卒業研究論文

大容量データ配信の効率化のための配信木の構築手法の検討とその評価

指導教員　松尾 啓志 教授
津邑 公暁 准教授

名古屋工業大学 工学部 情報工学科
平成18年度入学 18115016番

伊藤 翼

平成22年2月8日

目 次

1　はじめに

計算機ネットワーク上の不特定多数の受信者に大容量の同一データを同時に送信するマルチキャストは，必要不可欠なデータの送信手法の一種となっている．このマルチキャストは，動画像コンテンツの配信サービスや，ホワイトボードなどのマルチメディアを用いた対話的なグループ通信サービスなどで用いられている．インターネットの急速な発展や，汎用計算機の高性能化と低価格化により，このようなサービスのいっそうの普及が予想される．

このようなサービスに用いられるマルチキャストには技術的問題と経済的問題の 2 つの問題がある．前者は，マルチキャストに参加するノードが増大すると通信網の負荷が増大するという問題であり，後者は，サービスの提供者に大規模な設備投資の負担が必要になるという問題である．この 2 つの問題を考慮した低コストな配信手段が求められている．

そこで，低コストで効率的なマルチキャスト手法を考える必要がある．このマルチキャストの効率化を実現するための様々なアプローチがあるが，そのうちの 1 つに効率の良い配信木を構築するという研究がある．これらの研究では，あらかじめデータを配信するための木を生成し，その木を用いて配信路を決めることでマルチキャストの性能向上を図る．本研究では，効率の良いマルチキャスト木を構築することでマルチキャストの性能を向上させることを検討する．

性能向上のためには，このマルチキャスト木を用いることでマルチキャストに参加する全ノードができる限り大きなバンド幅でデータを受信することが重要である．そこで，各ノードのバンド幅 (帯域幅) の合計値を最大化するために，計算機ネットワークをグラフ問題に形式化し，マルチキャスト木の構築を組み合わせ最適化問題の枠組みで解く．この最適化によって構築された木をベースとして，各ノードに，複数に分割された配信データの塊を中継させることで，マルチキャストの効率化を図ることを考える．また，実際のネットワーク環境では，マルチキャストするデータ以外の通信によって，通信リンクの使用状況が変わり，バンド幅に変動が起こりうるため，構築したマルチキャスト木のままでは常に安定した性能が得られるとは限らない．そこで，バンド幅の変動に応じて，マルチキャスト木を動的に再構築することでマルチキャストの性能を維持することも検討する．

本論文では，2 章で本研究の背景，データ配信のモデルと既存研究について述べる．3 章では本研究で提案する手法について述べ，4 章ではその実装について述べる．5 章では，提案する手法の評価とそれに対する考察を述べる．最後に 6 章で本研究についてまとめる．

2　研究の背景

2.1　大容量データの同時通報 (マルチキャスト)

近年，インターネット環境における商業ベースでの動画配信サービスや一般ユーザによる投稿動画コンテンツの配信が盛んに行われている．そのようなデータの配信にでは不特定多数の相手に大容量のデータを同時通報する**マルチキャスト**が必要不可欠である．しかし，単純なマルチキャスト手法を用いた場合，通信網の負荷の増大や，配信サービスの提供者の経済的な負担の増大といった問題が生じる．例えば，最も基本的なマルチキャストの方法として，単一の送信元ノード (配信元) が全ての宛先ノード (配信先) に対して同じ比率で配信する場合，送信元ノードがボトルネックとなり性能が低下する．サービスの提供者は，この性能低下を防ぐための大規模な設備投資に大きくコストを割くことになる．そのため，宛先ノード数の増加に対処することが困難である．このような問題を解決するデータ転送手法として，宛先ノードにデータを中継させることで，マルチキャストを行うパイプライン転送がある．

2.2　データの配信モデル

本研究で扱うマルチキャストの前提として考えるべきデータの配信モデルについて説明する．データ配信モデルを以下の図 1 と図 2 に示す．

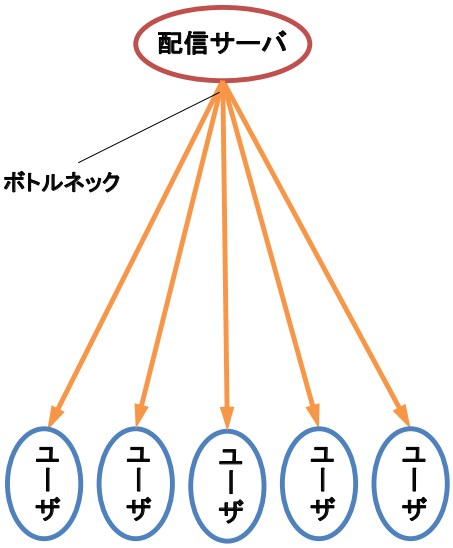

図 1: 集中型の配信モデル

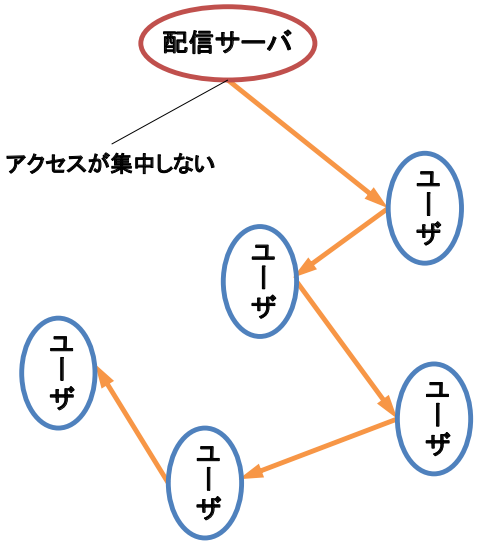

図 2: パイプライン型の配信モデル

図 1 は配信サービスの提供者の配信サーバ（送信元）が全てのユーザ（宛先）に対して同じ比率でデータを配信するようなモデルである．図 1 のような配信モデ

ルでは，配信サーバの負担が大きくなる．仮に，ユーザの数が n の場合，配信サーバの通信リンクを n 本の通信が共有することとなり，均等に分割した場合，**バンド幅 (帯域幅)** が 1/n となる．従って，このような配信形態をとった場合，送信元の通信リンクの占有負荷が増大するため，図 1 のように配信サーバの通信リンクがボトルネックとなる．また，そもそもデータの配信自体に経済的なコストがかかるため，そのコストが増大するという問題が生じる．さらに，通信網の負荷の増大に伴って，配信サーバや通信リンクの強化など設備投資の負担も増大する．これに対して図 2 の配信モデルは，配信サーバが全てのユーザに対して配信するのではなく，配信サーバが配信データを各ユーザに中継させることで配信を行うモデルである．このような送信手法を**パイプライン転送**と呼ぶ．このモデルでは，配信サーバが配信データを複数の部分に分割し，それを各ユーザに順々に中継させることで配信を行う．ユーザは自分の前に受信したユーザから，配信データの一部分を受信するとすぐに次のユーザにそれを転送する．このように配信データを受信しながら，次の相手に送信することで，前述したような通信リンクの共有による性能低下の改善と設備投資の負担の軽減が可能となる．[6][9][10] では，このパイプライン転送を用いて配信データの送信の効率化を図っている．本論文では，配信データを転送する手法として，パイプライン転送を用いることを前提とする．

2.3 マルチキャストの性能の評価方法

マルチキャストの性能を評価する指標について説明する．本研究では，配信木を構築することでマルチキャストの性能向上を図る．配信木とは，配信データをマルチキャストするための通信網であり，これを**マルチキャスト木**と呼ぶ．このマルチキャスト木の評価は，マルチキャストの対象となる各宛先ノードの**流入バンド幅**の合計値で評価する．流入バンド幅とは，パイプラインに参加している各ノードが自身の前のノードから配信データを受け取る際のバンド幅である．この評価指標は [9] で用いられている．

一般にデータ送信の完了時間は通信遅延とバンド幅によって決まる．動画など大容量コンテンツの配信や，大規模計算の大容量演算データの送信のためのマルチキャストの場合，データの送信の完了時間において，バンド幅の値のみが重要となる．以下の式 (1) にバンド幅，データのサイズとデータの送信の完了時間の関係を示す．

T : データの送信の完了時間
W : データのサイズ
D : 通信遅延
B : バンド幅

$$T = W/B + D \tag{1}$$

式 (1) はある 2 ノード間でのデータの送信の完了時間 T とバンド幅 B の関係を表している．仮に $W = 100$ MB , $B = 100$ Mbps , $D = 50$ ms の場合, T の値の 9 割以上がバンド幅に依存することになる．

また，ある 1 つの送信元ノードが n 個の宛先ノードにパイプライン転送をする場合を考える．この場合のデータの送信の完了時間は式 (2) のように表される．

T' : パイプライン転送の完了時間
W' : 分割されたデータのサイズ
D : 通信遅延
B_i: ノード i のバンド幅

$$T' = \sum_{i=1}^{n} (W'/B_i + D) \tag{2}$$

パイプライン化することによるオーバヘッドを無視すれば，T' は B_i に大きく依存することがわかる．従って，各宛先ノードの流入バンド幅を最大化すれば，性能向上が可能となる．本論文では，各宛先ノードの流入バンド幅の合計値を評価値として用いる．以降，これを**合計バンド幅**と呼ぶ．

