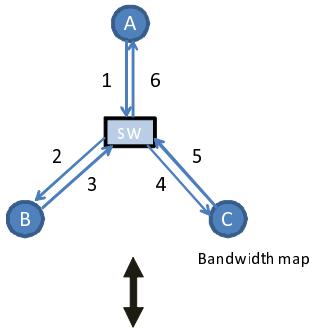
1. はじめに

コンピュータネットワークの通信性能の指標として一つにバンド幅がある．バンド幅とは一秒あたりどれほどの情報量を送受信できるかを表す数字で，単位は bit/sec である．一般的にバンド幅測定はあるコンピュータから別のコンピュータへ大きいデータを送信し，そのデータサイズを送受信にかかった時間で割るというナイーブな手法が用いられる．バンド幅を知るということは，そのネットワークの使用者にとって非常に重要である．例えば多人数参加型の映像配信を行う際には，配信者は自分の使用できる最大のバンド幅を越えるビットレートで映像を配信してはサービスとしてのクオリティは低下し，更に受け手側ではそれぞれの使用できるバンド幅に見合った値に受信ビットレートを調節してやらなければサービスを快適に受けることはできない．近年においては分散システムの登場によりバンド幅測定の重要性が高まった．分散システムとは多数のコンピュータがネットワーク接続された環境で，それらのコンピュータを協調させて一つのタスクを実行することでシステム全体の処理速度や頑健性を高めることができる．この際，通信性能の低い箇所にあるコンピュータばかりに仕事を割り当てるなどしてしまうとシステム全体の処理速度や頑健性は思ったように上がらないのは明らかであり，特にデータ転送がボトルネックとなるデータインテンシブな分散アプリケーションを実行するにはそれに先立って各所のバンド幅を測定し，ネットワークを考慮してうまく仕事を振り分けてやることが望ましい．これらの例のように，バンド幅測定は，ネットワーク越しに行われる何らかの分散アプリケーションの性能を最適化するための下地として用いられることが多く，本研究を含むバンド幅測定の研究全体が目標とし，貢献を目指すものである．

測定したバンド幅の表現方法には，バンド幅マップとバンド幅マトリックスの二つがある．バンド幅マップとは，測定対象としているネットワークのグラフのエッジにバンド幅の値を重みとして付加した，重み付き有向グラフである．バンド幅マトリックスとは，測定対象としているネットワークに参加している全エンドホストに対して，全ての組み合わせのバンド幅測定の結果を 1 セルとした N × N のテーブル（N はエンドホストの数）となる．例として，三つのエンドホストが一つのスイッチでスター型に接続された，最も単純なネットワークを，バンド幅マップとバンド幅マトリックスの両方で表現したものを**図 1** に示す．

†1 東京大学
University of Tokyo

Vol.2009-HPC-121 No.29
2009/8/6



Snd ＼ Rcv	A	B	C
A	-	1	1
B	3	-	3
C	5	2	-

Bandwidth matrix

図 1 簡単なバンド幅マップとバンド幅マトリックスの例
Fig. 1 Examples of Bandwidth Map and Bandwidth Matrix for a Simple Network

図中の数字はバンド幅の値を表している．

バンド幅マップとバンド幅マトリックスを比べると，前者の方が情報量は多く，取得しておく価値が高い．図 1 からも明らかに，同じ測定対象のネットワークでもバンド幅マップとバンド幅マトリックスとでは，前者の方が情報量が多い．例えば図 1 中のバンド幅マトリックスでは，スイッチからエンドホスト A に向かうエッジのバンド幅 6 と，スイッチからエンドホスト C に向かうエッジのバンド幅 4 が見えていないが，バンド幅マップにはそれらの値が現れている．更に，ホスト B, C の二つからホスト A へ同時にネットワークストリームを流すような場合を考えると，バンド幅マップから読み取るにスイッチからホスト A へ向かうエッジのバンド幅が 6 であるために，二つのストリームの合計バンド幅は 6 に律速される．このようなことはバンド幅マトリックスからでは判断することができず，システムに適さないデータ転送手順を踏ませてしまう可能性も生じる．よってバンド幅マトリックスではなくバンド幅マップを得ることができなければ，前に例で挙げた多人数参加型映像配信や分散アプリケーションなどの最適化に用いることはできず，研究の目標を達成することはできない．