2.4 パイプライン転送における合計バンド幅

図 3 を用いて，パイプライン転送における合計バンド幅を説明する．

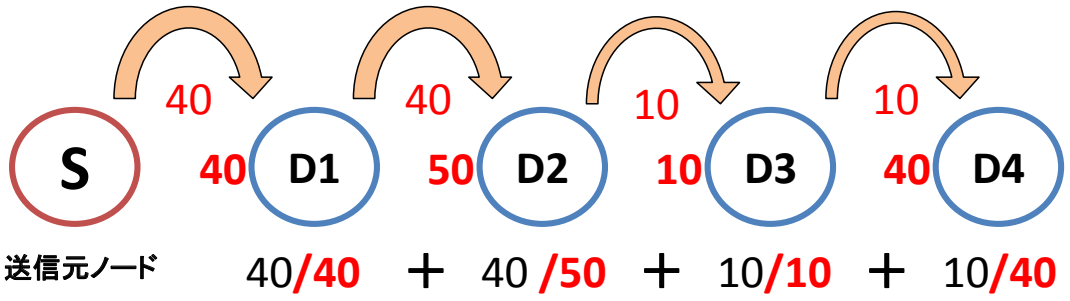

図 3: パイプライン転送における合計バンド幅

図 3 では，送信元ノード S が宛先ノード D1〜D4 にパイプライン転送で配信データを送信する．S は D1 に配信データを送信し，S からそれを受信した D1 は D2 に送信する．同様に D2 は D3 に，D3 は D4 に配信データを送信する．ノード D1〜D4 の左の数字は各ノードが受信することが可能な最大の流入バンド幅であり，矢印の近くの数字は各ノードの実際の流入バンド幅である．図 3 の場合の合計バンド幅は，$40 + 40 + 10 + 10 = 100$ となる．ノード D1 と D3 はそれぞれ本来持つ最

大の流入ンド幅 40 と 10 で配信データを受信することができる．しかし，D2，D4 は本来持つ最大の流入バンド幅で受信することができない．D2 は最大の流入バンド幅が 50 に対して流入バンド幅が 40，D4 は最大の流入バンド幅が 40 に対して流入バンド幅が 10 となる．このようにパイプライン転送に参加するノードの流入バンド幅は自分より前に転送を担当するノードの流入バンド幅に依存する．

2.5 パイプライン転送の問題点

パイプライン転送には，遅い（バンド幅の小さな）ノードがパイプラインに参加した場合に，全体の性能が低下してしまうという問題がある．具体的な例を図 4 に示す．図 4 のノード S は送信元ノード，ノード D1〜D3 は宛先ノードである．この例のノード D1 のようにバンド幅 10 を持つ遅いノードがパイプライン転送に参加した場合，本来はノード S からバンド幅 100 で受信可能なノードがバンド幅 10 で受信することになり，性能が低下する．

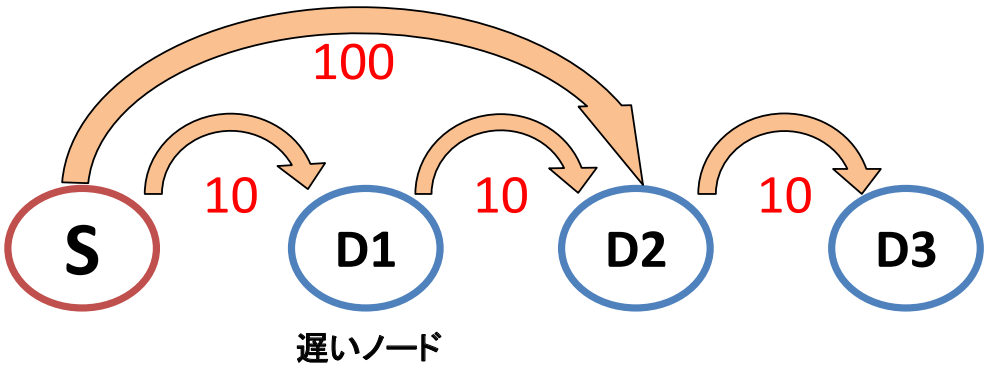

図 4: 遅いノードの参加

2.6 グラフ問題としての表現

2.6.1 グラフ理論

グラフ理論は，現実社会のあらゆる「ネットワーク」を定式化して問題を解くために用いられる．グラフ理論において，グラフ G は頂点の 集合 V と辺の 集合 E によって構成される．また各辺が重みを持つようなグラフを重み付きグラフと言う．重みつきグラフ G は $G = (V, E, c)$ と表される．c は $E \to R$ で表され，R は辺を 重み c に関連付ける重み関数となる．さらに，方向性を持たない辺を含むグラフを無向グラフと言う．

本研究では,「マルチキャスト木の構築」という目的達成のために，計算機ネットワーク上の問題をグラフ問題に適用させる．具体的には，以下のように計算機ネットワークをグラフに対応付ける．

- 計算機ネットワーク = 重み付き無向グラフ
- 計算機 (ノード) = 頂点
- 通信リンク = 辺
- バンド幅 = 重み

2.6.2 最短経路問題

最短経路問題とは重み付きグラフ $G = (V, E, c)$ が与えられた時，V の 2 頂点間の最短経路を求める問題である．最短経路とは 2 頂点間の経路の辺の重みの和が最小となるような経路である．最短経路問題の中でも，単一始点最短経路問題は，任意の $v \in V$ を始点として，頂点 v から他の全ての頂点への最短経路を求める問題である．この単一始点最短経路問題を効率よく解くことができる **Dijkstra のアルゴリズム**[2] がある．

Dijkstra のアルゴリズムでは，まず集合 $X = \{s\}$ を生成する (s は始点)．この X には最短経路が確定済みである頂点が追加される．頂点 s から各頂点 x までその時点での最短距離 $d[x]$ を更新しながら，最短経路となる x を 集合 X に順々追加し，X が全頂点を含むまで拡大させることで，s から全頂点への最短経路を求める．もし，ある時点である 頂点 u を X に追加した場合，まず $u \in X$ に隣接した頂点の中でも距離が最も小さい 頂点 v を最短経路の確定候補に選ぶ．そして，距離 $d[u] + c(u, v)$ が $r \notin X$ かつ $r \neq v$ である 頂点 r の $d[r]$ より小さければ，頂点 v の X への追加と $d[v]$ の更新を保留して 頂点 r を集合 X に追加する．もし距離 $d[u] + c(u, v)$ が $r \notin X$ の $d[r]$ よりも小さければ 頂点 v を X に追加し $d[v]$ を距離 $d[u] + c(u, v)$ で更新する．ただし，距離 $d[u] + c(u, v)$ が $d[v]$ よりも大きければ $d[v]$ は更新されない．このように，頂点 s を含む 集合 X から順に探索範囲を拡大していき，それぞれの時点での最短距離を更新していく．集合 X が全頂点を含むまで探索を実行することで，最短経路を求める．

2.6.3 深さ優先探索

深さ優先探索は，グラフ探索手法の 1 つである．深さ優先探索では，Dijkstra のアルゴリズムのように始点からの距離は関係なく，始点から探索できる頂点がある限り探索を行う．具体的には，探索先の頂点に 1 つでも隣接頂点が存在する場合はその頂点を探索し，探索先の頂点にそれ以上探索可能な頂点が存在しない場合は 1 つ前の頂点に戻り，また新たな頂点を探索する．最終的には，全ての頂点を探索し終えるまでこれを繰り返すことで，始点から全頂点への経路を求める．

2.7 既存手法

2.7.1 Dijkstra のアルゴリズムを用いる手法

2.6.2 で述べたように Dijkstra のアルゴリズムは，グラフ上のある頂点から他の全ての頂点の最短経路を効率的に求めることができる．つまり，最短経路木を生成することができる．最短経路問題の場合は，グラフの重みがコストで表わされるが，これをバンド幅と置き換えれば，ある送信元ノードから他の宛先ノードまでの経路上のバンド幅の合計値を最大化することが可能である．従って，バンド幅の合計値を最大化するマルチキャスト木を構築することができる．この Dijkstra のアルゴリズムを用いたマルチキャスト木の構築手順を以下に示す．

1. 送信元ノードから Dijkstra のアルゴリズムを用いてバンド幅の合計値が最大となる宛先ノードを選ぶ
2. 1. で選んだノードを送信元として Dijkstra のアルゴリズムを用いて新たに宛先ノードを選び，経路をつくる
3. 全ての宛先ノードが経路に含まれたら木の構築を終了する

このように，常に 2 つのノード間でバンド幅の合計値が大きくなるように，ノードを順々に経路に追加すれば，遅いノードをパイプライン転送の早い段階に参加させることはなくなり，性能低下を防ぐことができる．

ただし，この手法には，通信リンクの共有が多く存在してしまう可能性があるという問題がある．Dijkstra のアルゴリズムを用いたものでは，バンド幅の大きいノードを優先的に配信路に加えていくが，この場合，配信路において通信リンクの重複が起こる可能性がある．

2.7.2 Multi Stream Pipeline Broadcast

Multi Stream Pipeline Broadcast[9] は **FPFR(Fast Parallel File Replication)**[6] を改良したものである．FPFR は深さ優先探索を繰り返して，複数のマルチキャスト木を構築するものであるが，[9] では FPFR よりも多く深さ優先探索を繰り返すことで，FPFR よりも多くのマルチキャスト木を構築する．木の構築の概要は次の通りである．

- 全ての宛先ノードを含む木と，その部分的な木を構築する
- 構築した複数のマルチキャスト木を並行に用いて配信データを送信する
- 配信データの送信は，構築した木の数分のステージに分かれ，各ステージでそれぞれ異なるマルチキャスト木を用いる

複数のマルチキャスト木は深さ優先探索を繰り返し用いることで構築される．この複数のマルチキャスト木を並行に用いて，それぞれの木で異なった宛先ノードに配信データを送信することで，各宛先ノードの受け取るバンド幅を最大化することが可能である．深さ優先探索を用いたマルチキャスト木の構築手順を以下に示す．

1. 送信元ノードから深さ優先探索を用いて，木を構築し，最小のバンド幅を木のスループットする
2. 各辺のバンド幅から前の木の最小バンド幅を差し引いて，バンド幅が 0 になった辺は消す
3. 新たに深さ優先探索で木を構築する
4. 2.，3. を繰り返して辿れる宛先ノードがなくなった場合，木の構築を終了する

このようにそれぞれの木の構築段階における最小のバンド幅を差し引いて，新たに木を構築することで，早いノード (大きいバンド幅を持つノード) が，遅いノード (小さいバンド幅を持つノード) に邪魔されることなく自身の可能な最大のバンド幅でデータを受信することができる．なお，配信データの送信は，配信データを複数の部分に分割して行われる．この複数のデータの大きさは，それぞれの木のスループットに適したものとなる．

ただし，Multi Stream Pipeline Broadcast では，計算機ネットワーク全体を管理するようなノードが，最小のバンド幅を差し引いた全体のスループットを把握していなければならない．

2.7.3 問題点

従来研究で提案された木の構築では，管理ノードが必要である．管理ノードはネットワーク全体の情報，すなわちネットワークトポロジや通信リンクのバンド幅といった情報を集約し，管理する必要があるため，その分のコストがかかる．このように全体の情報を集約するような管理役が存在するマルチキャストの通信網の形態を集中型と呼ぶ．また，実際のネットワーク環境では，他のトラフィックによって通信リンクのバンド幅が変動する可能性が十分ある．その場合，一度構築したマルチキャスト木のままでは性能が低下する可能性がある．集中型のように，バンド幅が変動する度に逐一管理ノードに情報を集約し，管理ノードがネットワーク全体に情報を伝達していたのでは効率が悪い．従って，各ノードが自身の周辺の情報のみを管理し，自律的に処理を行うような分散型でのマルチキャスト木構築が望ましい．

3　提案手法

本研究では，各ノードの通信量を大きくできるような木をマルチキャスト木として用い，その木をベースとしてパイプライン転送を行うことを提案する．さらに，各通信リンクのバンド幅の変動に応じた通信リンクの動的な切り替え手法についても提案する．これらは，2.7.3で述べたような問題解決のために分散型で実現する．

また，分散型でのマルチキャスト木の構築のために計算機ネットワークの問題を2.6で説明したようなグラフ問題に置き換える．また，マルチキャストのための木の構築手法は，既存研究で提案された**GHSアルゴリズム (Gallager,Humblet and Spira 's Algorithm)**[3]を用いる．

3.1　組み合わせ最適化問題

組み合わせ最適化問題とは，ある問題においてある条件の元で選べる組み合わせの中で，最良の組み合わせを選ぶ問題である．マルチキャスト木の構築問題の場合は，計算機ネットワーク中の辺の候補の中から，合計バンド幅を最良にできるような辺の組み合わせを選ぶ．

3.1.1　最小全域木問題

最小全域木問題とは組み合わせ最適化問題の中の一問題である．全域木とは，与えられたグラフ中の頂点全てを含むような木である．その全域木の中でも，各辺の重みの和が最小となるものが最小全域木である．例えば，図5で表されるようなグラフがある場合，図6のような**最小全域木 (MST: Minimum Spanning Tree)**ができる．

一般に辺の重みを通信コストで表した場合は，総通信量を最小にできる木がMSTである．従って，通信の最適化を考える場合，このMSTはよく用いられる．本研究では，木の重みはバンド幅で表されるので，MSTはそれを最大化する木として考えることになる．

3.2　逐次処理によるMSTの構築

逐次処理によるMSTの構築アルゴリズムの代表的なものに，**Primのアルゴリズム**[8]と**Kruscalのアルゴリズム**[7]という2つのアルゴリズムがある．

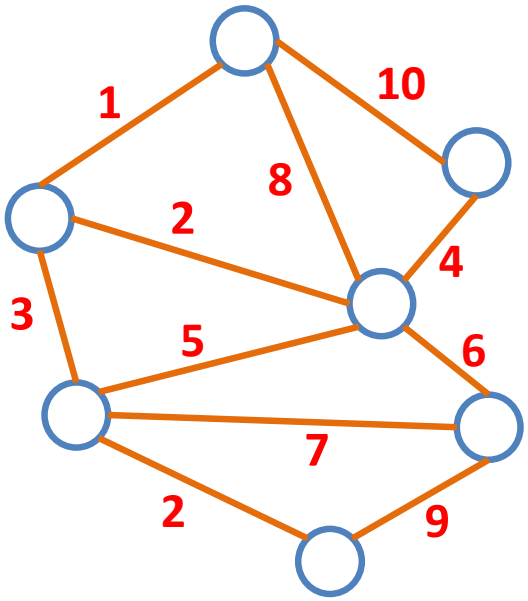

図 5: 重みつき無向グラフ

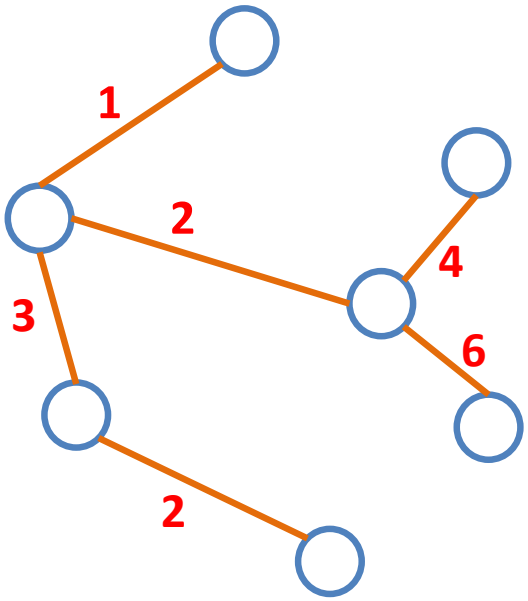

図 6: 最小全域木

3.2.1 Prim のアルゴリズム

Prim のアルゴリズムを用いた MST の構築手順は以下の通りである．$G = (V, E, c)$ が与えられた場合，$T = (V', E', c)$ を MST を構成するグラフとする．

1. $T = \phi$ とする
2. G から任意の頂点を 1 つだけ選び，V' に加える
3. $v \in V'$ と $u \in V - V'$ に含まれる 頂点 u とで構成される 辺 $\{u, v\}$ の中で，重みが最小となる u を選び，V' に加える
4. 2. を繰り返し，$V = \phi$ になれば，アルゴリズム終了．T は最小全域木となっている

3.2.2 Kruscal のアルゴリズム

Kruscal のアルゴリズムを用いた MST の構築手順は以下の通りである．T の任意の部分グラフを $F = (V, E_F)$ とする $(E_F \subseteq E')$．F を構成する T の部分木の中で，頂点 x が属するものを F_x と表す．

1. $F = (V, \phi)$ とする
2. E から重み $c(e)$ が最小となる 辺 e を取り出し，e を構成する 頂点 $u, v \in V$ の間に $u \notin F_v$ かつ $v \notin F_u$ という関係が成り立つ場合，e を E' に加える．

3. F_vと F_u は F_v もしくは F_u のどちらかに統合される

4. 2.，3. を繰り返して，T が連結となれば，アルゴリズム終了．T は最小全域木となっている

どちらのアルゴリズムも T が最小全域木であるという正当性は保証されており，与えられたグラフ G に対して必ず最適解が求まる．そして，$e \in E$ の中で $c(e)$ に重複がなければ，MST は一意に定まる．

この 2 つのアルゴリズムの計算時間量はどちらも $O(|E|\log(|V|))$ となるものがあることが知られている．従って，ノード数が増加してもある程度の拡張性を保って，MST の構築が可能であると考える．ただし，Prim のアルゴリズムや Kruscal のアルゴリズムでは，グラフ全体の大域的な情報，すなわち計算機ネットワーク全体の情報が必要である．従って，Prim のアルゴリズムや Kruscal のアルゴリズムを 2.7.3 で述べたような分散型にそのまま適用させることはできない．そこで，各ノードが局所的な情報から MST を構築できるような分散アルゴリズムの適用が必要となる．

3.3 分散処理による MST 構築

本研究では，分散アルゴリズムである GHS アルゴリズム [3] を用いる．GHS アルゴリズムは，分散環境下で，MST を構築するための分散アルゴリズムである．このアルゴリズムでは，3.2.2 で述べた [7] のアイデアを用いている．本研究の目的は「マルチキャストを効率化するマルチキャスト木の構築」であり，木の構築自体に多くの時間が割かれると，本質を見失ってしまう可能性がある．そこで，メッセージの交換量が最小のアルゴリズムとして知られている GHS アルゴリズムを用いる．本研究では，このアルゴリズムを用いて分散環境下での MST の構築を行う．

3.3.1 GHS アルゴリズムの概要

GHS アルゴリズムの概要を説明する．図 7 は，GHS アルゴリズムの概念図である．3.2.2 で述べたように $G = (V, E, c)$ が与えられた時に，$F = (V, \phi)$ から出発し，F を構成する T の部分木を拡張していく．GHS アルゴリズムでは，この部分木は**フラグメント**と呼ばれる．それぞれのノードが所属するフラグメントに属さないノードと繋がっている辺の中でも重みが最小のものを並列に付け加えていく点が [7] との違いとなる．このようにフラグメントの外に向かう辺を**外向辺**と呼び，この外向辺の中でも重みが最小のものを**最小外向辺 (MOE: Minimum-weight Outgoing Edge)** と呼ぶ．また，フラグメントには**コア**というリーダーとなる辺が存在する．ただし，実際にはそのコアを構成する 2 つのノードがコアを管理することになる．MST 構築の流れを次に示す．