しかし，バンド幅マップを得ることはバンド幅マトリックスを得ることよりも一般的に難しい．ネットワークには通常複数のスイッチやルーターが存在し，それらは通信の末端になることができないためである．通信の末端とは，エンドホストのようにセンダーやレシーバーとなって，通信パケットを受け止めることができるノードである．現在広く用いられているバンド幅測定手法は，二つのエンドホスト間でデータ転送を行い，かかった時間で割るというものであるが，二つのエンドホスト間には一つ以上のスイッチ，すなわち二つ以上のエッジが存在することがほとんどである．既存手法のような End-to-End の手法では，複数の測定すべきエッジがその End-to-End 間にあるにもかかわらず，結果は一つしか得ることはできない．加えて言及すると，その一つの結果とは経路中の複数のエッジのうちバンド幅が最小のものの値（律速されるため）であり，どのエッジにその重みを付加するべきかは不明のままである．すなわち既存手法のようなナイーブな手法ではバンド幅マップを得ることは不可能である．逆にバンド幅マトリックスを得ることは簡単である．既存手法の End-to-End 測定を N(N-1) 回行えばよい（N はエンドホスト数.）ただし，たった今述べた問題があるため，バンド幅マップより情報量のそぎ落とされたものとなり，ネットワークやシステムの規模が大きくなるにつれて，そぎ落とされる情報の量も大きくなる．

そこで本研究はネットワークのバンド幅マップを取得する手法の考案を目的とする．本稿ではまずいくつかの既存手法を紹介し，それらの問題点を指摘する．次に本研究で扱うエッジのバンド幅という概念を定義しなおし，アルゴリズムの提案をする．ところで，コンピュータネットワークの通信性能を表す指標としてバンド幅の他に遅延時間がある．バンド幅が，メッセージを受け取り始めた瞬間から受け取り終わる瞬間までの時間に関する性能指標であるのに対して，遅延時間とは，コンピュータがメッセージを送信した瞬間から相手のコンピュータがメッセージを受け取り始める瞬間までの時間を言う．遅延時間は，send や receive のシステムコールの呼び出しや経路中のネットワーク機器のパケット受け渡しなどにかかる微小の時間が積み重なることによって生じる．遅延時間を測定する一般的な手法は，数バイトの小さいメッセージを ping-pong し，かかった時間を二で割るという操作である．一般的に遅延時間の値はマイクロ秒からミリ秒の単位で，バンド幅測定で扱う時間感覚と比べてはるかに小さい．例えば現在最も普及しているギガビットイーサを用いて 100 メガバイトの文書を送受信する場合には 1 秒近くかかる．つまり，大きいメッセージの送受信を行う通信アプリケーションにとって遅延時間は無視できる範囲であり，本研究の目標として例に挙げた映像配信や分散システム上の分散アプリケーションでも遅延時間は現実的に無視できるものとして扱ってよく，故に本研究は重みとしてバンド幅だけを考えたバンド幅

マップが構築できればよいものとする．参考までに，重みとして遅延時間を考えた所謂遅延時間マップを構築するアルゴリズムの研究も存在する[3]．

本稿で扱う用語やグラフ構造の定義を以下に示す．本稿では実際のネットワーク上のスイッチやハブ，ルーターなど，通信の分岐点となりうるネットワーク機器を総じてスイッチと呼ぶことにする．また，分岐に関与しないスイッチ，すなわち二本のリンクしか接続されていないスイッチは取り除き，それに接続されていた二本のリンクは一本の仮想リンクとする．この操作は目標達成に適したバンド幅マップを構築するための条件を崩さない．想定するバンド幅マップのグラフ構造において，エッジは実際のネットワークのリンクもしくは仮想リンクを表し，エッジの向きとそれらの重みはそれぞれリンクの上り下りのバンド幅の値とする．実際のネットワーク上のスイッチとエンドホストがグラフ上のノードとなり，グラフ上の末端ノードはエンドホストのみ，中間ノードはスイッチのみが配置されうるとする．

この節の終わりに，バンド幅マップを応用した分散アプリケーション，ブロードキャストの研究を具体例として挙げる．ブロードキャストとは分散システムでよく行われる処理であり，計算に参加している自分以外の全エンドホストへデータコピーを送りつける操作である．一つのホストから他の多数のエンドホストへデータを配る場合，一対多にコネクションを張って転送していては送信元のノードから出る一本のエッジはもちろん詰まってしまう可能性が高いし，経路中にあるエッジも複数の転送に共有されて詰まってしまう可能性があり，システム全体のパフォーマンス低下や頑健性低下に繋がってしまう．そこで予めバンド幅マップ，送信者と受信者の集合が分かっていれば，転送のタイミングを制御してやることによって経路中の各エッジを詰まらせない範囲で最大限のスループットを出す手法を考えることができる．高橋らの研究では，エッジの詰まりを防ぎつつもそのエッジのバンド幅を最大限に利用するよう，複数のデータ転送パイプラインを設けることでブロードキャストのパフォーマンス向上を実現した[4]．高橋らの手法はまず全エンドホストに行き渡るスパニングツリーを作る. このツリーに従ってパイプライン転送をしたとすると，ツリー中の最小のバンド幅を持つエッジに律速されることは明らかである. 次に，まだキャパシティを持て余しているエッジとノード群で成る部分トポロジーから同様にパイプライン部分木を作り，木を作ることができなくなったらそれらに沿うパイプラインに並列にデータを流すことで実現する. 吉富らの研究では遅延時間マップとバンド幅マップを利用することでブロードキャストの性能を向上させる手法が挙げられた[8]．この研究では，メッセージの送受信にかかる時間はメッセージ長と経由するネットワークの遅延とバンド幅で一意的に決定できるとし，ネットワーク性能に見合わせてブロードキャストアルゴリズムを切り替えることによりスループットの改善を実現している．