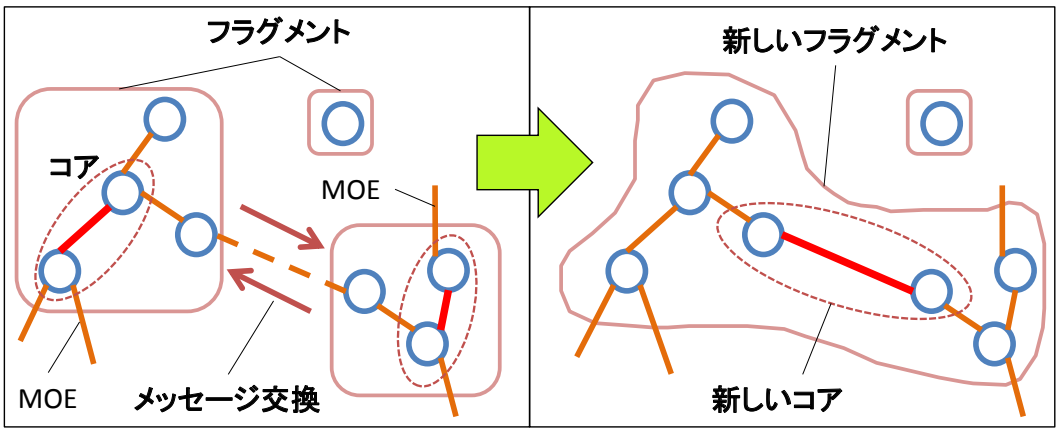

図 7: GHS アルゴリズムの概要図

- 各ノードは自身だけを含むフラグメントを生成する
- 各ノードは隣接している MOE を探索する
- 各ノードは発見した MOE を所属フラグメントのコアに報告する
- コアは報告された MOE の中からフラグメント内で最小の重みを持つ MOE を選択する
- この MOE と繋がっているの 2 つのフラグメントがお互いに同意したとき，その MOE を MST の辺として採択する
- MOE が同一フラグメント内の辺である場合はその MOE を拒否する

このように各フラグメントが MOE を追加することで，並列に自身のフラグメントを拡大していき，最終的にフラグメントが 1 つに統合された時に MST が構築されている．

3.3.2 ノードと辺の状態

GHS アルゴリズムでは，各ノードと各辺が状態を持つ．この状態について説明する．各ノードは以下の 3 つの状態を持つ．

Sleeping: 初期状態

Find: MOE を探索中の状態

Found: MOE を探索済みの状態

各辺は以下の 3 つの状態を持つ.

Branch: MST の辺である状態

Rejected: MST の辺でない状態

Basic: *Branch* と *Rejected* のどちらにも属さない状態

各ノードはコアからの要求によって自身の状態を Found に変化させ，MOE を探索する．また，各ノードは各辺の状態を変化させていき，最終的には辺の状態は Branch と Rejected のどちらかの状態となる．この状態が Branch である辺を参照すれば，MST に属する辺であることがわかる.

3.3.3 レベルとフラグメント ID

GHS アルゴリズムでは，各フラグメントはレベルと，ID を持つ．ID はそのフラグメントの識別子として用いられ，レベルはメッセージの交換数を減らすための工夫である．メッセージの交換の際の基本的な方針として,「自分よりレベルが低いフラグメントからのメッセージを受信した場合，返信を保留する」というものがある．このようにすることで，各フラグメントがレベルの値にばらつきのないように成長し，メッセージ数を減らすことが可能である.

3.3.4 メッセージ交換

GHS アルゴリズムでは，各フラグメント内の各ノードがメッセージを交換することで，フラグメントの拡張が行われる.

各ノードは以下のメッセージを交換し合う.

$Connect < L >$: 他のフラグメントへの MOE の接続要求メッセージ

$Initiate < L, W, S >$: フラグメント内の初期化メッセージ．そのフラグメント内の各ノードに伝搬し，レベル L，フラグメント ID となる W と，状態 S で初期化する．状態 $S = Find$ であれば，MOE の探索要求メッセージにもなる

$Test < L, F >$: MOE の調査メッセージ．隣接ノードに L, F を伝え，その辺を MOE の候補としてよいか調査する

Accept: $Test$ メッセージを受け取った場合，MOE の候補とすることを許可するメッセージ

Reject: $Test$ メッセージを受け取った場合，MOE の候補とすることを拒否するメッセージ

Report $< W >$: 発見した MOE の重み W をフラグメントのコアに報告するメッセージ

これらのメッセージを受信した各ノードは，そのメッセージに対する局所的な処理を行い，他のノードにメッセージを送信する．

3.3.5 MOE の探索

各フラグメントにおける MOE の探索の概略を図 8 に示す．図 8 では，フラグメント A のコアが $Initiate < L_A, A, Find >$ メッセージを伝搬させ，A 内の各ノードを ID=A とレベル=L_A で初期化するとともに，MOE の探索を要求する．$Initiate$ メッセージを受信したノードは MOE の候補となる辺に繋がるノードに $Test$ メッセージを送信する．$Test$ メッセージが送信されるのは，状態が $Basic$ であり，かつその中でも重みが最小の辺と繋がるノードである．図 8 では，辺$\{a, y\}$ の重みは 辺$\{a, x\}$ のそれより大きいため，ノード a は 辺$\{a, x\}$ に向けて $Test < L, A >$ を送信する．$Test < L, F >$ メッセージを受信したノード x は自身のフラグメント ID と A を比較し，それが同じであれば $Reject$ メッセージを，違っていれば $Accept$ メッセージを ノード a に返信する．ノード z は同一フラグメント内のノードであるため，辺$\{a, z\}$ の状態は $Rejected$ となる．$Accept$ メッセージを受信したノードは $Report < W >$ メッセージを伝搬させ，発見した MOE の重みをコアに報告する．各ノードから $Report$ メッセージを受信したコアは，報告された MOE の中でも最小の MOE を選択し，その MOE への接続要求メッセージ $Connect < L >$ を送信する．

3.3.6 フラグメントの結合

フラグメントが結合するのは，図 9 と図 10 のような場合である．$Connect$ メッセージを送信したフラグメント A の ノード a が，フラグメント B の ノード b から $Connect$ メッセージもしくは $Initiate$ メッセージを受信したとき A と B は結合する．この時，結合は以下のようにレベルによって場合分けされる．

1. $(L_A = L_B)$ A と B は合併する
2. $(L_A < L_B)$ B は A を吸収する
3. $(L_A > L_B)$ B は $Connect$ メッセージの返信を保留

1. の場合は図 9 に，2. の場合は図 10 にそれぞれ相当する．1. の場合，ノード b はノード a に対して $Connect < B >$ を送信しているため，お互い同意し，$Initiate < A + 1, W, Find >$ を送信し合い，A と B は合併する．2. の場合，ノード b はノード a に対して $Initiate < L_B, B, S >$ メッセージを送信し，B は A を吸収する．この時，$S = Find$ の場合は元の A のノード a は B の MOE の探索を要求するが，$S = Found$ の場合は探索を要求しない．

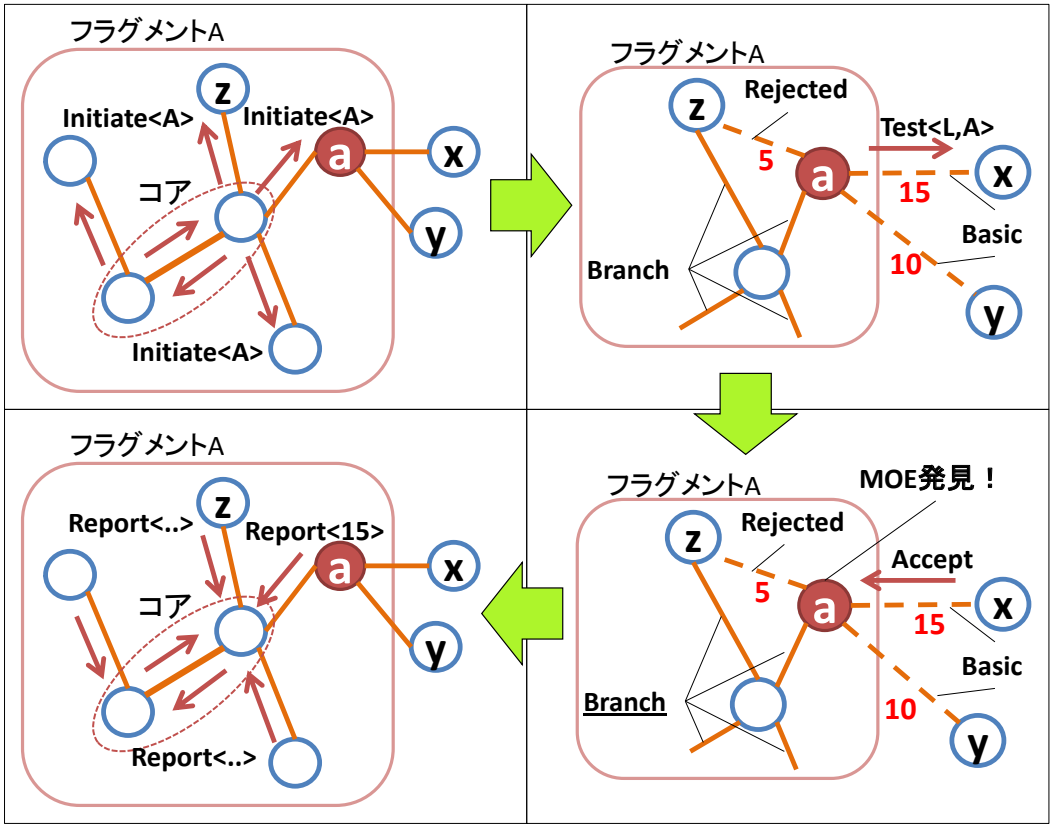

図 8: MOE の探索

3.4 MST 上のパイプライン

GHS アルゴリズムによって構築した MST を用いてパイプライン転送を行う．3.1.1 で述べたように MST は辺の重みの和を最大化できる木である．従って，MST から得られる各通信リンクのバンド幅の単純な和は最大化されるはずであり，その上でパイプライン転送を行うパイプラインをつくれば，マルチキャストの性能を向上できると考える．ただし，実際のパイプライン上のノードの合計バンド幅と MST の辺の和との間には多少ギャップがあるはずだが，少なくともバンド幅を考慮しない単純なマルチキャスト木よりも，性能は高くなるはずである．