2. 関 連 手 法

この節ではバンド幅マップを構築するという筋で関連し，参考にすることができる既存手法を紹介する．

2.1 Iperf

Iperf[5] は，二つのエンドホスト間の通信スループットを測定するプログラムである．Iperf の原理を次に説明する．プログラムがサーバーとクライアントに分かれていて，クライアントはサーバーへ 10 秒の間可能な限りのメッセージを送信し続け，サーバーは受信することができたビット数を 10 秒で割ることで通信スループットを計算する．ここで通信スループットと記述したのは，二つのエンドホスト間に複数のエッジが存在する場合，Iperf はそれらのエッジから一つを特定してバンド幅を測定することができないためである．理由は，通信経路中の全てのエッジのうち最もバンド幅の低いエッジに律速されてしまうからである．すなわち観測される値は複数のエッジのもつバンド幅のうち最小の値のみであり，かつそのバンド幅を持つエッジがどれなのかは判別できない．よって，Iperf ではバンド幅マップの構築は不可能である．Iperf の設計はバンド幅マップを構築するということまでは考えられていないが，現在最も普及している通信性能測定プログラムであり，原理が単純で分かりやすくかつ本研究の考え方の基本ともなる手法でもあるので，既存手法として紹介した．以後本稿では Iperf と同等の原理で二つのエンドホスト間の測定を行うことを単に「Iperf を実行する」と記述する．

2.2 Network Weather Service

Network Weather Service[7][6]（以降 NWS）は，グリッド環境などで広く用いられている分散システム向けの統合ネットワーク監視システムである．ネットワークの接続性の監視や各エンドホスト間での遅延時間の測定など，ネットワークに関連する様々なサービスを提供しているが，その中でもバンド幅測定に関連するサービスモジュールを紹介する．NWS の原理を次に説明する．システム内の全てのエンドホストに NWS デーモンが起動し，ホストの全ての組み合わせ N(N-1) ペアに対して Iperf を逐一実行し，システムのバンド幅マトリックスを取得する．ここで Iperf を逐一実行しなければならない理由は，測定するために流すネットワークストリームが衝突するという性質を持つためである．Iperf を原理とした一対一測定では一つのエッジに同時に二つ以上のネットワークストリームが流されるとそれらの通信が競合し，両測定の結果が誤ってしまう．NWS ではバンド幅マトリックスの測

Vol.2009-HPC-121 No.29
2009/8/6

定に際して，ネットワークストリームの衝突を避けるために，システム内の全ホストで唯一のトークンを受け渡しするトークンリングを生成し，トークンを所持しているホストのみが測定ペアの一方となることができるという制約を設けている．この手法では，明らかに N × N のバンド幅マトリックスしか得ることができず，我々が目標としているバンド幅マップの取得をするには不十分なものである．更に言及すると，グリッド環境などの複数のサブネットが組合わさる大規模なシステムでは，バンド幅マトリックスの持つ意味は小さくなっていく．なぜならばそのような環境ではネットワークの接続形態も非常に複雑なものとなる傾向が強く，そうなると測定対象のネットワークグラフの直径が肥大化し，二つのエンドホストペアの間に存在するスイッチの数が多くなるからである．Iperf を行うペアの間のスイッチ数が多くなればなるほど，バンド幅マトリックスはバンド幅マップと比較して抜け落ちる情報が多くなってしまう．

2.3 TopoMon

TopoMon[1] は分散システムのバンド幅マップを構築するということに着目した初期の研究で，NWS の拡張モジュールとして動作する．この研究の目的や追求する結果の方向性は我々の研究とほぼ同じである．バンド幅マップ構築アルゴリズムは以下のようになっている．まずネットワークトポロジーのすべてのエッジのバンド幅を 0 とする．次にバンド幅マトリックスのあるセルを参照し，そのセルに対応するエンドホストペアについて，ペア間の通信で経由するすべてのエッジのバンド幅の値を更新する．この時，既にそのエッジに振られていた値よりセルの値のほうが小さければ更新は行わない．これをすべてのセルについて行う．ネットワークトポロジーは NWS デーモン間の traceroute で取得し，バンド幅マトリックスは上で述べた N(N-1) ペアに対する Iperf 測定によって取得する．この手法ではバンド幅マトリックスとネットワークトポロジーが手元にあればローカルで計算することができるので NWS の追加モジュールとしては非常に実用性のある実装である．しかし，論文中でも指摘されているとおり，測定ペア間のスイッチのホップ数が多くなればなる程，または測定対象のエッジがネットワークグラフの中心に近いほど，経路途中のボトルネックリンクの影響をうけやすく，誤ったバンド幅の値を割り振ってしまいがちになる．Mathijs den Burger らが論文の最後で Future work として述べている通り，Iperf の重ねがけでエンドホストより遠い位置にあるエッジのバンド幅を埋め尽くすことによりこの誤りを回避することができる．次に述べる bhtree や本研究はこのヒントを汲んでいる．

2.4 bhtree

bhtree[2] は，我々が行った先行研究である．本研究と目的や目標は同じとして大規模ネットワークのバンド幅マップを取得するアルゴリズムの提案をした．得られた結果としては，300 ホスト程がネットワーク接続された広域分散環境において，およそ 2 分程度でバンド幅マップの構築が可能であった．bhtree の実験で得られたバンド幅マップを可視化したものを**図 2** に示す．図中の灰色のノードがスイッチでオレンジ色のノードがエンドホストを表す．また，各所に記入してある数字は各スイッチまたはホスト間を結ぶエッジのバンド幅（Mbps）である．

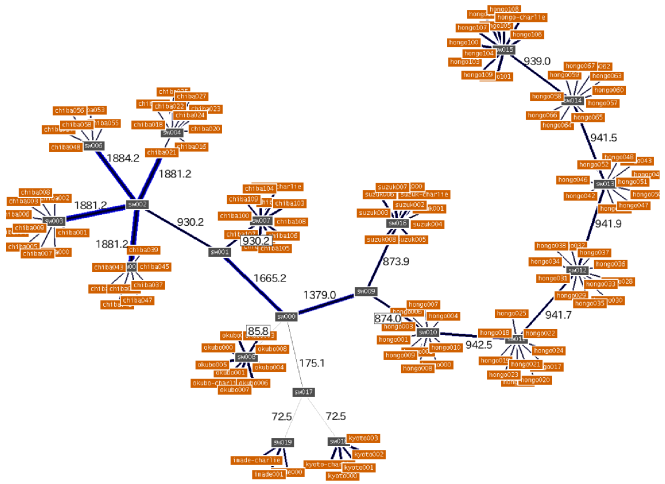


図 2 ***bhtree*** の出力
Fig. 2 Result of ***bhtree***

bhtree は与えられたネットワークトポロジーを細かく分解し，測定の競合しない部分トポロジーを並列に測定することで高いスケーラビリティを得ることができた．しかし，アルゴリズムの特性上，ネットワークトポロジーは向きの無いツリー構造を仮定していた．すなわち，実際のネットワークには多く存在するはずのバンド幅非対称なエッジや循環構造をもつネットワークに対して正しい答えを出すことができなかった．更に，後述の “バンド幅のキャパシティを埋めきる” という概念にやや不十分な点があり，実際のバンド幅の値より小さい値を記録してしまうエッジが含まれた結果を出しており，そもそも何をもってエッジのバンド幅とするかの定義が曖昧なままであった．

本研究は bhtree の発展型とも言える．本研究では実際のネットワークにより近いデータ

構造である有向グラフを入力データとしており，各エッジのそれぞれの向きにバンド幅の値を重みとして付加した重み付き有向グラフを出力する実装系を目標とする．また，エッジの向き付きバンド幅という概念は数々の既存手法でも聞き慣れない概念であったので，第 3.1 節で示す通り，本研究では何をもってバンド幅と決定してよいかを明確に定義してから話が進む．また，bhtree の実装系ではネットワークの直径やスイッチの分岐数に応じてアドホックな処理を行う部分もあったのでアルゴリズムとして決して見通しの良いものではなく一般性に欠けていた．本研究ではより見通しの良いアルゴリズムを考案し，一般のネットワークにも適応可能なものを目指す．

3. 提案手法

この節では，バンド幅マップを取得するための我々の提案手法を紹介する．まず初めに我々が追求する，エッジの向き付きバンド幅とは何かを定義し，その正当性を確認する．次に，その定義を満たすようなバンド幅で構築されるバンド幅マップの取得アルゴリズムの詳細を述べる．なお，本手法は前提としてネットワークのトポロジー情報とパケットのルーティング情報は既知でとする．すなわち，全てのエンドホストとスイッチの接続形態と，任意の二つのエンドホスト同士の通信経路は，本手法の実装系の入力データとなる．これらの情報は traceroute や既存のトポロジー推定手法[3] を用いれば容易に取得することができる．実装系の出力は，入力された有向グラフのエッジにバンド幅の値という重みを付加した，重み付き有向グラフとなる．