また，パイプライン転送を行うためには，データの送信自体に何らかの順序が必要である．よって，1 つあるいは，複数の中継を担うノードと，データの中継を行う必要のないノードとで当然，処理が異なる．つまり，各ノードの担うデータ送信を変える必要がある．

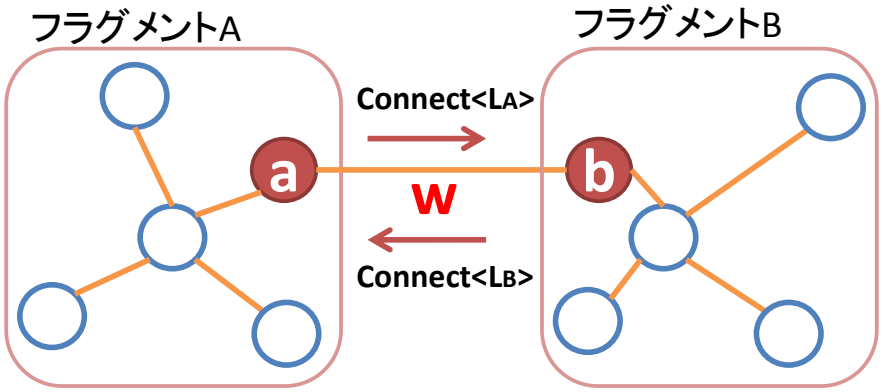

図 9: フラグメントの合併

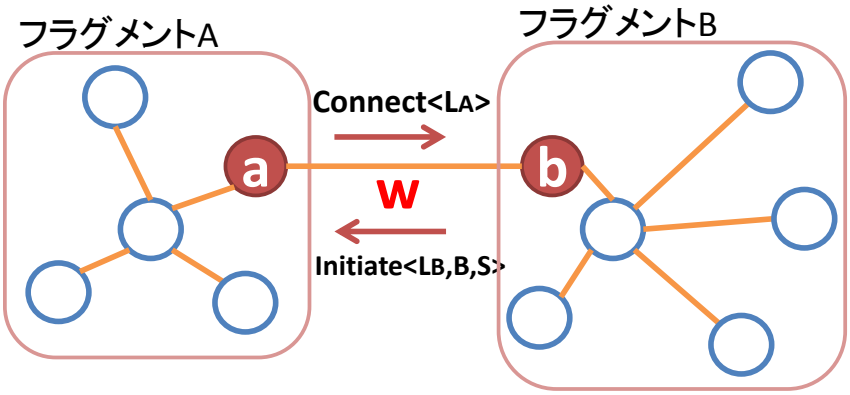

図 10: フラグメントの吸収

3.5 動的な辺の切り替え

通信リンクのバンド幅に変動があった場合の辺の切り替えについて説明する．

実際の計算機ネットワークでは，他のトラフィックなどの影響により，通信リンクのバンド幅が変動しうる．その場合，既に構築した MST のままでは性能が低下する可能性がある．この性能低下を防ぐためにバンド幅の小さくなった通信リンクを他の通信リンクで代用することが考えられる．本研究では，一度 GHS アルゴリズムを用いて MST を構築した後に，MST に含まれない辺を用いて木を再構築することで，辺の切り替えを実現する．木の再構築に関しては，**Self-Stabilizing Algorithm(自己安定アルゴリズム)** という研究分野で多く研究されている．その中で MST の再構築の方法についても多く研究されている [1][4][5].

これらの研究で用いられる辺の切り替えの概要について説明する．まず初期木としてグラフ $G = (V, E, c)$ から MST である $T = (V, E', c)$ を構築する．

- 頂点 $u, v \in V$ が $c(\{u, v\})$ の変動を検知する

- $u, v \in V$ は，辺$\{u, v\}$ を追加した場合にできる閉路づたいにメッセージを巡回させ，$c(\{u, v\})$ を伝える
- メッセージを受信したある頂点は，自身の管理する辺の重みをメッセージに載せ，次の隣接頂点に転送する
- 巡回したメッセージを受信した u, v は各頂点の重みと $c(\{u, v\})$ を比較する
- $c(\{u, v\})$ より大きいものがある場合，その中でも最大の重みを持つ辺を E' から消すためのメッセージを巡回させる

このような木の再構築手法を用いた場合，全体のボトルネックとなる辺の検出とその辺の削除のために，少なくともそれぞれ 1 回ずつ閉路づたいにメッセージを巡回させる必要がある．この閉路が大きいものとなれば，全体として木を再構築するまでに，その分多くのコストがかかってしまう．そこで，辺の切り替えを行ってよいノードを限定して，木を構築し直すことを考える．

この切り替えを行ってよいノードを a とする．あるタイミングでバンド幅に変動があり，ノード a がその時点で使用している辺よりも良い辺 (バンド幅の大きい辺) を発見した場合，自身の親ノードとなる ノード b に MST の 辺$\{a, b\}$ を使用しない旨を伝え，良い辺に切り替える．もし，辺$\{a, b\}$ よりも良い辺がない場合は，MST の辺をそのまま使用する．ある ノード a が自身の親ノードとなるノードとの間の辺を一本消して，別の辺をどこかに一本付け加えたとしても，木であることが保たれるはずである．

4 実装

4.1 ノードの役割とコネクションリスト

3.3.2 で述べたように GHS アルゴリズムでは，各ノードは各辺の状態を管理している．MST の構築が終了した時点で，各辺の状態は $Branch$ か，$Rejected$ のどちらかに定まっているはずである．$Branch$ とは MST の辺である状態であり，$Rejected$ とは MST の辺でない状態のことである．各隣接辺の情報は各ノードがコネクションリストというリストに記憶することにする．このコネクションリストには，隣接辺を挟むノードの ID，その隣接辺のバンド幅と辺の状態の 3 つのエントリがある．リスト内の辺の情報はバンド幅の値が大きい順にソートされている．従って，MST の構築が終了した時点で，隣接辺の中でバンド幅が最大であり，かつ状態が $Branch$ であるものがリストの先頭のエントリに存在するはずである．

また，MST の中で，各ノードは葉ノードと節ノードに分類される．葉ノードは，親ノード 1 つのみとの間に MST の辺をもつノードであり，節ノードはそれ以外のノードである．

4.2 パイプラインの順序付け

パイプライン転送における配信データの中継のための順序付けについて説明する．送信元ノードを親としたMST上に配信データの送信の流れを持たせる必要があるが，基本的には送信元ノードから順に各ノードの隣接辺の状態が$Branch$である辺づたいに配信データを送信していけばよい．

葉ノードの場合は，コネクションリスト中の隣接辺の状態が$Branch$であるものが1つの場合，自分が葉ノードであると判断し，親ノードからの配信データを受け取るだけでよい．節ノードの場合はコネクションリスト中の隣接辺の状態が$Branch$であるものが2つ以上の場合，自分が節ノードであると判断する．節ノードは，データを中継する必要があるため，親ノードから配信データを受信するとすぐに次のノードにそのデータを送信する．次のノードとは，辺の状態が$Branch$であり，かつ流入バンド幅がその時点で最大となる葉ノードである．配信データの一部分を全ての葉ノードに対して送信し終えたら，別の節ノードに配信データを送信する．このように順序付けを行うことで，MSTをベースとしたパイプライン転送は可能であると考える．

4.3 辺の切り替えアルゴリズム

4.3.1 メッセージ交換

実装した辺の切り替えアルゴリズムは，各ノード間でのメッセージ交換によって実現される．各ノードは以下のメッセージを交換する．

$Remove$: 辺を使用しない旨を親ノードに伝えるメッセージ

$Allow$: 葉ノードに辺の切り替えの許可を出すメッセージ

$ChangeConnect < PRI >$: 辺の切り替えを要求するメッセージ

$Commit$: その辺への切り替えを許可するメッセージ

$Abort$: その辺への切り替えを拒否するメッセージ

$Insert$: 切り替えた元の辺を再び使用する旨を親ノードに伝えるメッセージ

PRIは各ノードの持つ辺の切り替えの優先度である．このPRIは，MSTの構築が終了した時点で決まる．この優先度は，葉ノードと，その葉ノードの親ノードとの間の辺のバンド幅と，それらのノードのIDによって定まる．バンド幅が小さい方が優先度が高く，逆にバンド幅が大きい方が優先度が低い．

4.3.2 切り替えの有無

各ノードはあるタイミングでコネクションリストを更新し，それを参照した内容に応じて局所処理を行うことで，問題の変化 (バンド幅の変動) に対応する．コネクションリストの更新が起こった時，各ノードは以下の場合分けによって辺の切り替え処理の有無を判断する．

1. リストの先頭の辺の状態が $Branch$ の場合
2. リストの先頭の辺の状態が $Rejected$ の場合

葉ノードは，1. の場合は辺の切り替えの処理を行い，2. の場合は辺の切り替えの処理を行わない．

4.3.3 切り替えの流れ

4.3.2 の 1. の場合ように葉ノードが切り替えたい辺を発見した時の切り替えの流れを説明する．例えば，図 11 のようなバンド幅の変動が起こった場合，葉ノード a は 辺$\{a, c\}$ に辺を切り替える．