3.1 バンド幅の定義

ここでは我々が求めるべき，エッジの向き付きバンド幅を定義する．2 節で紹介した手法も含め，これまでの手法でエッジに向き付きバンド幅を関連付けるという概念はあまりなかったと思われる．Iperf や NWS では測定に関与する二つのホストペアに対してバンド幅が関連付けられていると言える．また，bhtree ではエッジにバンド幅を関連付けてはいるが，向きは考慮していない．そこで，我々はバンド幅マップ取得アルゴリズムの考案に先立って，エッジの向き付きバンド幅というものを明確に定義し，その正当性を確認する必要がある．

まずエッジの向き付きバンド幅はどのように測定されるべきかを以下の様に決める．今，バンド幅の値が知りたい一つのエッジの一つの向きに着目しているとする．そのエッジの向き付きバンド幅とは，システム中の N(N-1) 組ある Iperf 測定ペアのうち，測定経路中そのエッジをその方向に流れることになる測定ペアを全て同時に流し，それらの観測結果の合計

G is a network graph.
$\boldsymbol{H}$ is a set of the all end-hosts in G.
$\boldsymbol{E}$ is a set of the all edges in G.
$\boldsymbol{P}$ is a set of the all (snd, rcv)-pairs consisting of the members of $\boldsymbol{H}$.
$\boldsymbol{P}_{e_k}$ is a subset of $\boldsymbol{P}$ such that the packet-flow between $p \in \boldsymbol{P}_{e_k}$ traverses via $e_k \in \boldsymbol{E}$.

We get B_{e_s}, the bandwidth of edge e_s, as following.
Invoke plural measurements of all pairs in $\boldsymbol{P}_{e_s}$ simultaneously;
$O_p \leftarrow$ Store the value of observed bandwidth of the measurement between $p \in \boldsymbol{P}_{e_s}$;

$$B_{e_s} := \sum_{p \in P_{e_s}} (O_p) \qquad (1)$$

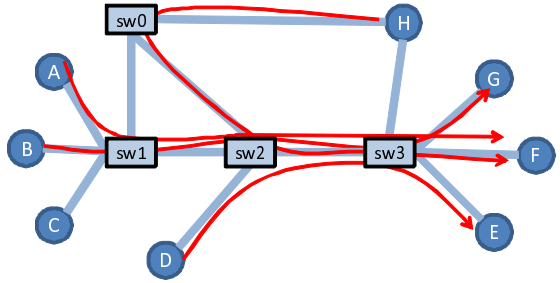
図 3 バンド幅の定義
Fig. 3 Definition of Bandwidth

をとったものである（N はシステム中の全エンドホスト数.）まとめると**図 3** の様になる．G をネットワークグラフ，$\boldsymbol{H}$をエンドホスト全体の集合，$\boldsymbol{E}$をエッジ全体の集合，$\boldsymbol{P}$を測定ペア全体の集合とする．

以下，図 3 に沿って説明する．(1) 式で e_s のバンド幅 B_{e_s} を決定してよい理由は以下の様である．まず，e_s のバンド幅を測定したいのであるから，$\boldsymbol{P}$のうち e_s に関与する測定ペアの集合 $\boldsymbol{P}_{e_s}$ のみを考えればいいことは明らかである．次に，$\boldsymbol{P}_{e_s}$ のうちの少数の測定ペアのみを選んでそれらを同時に測定したのでは e_s のバンド幅のキャパシティを埋めきることができず，正しいバンド幅より小さい値が観測されてしまう可能性があるので，$\boldsymbol{P}_{e_s}$ の全ての測定ペアを同時に測定しそれらの合計を答えとする必要がある．ここで，もし e_s のバンド幅のキャパシティを埋めきることができたと判断できれば，それを e_s のバンド幅だと決定できる．逆に，$\boldsymbol{P}_{e_s}$ 全ての測定ペアも使ってもなお e_s のバンド幅のキャパシティを埋めきることができないことも考えられる．しかしその場合であっても，システム全体を導引してでさえ e_s のバンド幅は使い切ることができないほど大きいということを意味するので，e_s のバンド幅は上と同様に決定しても実用上問題はないと言える．もしくは出力の段階で，この値以上，と注釈を付けることは容易にでき，実用上誤った答えを出すことは無い．以上によって，(1) 式で求めたバンド幅 B_{e_s} は，エッジ e_s のバンド幅であると決定してよい．

ここで挙げた定義に沿ったバンド幅測定のイメージ図を，一つ例として**図 4** に示す．図 4 の説明をする．今，sw2 で表されるスイッチと sw3 で表されるスイッチの間のエッジについて，sw2 から sw3 の向きのバンド幅を測定しようとしているとする．その向き付きのエッ

Vol.2009-HPC-121 No.29
2009/8/6



e_{target} : (sw2 -> sw3)
$P_{e_{target}}$: { (A -> F), (B -> G), (D -> E), (H -> F) }
$B_{e_{target}}$:= $w + x + y + z$

p ∈ $P_{e_{target}}$	routing	observed bandwidth
A -> F	A -> sw1 -> sw2 -> sw3 -> F	w
B -> G	B -> sw1 -> sw2 -> sw3 -> G	x
D -> E	D -> sw2 -> sw3 -> E	y
H -> F	H -> sw0 -> sw2 -> sw3 -> F	z

図 4 バンド幅測定の例
Fig. 4 Example of Bandwidth Measurement

ジを e_{target} と書いている．ここには A ...H の 8 つのエンドホストがあるので，全ての通信可能なペアは 56 通りあることになる (これを$\boldsymbol{P}$と書く)．そのうち，sw2 から sw3 の向きにパケットがルーティングされる通信ペアが，図中のように 4 通りのみであったとする (これを $\boldsymbol{P}_{e_{target}}$ と書いている．$\boldsymbol{P}_{e_{target}}$ は$\boldsymbol{P}$の部分集合である)．このとき，e_{target} のバンド幅 $B_{e_{target}}$ は，それら 4 ペアのバンド幅測定を同時に行ったときの観測値の和（ここでは $w + x + y + z$）で表される．

図 4 からも分かる様に，ネットワークトポロジー上の位置的な関係は測定ペアの決定に際して意味をなさない．すなわち，注目しているエッジの下流側のスイッチに近く位置しているエンドホストであっても測定に際してレシーバーとなりうるし，上流側のスイッチに近く位置しているエンドホストであってもセンダーとなる可能性もある．

3.2 アルゴリズム

ここではバンド幅マップ取得手法の全体的なアルゴリズムについて述べる．3.1 ではネットワークグラフの内の一つのエッジ，向きについてバンド幅を決定することを述べた．バン

$\boldsymbol{E}$ is a set of the all edges in network graph.
$\boldsymbol{P}_e$ is a set of (snd, rcv)-pairs such that the packet-flow between $p \in \boldsymbol{P}_e$ traverses via $e \in \boldsymbol{E}$.

for each edge e in $\boldsymbol{E}$ **do**
  Invoke plural measurements of all pairs in $\boldsymbol{P}_e$ simultaneously;
  $O_p \leftarrow$ Store the value of observed bandwidth of the measurement between $p \in \boldsymbol{P}_e$;
  $B_e := \sum_{p \in P_e} (O_p)$;
end for

図 5 バンド幅マップ構築アルゴリズム（1）
Fig. 5 An Algorithm to Build Bandwidth Maps (1)

ド幅マップを取得するということは，入力されたネットワーク有向グラフの全てのエッジに重みとしてバンド幅の値を付加するということであるので，非常に単純に考えて**図 5** を実行すれば達成することができる．すなわち，全てのエッジ，向きに対して (1) 式で表される測定を実行する．

しかし，図 5 の方法では一つのエッジを測定するたびにシステム全体を導引しなくてはならないため，非効率でありネットワークにかかる負荷も大きい．これでは現実的な手法とは言うことができないため，図 3 の定義を崩さない範囲で，手順と負荷を削減したアルゴリズムを考える必要がある．

まず図 3 の定義の意味を崩さない範囲で，ネットワークストリームを束ねる本数を削減する方法が考えられる．図 3 では，エッジ e_s のキャパシティを埋めにかかるために初めから $\boldsymbol{P}_{e_s}$ の全てのペア間でネットワークストリームを発生させ，大きな束を作って測定を行うよう記述してあるが，$\boldsymbol{P}_{e_s}$ 全てを使わずとも e_s のキャパシティを埋めきれる可能性もある．その場合，$\boldsymbol{P}_{e_s}$ から埋めきる分のみのネットワークストリームの束を発生させるようにすれば，システムにかかるネットワーク負荷を軽減することができ，それの為に導引するエンドホスト数も削減することができる．これを行うには，以下のようにする．$\boldsymbol{P}_{e_s} = (p_0, p_1, ..., p_n)$ とし，p_0 の (snd, rcv)-pair から $p_1, p_2, ...$ と一つずつ順に束ねるネットワークストリームの本数を増やすようにする．すると，あるペア p_m にストリームを流す時，$\sum_{p\ in\ \{p_0,...,p_{m-1}\}}(O_p) = \sum_{p\ in\ \{p_0,...,p_m\}}(O_p)$ となることがある．このような m が存在した場合，それ以降の $p_{m+1}, ..., p_n$ 間のネットワークストリームを束に加える必要はない．なぜならば $p_0, ..., p_{m-1}$ を束ねたネットワークストリームだけで観測されるバンド幅は頭打ちとなり，それ以上束ねても同じ値が観測されるだけであるからである．この