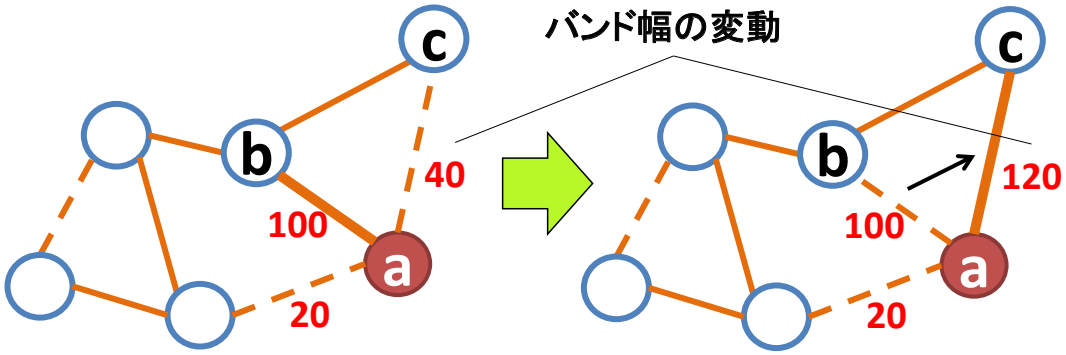

図 11: 辺の切り替え

葉ノード a は自身の親ノードとなる節ノード b に対して，$Remove$ メッセージを送信し，自身のコネクションリストの 辺$\{a, b\}$ の状態を $Rejected$ に書き換える．$Remove$ メッセージを受信した ノード b はその時点で自分が葉ノードになっていないことを確認した上で，ノード a に $Allow$ メッセージを送信する．ノード b から $Allow$ メッセージを受信した ノード a は，コネクションリストの先頭のエントリにある ノード c に対して，$ChangeConnect(PRI)$ を送信することで，辺$\{a, c\}$ の使用要求を出す．ノード a から $ChangeConnect(PRI)$ を受信した ノード c は自分が切り替えたい辺と ノード a に要求された 辺$\{a, c\}$ が異なっていれば，ノード a に

$Commit$ メッセージを送信することで，ノード a の 辺$\{a,c\}$ への切り替えを許可する．この時，ノード cは自身のコネクションリストの 辺$\{a,c\}$ の状態を $Branch$ に書き換える．もし，ノード cが切り替えたい辺と 辺$\{a,c\}$ が同一であれば，優先度 PRI の値を比較する．$PRI_a < PRI_c$ の場合は，ノード cは ノード aに対して $Abort$ メッセージを送信することで，ノード a の 辺$\{a,c\}$ への切り替えを拒否する．この時，自身のコネクションリストの 辺$\{a,c\}$ の状態を $Branch$に書き換える．$PRI_a > PRI_b$ の場合は，ノード cは ノード aに対してメッセージを返信せずに，自身のコネクションリストの ノード a の状態を $Branch$に書き換える．$Abort$ メッセージを受信したノードは，自身と $Abort$ を送信してきたノードとの間の辺と，元の辺 (以前の親ノードとの間の辺) 以外の辺の中で，状態が $Rejected$ かつ，バンド幅が最大であるノードに $ChangeConnect < PRI >$ メッセージを送信する．もし，そのようなノードがなければ，以前の親ノードに $Insert$ メッセージを送信することで，元の辺を再追加する要求を出す．この時，自身のコネクションリストの元の辺の状態を $Branch$ に書き戻す．$Insert$ メッセージを受信した親ノードは，自身のコネクションリストの $Insert$ を送信してきたノードとの間の辺の状態を $Branch$ に書き戻す．

最終的には，各ノードのコネクションリストの同期がとれるので，それを参照すれば，使うべき辺がわかる．このような辺の切り替えを行えば，葉ノードが自身の親ノードと，辺を切り替えたい先のノードとの間でのみメッセージを交換すればよいので，メッセージの交換量は全体として少ないはずである．また，各葉ノードが自分の知りえる範囲で大きいバンド幅の辺を使おうとするため，その木の性能はある程度は保てるはずである．ただし，このようにして構築された木は MST ではないことは明らかであり，バンド幅は最大化されない．

5 評価

5.1 実験 1: 合計バンド幅による評価

実験 1-1，実験 1-2 ともに送信元ノードから宛先ノードへデータをマルチキャストするためのマルチキャスト木を構築することを目的とする．構築したマルチキャスト木は，合計バンド幅を用いて評価する．

5.1.1 問題設定

実験 1-1 と実験 1-2 における問題設定は以下の通りである．

- 通信リンクは全二重
- 通信リンクは対称的

- 送信元ノード数は 1
- 宛先ノードの数は総ノード数 n に対して n-1
- 総ノード数は 10，50，100 の 3 通り
- バンド幅の種類
  - **(a)** 50 から 60 の間に一様分布
  - **(b)** 50 から 100 の間に一様分布
  - **(c)** 10，100 が混在

今日のネットワーク環境では，ほとんどの通信リンクで全二重通信が可能である．データを受信しながら送信するというパイプライン転送を行うための問題設定として，通信リンクは全二重とする．また，通信リンクは対称的であると仮定するので，ノードの流出バンド幅 (各ノードが送信する時のバンド幅) も流入バンド幅も同じ値を持つ．対称的であれば，2 方向の通信リンクのバンド幅を 1 本のリンクとして扱うことができるため，問題を簡単化できる．また，各リンクに対するバンド幅 (重み) は (a)5〜50 に一様分布，(b)40〜50 に一様分布と (c)10,100 が混在の 3 つの場合であり，バンド幅の分散値は (a)<(b)<(c) の順に大きい．なお，この実験はあくまで各マルチキャスト木の合計バンド幅を見積もる実験であり，実際のデータ送信は行っていない．評価対象は DepthFirst，Random，Dijkstra，FPFR+α，MST の 5 つである．DephthFirst は深さ優先探索を一度だけ用いたもの，Random はランダムに木を構築したもの，Dijkstra は Dijkstra のアルゴリズムを用いたもの，FPFR+α は 2.7.2 の手法を用いたもの，MST は MST を用いたものである．図 12 と図 13 のグラフの縦棒は左から順にそれぞれ DepthFirst，Random，Dijkstra，FPFR+α，MST に対応する．

5.1.2 実験 1-1: 完全グラフ

実験 1-1 では，完全グラフを用いる．この完全グラフを用いることで，オーバレイネットワークにより構築されるフルコネクト型のネットワークトポロジを想定する．ただし，本来は各ノードの流入バンド幅は 1 つであるので，完全グラフのように各ノードの次数が総ノード数 n に対して n-1 であるグラフを用いた場合，実際の合計バンド幅に対する評価と差異が生じる可能性がある．実際に通信端点となる計算機と完全グラフにおける頂点対を 1 対 1 に対応付けようとすると，スイッチに複数の計算機が接続されている LAN のようなネットワークを意識しない評価となる．だが，オーバレイネットワークを想定した場合，その頂点対の間にまた別のノードを挟むと仮定すれば，それらを考慮した上での流入バンド幅であると考える．

結果を図 12 に示す．図 12 より，(a)，(b)，(c) のいずれの場合においても FPFR+ α と MST の合計バンド幅が安定して大きいことがわかる．逆に Random と Dijkstra は (a)，(b)，(c) のいずれの場合においても，合計バンド幅が小さいことがわかる．Dijkstra では，最大の流入バンド幅を得られるノードを順々にマルチキャスト木に追加していくため，同じ経路の往復が多い．つまり辺の共有が多く発生し，合計バンド幅が小さくなると考えられる．Random は経路をランダムに生成するので，こちらも複数の辺の共有が起こる可能性が十分にあり，合計バンド幅が小さいという結果になった．DephthFirst は，バンド幅の分散値が低ければ，合計バンド幅値は大きいが，分散値が高い (b) や (c) のような場合は合計バンド幅値が小さくなってしまう．また，数種類のバンド幅が混在している実際のネットワークを想定した (c) のようなバンド幅値の分布の場合は，各手法による合計バンド幅の差が大きくなる．ただし，完全グラフの場合は，各ノードがマルチキャスト木の辺として選べる辺が多く存在するため，FPFR+ α と MST の合計バンド幅の差が小さい結果となった．

5.1.3 実験 1-2: ランダムなグラフ

実験 1-2 では，各ノードの隣接する辺数に制約を加えたグラフを用いる．完全グラフの場合に比べて，各ノードが使用できる通信リンク数が限られたメッシュ型のようなネットワークを想定する．実際の通信を想定した場合，他のトラフィックによって通信リンクが使用されているなど接続している通信リンク全てが常に使用可能であるとは限らないため，各ノードの接続リンク数に制約を加える．ただし，この場合もやはり 5.1.2 で述べたように実際の物理トポロジを想定した場合と比べてバンド幅に差異があると考えられる．

結果を図 13 に示す．図 13 では，実験 1-1 と同じく，(a)，(b)，(c) のいずれの場合においても FPFR+ α と MST の合計バンド幅が安定して大きいが，MST よりも FPFR+ α の方が大きいことがわかる．これは FPFR+ α が複数の木を用いて各ノードの余ったバンド幅を使用できるためであると考えられる．ランダムなグラフは完全グラフに比べて，使用できる辺の数に制限があるため，FPFR+ α と MST との合計バンド幅の差が実験 1-1 に比べて大きくなったと考えられる．また，リンク数に制限がある分，辺の共有数は必然的に減るので，Dijkstra の合計バンド幅は実験 1-1 に比べて大きくなっている．しかし，それでも Dijkstra の合計バンド幅は全体と比べて小さいものとなっている．Random も，実験 1-1 と同じように合計バンド幅が小さい結果となった．

実験 1-1，実験 1-2 より FPFR+ α と MST の性能が他の手法に比べて高いことがわかった．また，バンド幅の分散値が高い方が，各手法の間で合計バンド幅の差が大きくなるため，そのような環境ではマルチキャスト木の最適化がより重要であると考えられる．

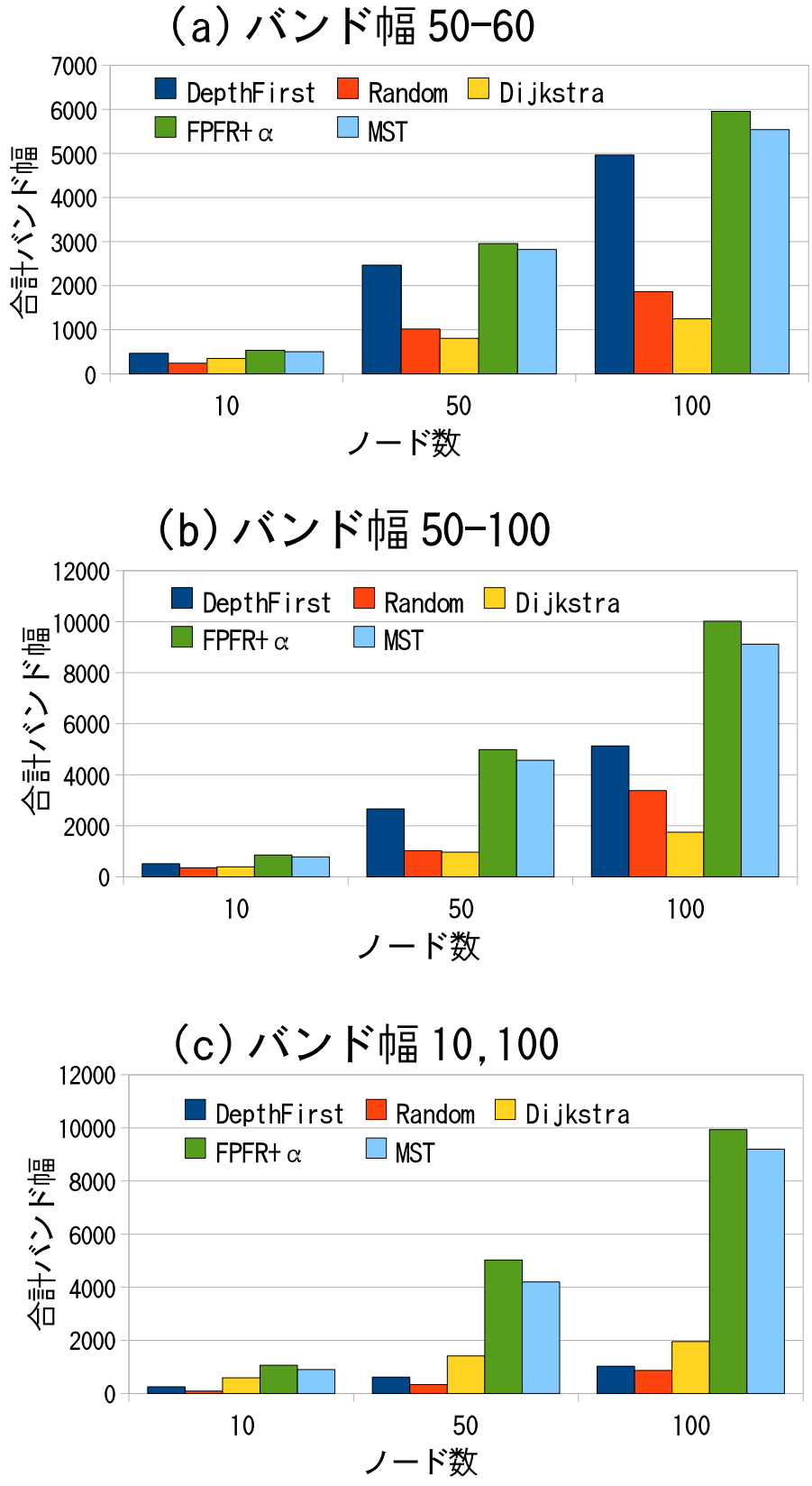

図 12: 合計バンド幅の評価 (完全グラフ)

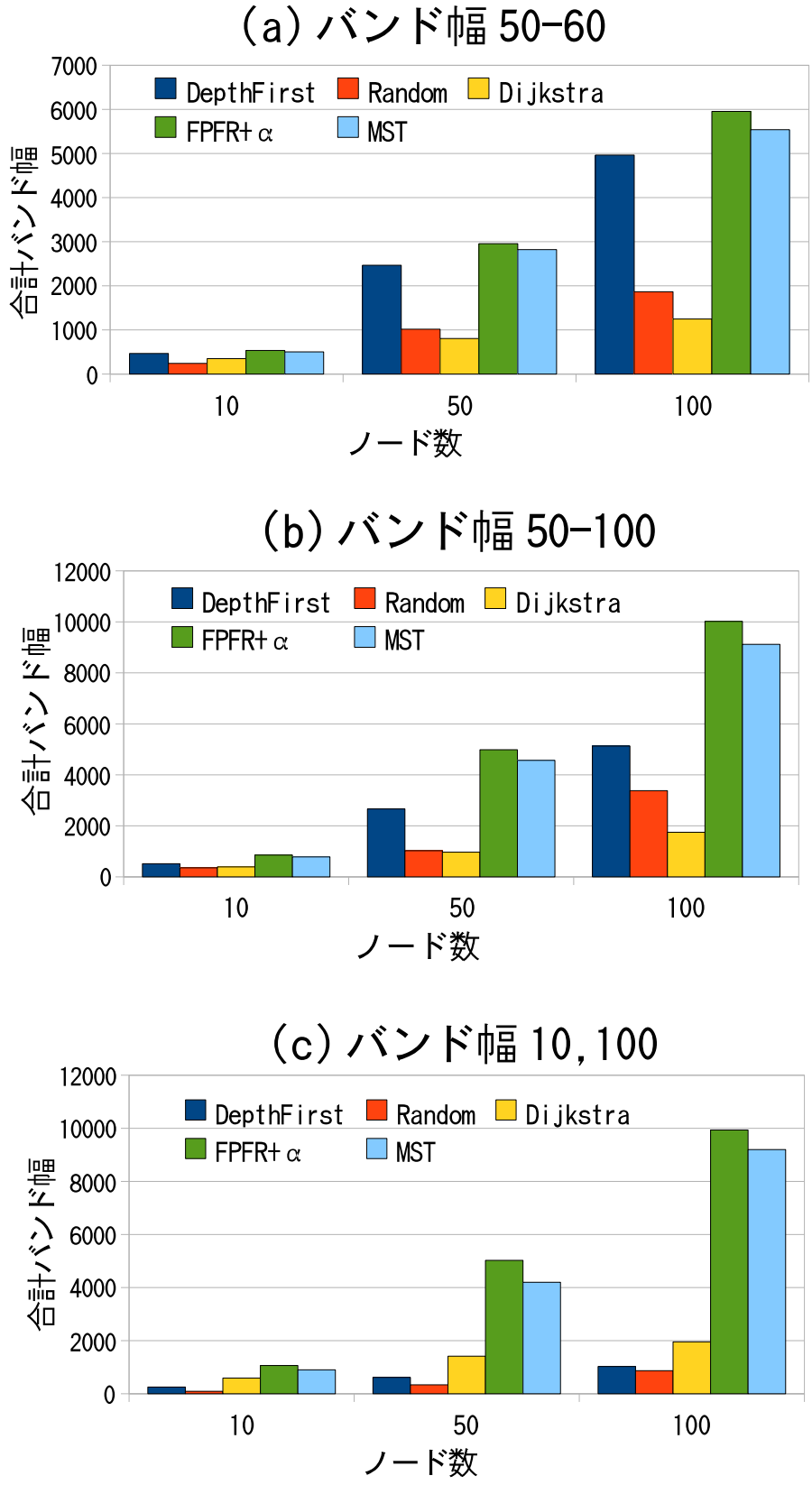

図 13: 合計バンド幅の評価 (ランダムなグラフ)

5.2 実験 2: GHS アルゴリズムの評価

GHS アルゴリズムを用いた場合，メッセージを交換して MST を構築するまでの時間を，ノード数が増加する場合と辺が増加する場合の 2 通りについて評価した．

5.2.1 実験 2-1: ノード数の増加に対するサイクル数

実験 2-1 では，ノード数の増加に対する MST 構築のサイクル数を評価した．このサイクルとは，各ノードの一連の動作である．この一連の動作を以下に示す．

- 各ノードがメッセージを受信
- 受信したメッセージに応じて，各ノードが局所的な処理を実行
- 他のノードにメッセージを送信

これを 1 サイクルとして，最終的にコアを管理するノードが終了判定するまでのサイクル数を計測した．実験 2-1 では，MST を構築することを 1 試行とし，各ノード数に対して 100 試行を行い，そのサイクル数の平均を評価した．また，与えたグラフは，ランダムなグラフであり，ノード数は 20 から 500 まで 20 ノードずづ増加させた．

結果を図 14 の上側のグラフに示す．グラフの縦軸はサイクル数で，横軸はノード数である．GHS アルゴリズムのメッセージ計算量はノード数 N，辺数 E に対して，$O(2E + 5N \cdot \log(N))$ となる．図 14 より，ノード数 N に対してサイクル数のグラフがほぼ $N \cdot \log(N)$ のグラフとないっていることがわかる．従って，実際の計算機ネットワークに適用させる場合，ノード数の増加に伴うメッセージ交換数の増加によって MST の構築自体が破綻することはないと考える．

5.2.2 実験 2-2: 辺数の増加に対するサイクル数

実験 2-2 では，ノード数を 100 に固定して，グラフの総辺数を増やした．与えたグラフはランダムなグラフである．MST を構築することを 1 試行とし，各辺数に対して 100 試行を行い，そのサイクル数の平均を評価した．

結果を図 14 の下側のグラフに示す．グラフの縦軸はサイクル数で，横軸は与えたグラフの総辺数である．図 14 より，総辺数の増加によって，サイクル数が急激に増えないことがわかる．従って，各ノードに多くの辺の接続の候補がある場合でも，ほぼ $O(N \cdot \log(N))$ で MST を構築することが可能であると考える．