手法を，図 3 の定義に従うように記述するとしたら，以下のように考えることができる．$\boldsymbol{P}_{e_s} = (p_0, p_1, ..., p_n)$ において p_0 から一本ずつ流すストリームを束ねていき，p_m を束ねた時点での測定観測値の合計と p_{m-1} までの合計値が等しくすなわち観測値が頭打ちになったとしたら，残りの $p_{m+1}, ..., p_n$ の (n-m) 本のストリームは観測値ゼロで仮想的に流しているとする．この時点で $\boldsymbol{P}_{e_s}$ の全てのペア間でネットワークストリームを発生させ束ねていることになるので，定義図 3 の (1) 式より，観測値の和を e_s のバンド幅として決定することができる．以上のようにして，バンド幅測定の定義を崩さない範囲でシステムにかけるネットワーク負荷を軽減する方法が考えられた．しかしこれには気をつけるべき点がある．観測値の合計が頭打ちになるということが，着目しているエッジ e_s に因るものとは限らないからである．それは，束ねているストリームがエッジ e_s 以外の箇所でエッジを共有している場合に起きる．図 4 の例でその場合を説明する．まず初めに A から F にネットワークストリームを流し，そのバンド幅を観測する．次に B から G へのネットワークストリームをそれに加え，両者のバンド幅の観測値の和に変化が無かった，すなわちこの時点で頭打ちとなったとする．しかしこの段階では e_{target} が原因で値の頭打ちが生じたと言い切ることはできない．なぜならば今束ねている二つのネットワークストリームは，e_{target} の他にも (sw1 → sw2) のエッジを共有しているためである．この場合，どちらのエッジが埋まりきったことで値の頭打ちが生じたかは判断することができない．

以上の問題点を踏まえて手順を書き直すと，以下のようになる．今，エッジ e_s のバンド幅を測定しようとしている．$\boldsymbol{P}_{e_s} = (p_0, p_1, ..., p_n)$ において p_0 から一本ずつ流すストリームを束ねていき，p_m を束ねた時点で合計値が頭打ちになったとする．ここで，$(p_0, ..., p_m)$ の，全てのペアの通信経路に出現するエッジの集合を$\boldsymbol{L}$とする．残りの $(p_{m+1}, ..., p_n)$ の (n-m) 本の測定ペアのうち，$\boldsymbol{L}$のエッジを全てその経路中に含むような測定ペアは観測値ゼロとして仮想的に束に参加しているとする（頭打ちになることが確定しているため）．以降，まだ束に参加していない $\boldsymbol{P}_{e_s}$ の測定ペアから一本ずつストリームを束ねていく作業を再開し，$\boldsymbol{P}_{e_s} = (p_0, p_1, ..., p_n)$ の全てのペア間を流して束を作り終わるまで繰り返す．$\boldsymbol{P}_{e_s} = (p_0, p_1, ..., p_n)$ の全てのストリームの束ができあがったら，(1) 式より，それらの観測値の和を e_s のバンド幅として決定する．バンド幅が，まさに (1) 式の値と確定することができる条件とは，ストリームの束の合計値が頭打ちになった時点での $(p_0, ..., p_m)$ の，全てのペア間の通信経路中に現れるようなエッジの集合$\boldsymbol{L}$の元が e_s ただ一つだった場合である．まとめると**図 6** のようになる．

次に，アルゴリズムの効率化について考える．2 節で例に挙げた NWS は，測定の衝突が

```
E is a set of the all edges in network graph.
P_e is a set of (snd, rcv)-pairs such that the packet-flow between p ∈ P_e traverses via e ∈E.

for each edge e in E do
    M ← φ;
    for each pair p in {P_e - M} do
        Start flowing network stream between p;
        M ← M+{p};
        if sum of observed bandwidths is unchanged then
            L ← set of edges that appear in every path in M;
            M ← M+{all p that contain edges all that apear in L};
        end if
    end for
    if L == {e} then
        B_e := sum of observed bandwidths;
    else
        B_e := sum of observed bandwidths or more;
    end if
end for
```

図 6 バンド幅マップ構築アルゴリズム（2）
Fig. 6 An Algorithm to Build Bandwidth Maps (2)

起こるのを防ぐためにトークンを回してペア間の測定を逐一進めていると紹介した．逆に考えると，衝突さえしなければ，複数のペア間の測定は並列して行ってもよいということになる．このように測定の並列化によって，バンド幅マップ取得アルゴリズムの効率化を図る．

例えば，以下のような方法が考えられる．入力のネットワークトポロジーを平衡木として見，深さを d とする．最初に，深さ d の葉ノードに当たるエンドホスト群と，深さ $d-1$ の位置にあるスイッチとでできる部分トポロジー群に着目する．それらの部分トポロジーはそれぞれネットワーク的に閉じた系であるので，互いに測定の際のネットワークの衝突は起こり得ない．故にそれらの部分トポロジーは並列して図 6 の入力として与えて測定を開始しても，干渉は起こさずにそれぞれ答えを得ることができる．ただし，各部分トポロジーに属するエンドホストの数は，元のトポロジーより少ないので，Iperf を束ねる際に導引できる本数が少ないことに注意する．すなわち，各部分トポロジー内の全エンドホストから Iperf を束ねても埋め尽くすことができないエッジが現れる可能性が高くなってしまう（図 6 の B_e := sum of observed bandwidths or more;）．その場合，その部分トポロジーを平衡木のルートに向かって拡張する．すなわち，深さ $d-2$ より深い部分で部分トポロジーを考

Vol.2009-HPC-121 No.29
2009/8/6

え，導引できるエンドホスト数を増やす．拡張された各部分トポロジーを再び並列に図 6 の入力として測定し，拡張前のトポロジーでは未確定であったエッジのバンド幅を突き止めていく．こうしてグラフの外周から中心に向かって徐々にエッジのバンド幅が決定していき，未確定のエッジが無くなるか，元のトポロジーが図 6 の入力となるまで繰り返す．この手順によって測定の並列化がなされ，全体の測定時間を短縮することができる．またここで挙げた，グラフの外周からエッジのバンド幅を決定していくという手順は，着目したエッジのバンド幅を決定させるという側面から見ても合理的である．理由は，測定エンドホストペア間に複数のエッジが存在するとそれらの最小のバンド幅の値しか得られず，しかもその値をどのエッジに割り振ってよいか分からないという Iperf 測定の欠点を思い出せば分かる．Iperf を実行するエンドホストペアと経由するエッジの集合がグラフの外周にあればあるほど，Iperf 測定経路中に介在する他のスイッチやエッジの個数が確率的に少なくなるため，着目しているエッジのバンド幅が決定されやすい．

4. おわりに

本稿では大規模なネットワークにおけるバンド幅測定手法の提案をした．本研究の貢献として並列分散システムにおけるデータ転送がボトルネックとなる分散アプリケーションの最適化を考え，それには有向ネットワークグラフにバンド幅の重みを付加したバンド幅マップが必要であると述べた．既存のバンド幅測定手法ではバンド幅マップの構築をすることは不可能であり，既存手法で得られるバンド幅の値というものが我々の目指すバンド幅マップ上のバンド幅とは意味の異なるものであるので，アルゴリズムの説明に先立ってエッジの向き付きバンド幅とはどのようなことでどのように測定すべきかを定義した．そしてナイーブなバンド幅マップの構築アルゴリズムを示した後，それを低負荷にする方法や効率を上げる方法を提示した．

今後の方針としては，定義を忠実に実行したナイーブなバンド幅マップ構築アルゴリズムを出発点として，いかにそれを低負荷なアルゴリズムにするか，また並列測定などを盛り込んでいかに効率の良いアルゴリズムにするかを様々なアプローチから追求していくことを考えている．さらにこのアルゴリズムを実装し，実際の大規模ネットワーク環境上で実験をし，精度や計算時間の評価をしていく．

更にその後の発展として，自動取得したバンド幅マップを応用して並列分散アプリケーションの最適化に役立てるなどの段階にまで目を向けていきたい．

参 考 文 献

1) Mathijs den Burger, Thilo Kielmann, and Henri E. Bal. Topomon: A monitoring tool for grid network topology. *In ICCS '02: Proceedings of the International Conference on Computational Science-Part 2*, pp. 558–567, 2002.
2) Sho Naganuma, Kei Takahashi, Hideo Saito, Tkeshi Shibata, Kenjiro Taura, and Takashi Chikayama. Improving efficiency of network bandwidth estimation using topology information. *Symposium on Advanced Computing Systems and Infrastructures (SACSIS)*, pp. 359–366, June 2008.
3) Tatsuya Shirai, Hideo Saito, and Kenjiro Taura. A fast topology inference — a building block for network-aware parallel computing. *In Proceedings of the 16th IEEE International Symposium HPDC 2007*, pp. 11–21, June 2007.
4) Kei Takahashi, Hideo Saito, Takeshi Shibata, and Kenjiro Taura. A stable broadcast algorithm. *to appear the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid2008)*, March 2008.
5) Ajay Tirumala, Feng Qin, Jon Dugan, and Jim Ferguson. Iperf. http://www.dast.nlanr.net/projects/Iperf/.
6) Rich Wolski. Dynamically forecasting network performance using the network weather service. *In Proceedings of the 6th High-Performance Distributed Computing Conference (Aug. 1997)*, pp. 316–325, August 1997. http://nws.cs.ucsb.edu/ewiki/.
7) Rich Wolski, Neil T. Spring, and Jim Hayes. Implementing a performance forecasting system for metacomputing: the network weather service. *Proceedings of the 1997 ACM/IEEE conference on Supercomputing*, pp. 1–19, 1997.
8) Shota Yoshitomi, Ken Hironaka, and Kenjiro Taura. An adaptive gather algorithm avoiding contention. *Symposium on Advanced Computing Systems and Infrastructures (SACSIS)*, pp. 71–78, May 2009.