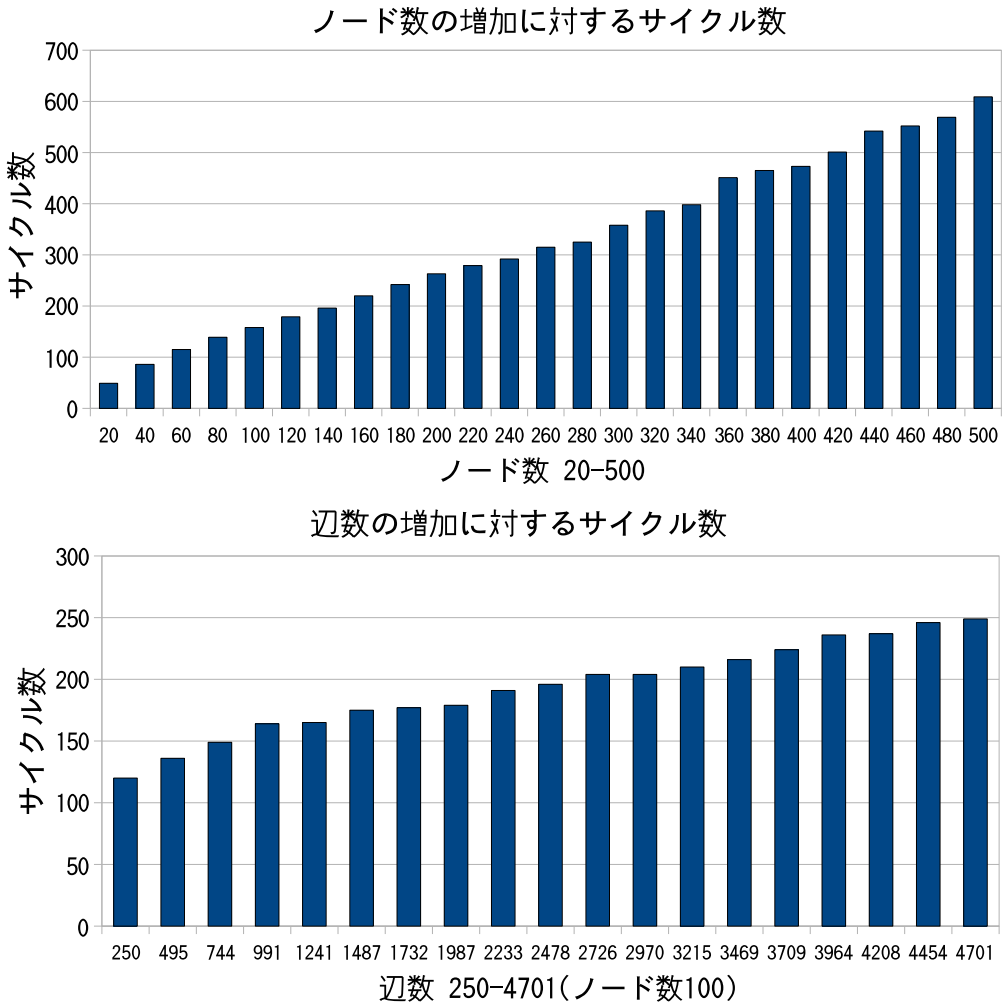

図 14: MST の構築サイクル数

5.3 実験 3: 辺の切り替えの評価

実験 3-1，実験 3-2 では GHS アルゴリズムを用いて構築した MST の辺の一部を切り替えて構築した木を評価した．問題は一度だけ変化し，各ノードは一度だけそれぞれのコネクションリストを更新して辺の切り替えを行う．

5.3.1 実験 3-1: 合計バンド幅

実験 3-1 では，バンド幅に変化があった場合に，辺の切り替えを行った場合と行わない場合とで木の性能を評価した．バンド幅は 10 から 1000 の範囲に一様分布する．ノード数は 10，50，100 の 3 通りであり，各ノード数でそれぞれ 100 回ずつ木の構築を行い，その合計バンド幅の平均を評価した．

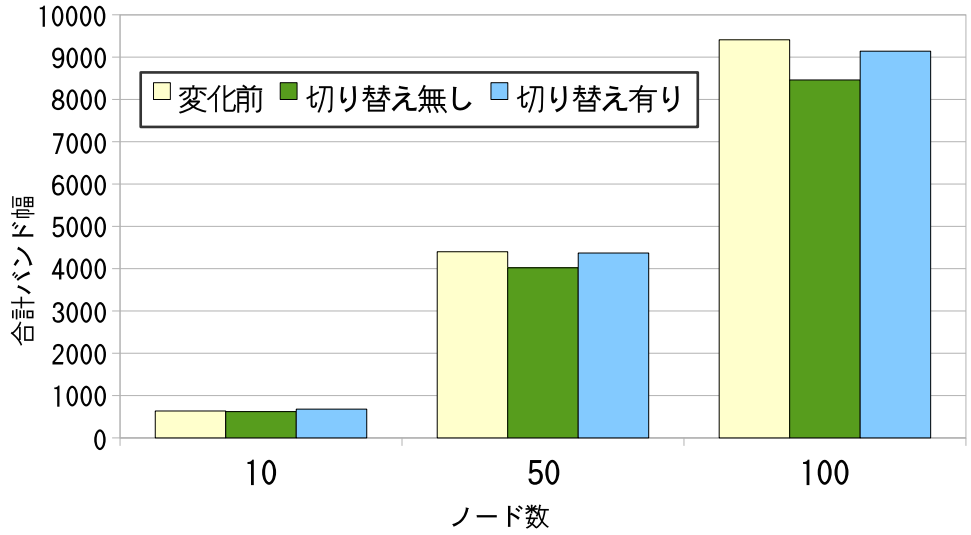

図 15: 辺の切り替え後の合計バンド幅

結果を図 15 に示す．グラフの縦軸は合計バンド幅で，横軸はノード数である．変化前とは問題の変化前の MST の合計バンド幅，切り替え無しは問題の変化後に辺を切り替えない場合の MST の合計バンド幅，切り替え有りは辺を切り替えた場合の木の合計バンド幅である．グラフの縦棒は左から順に変化前，切り替え無し，切り替え有りに対応する．図 15 より辺の切り替え無しの場合よりも辺の切り替え有りの場合の方が合計バンド幅が大きいことがわかる．従って，この辺の切り替えを行うことで，木の性能の低下をある程度防ぐことができると考える．

5.3.2 実験 3-2: サイクル数

実験 3-2 では，ノード数の変化による辺の切り替えにかかる時間を評価した．バンド幅は 10 から 1000 の範囲に一様分布する．ノード数は 10，50，100 の 3 通りであり，それぞれ 100 回ずづ木の構築を行い，そのサイクル数の平均を評価した．

結果を図 16 に示す．グラフの縦軸はサイクル数で，横軸はノード数である．左の縦棒は MST 構築時のサイクル数，右は MST の一部の辺を切り替え，木を再構築するのにかかるサイクル数である．図 16 より，どのノード数の場合でも，再構築時のサイクル数が少ないことがわかる．提案の辺の切り替えアルゴリズムでは，各葉ノード同士がほぼ非同期に辺の切り替えを行うためである．従って，ノード数の増加にかかわらず，全体として少ないメッセージの交換量で，次の木を構築できると考える．ただし，各葉ノードが同じ辺に切り替えを行おうとした場合や，一度消した辺を元に戻すような場合が多く発生すると，メッセージの交換数は増加するはずである．

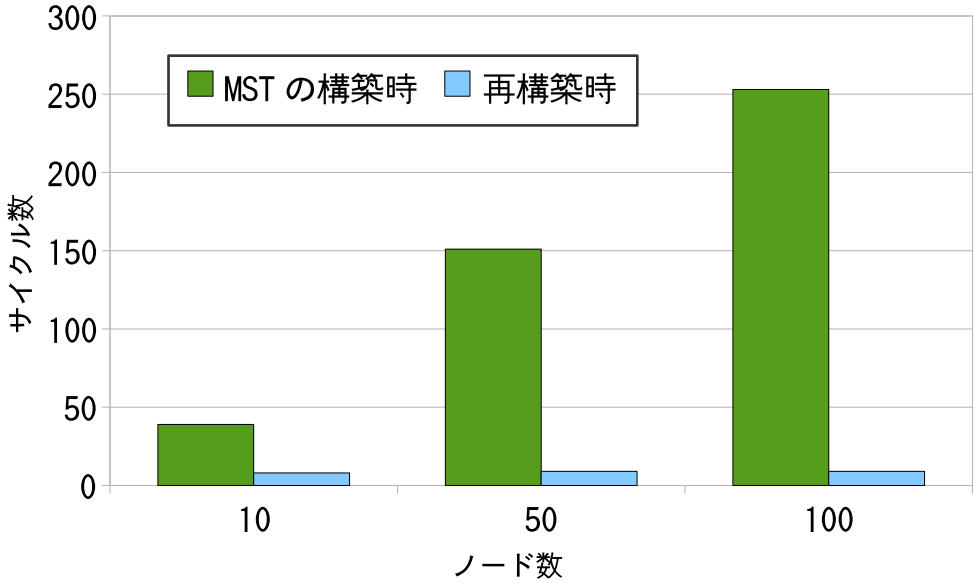

図 16: 辺の切り替えにかかるサイクル数

6　おわりに

本研究では，既存の分散アルゴリズムによって構築した木と既存のデータの送信手法を組み合わせることで，大容量データの同時配信の効率化を図ることを提案した．さらに，バンド幅の変動に応じて，少ないメッセージ交換で辺を切り替える手法を提案し，マルチキャスト木の性能低下を防ぐことを提案した．その提案についてシミュレーションを用いた実験を行い，評価することでマルチキャストの性能向上の検討を行った．提案手法のマルチキャスト木は 5.1 の実験で評価した他の手法に比べて性能が高いことがわかった．また，提案手法は分散型を想定するため，集中型を想定した手法と比べて管理ノードがない分の低コスト性と，ノード数が増えた場合の拡張性に優れている．

ただし，提案手法は，[9] のような複数の木を用いたマルチキャストと比べて性能が劣るため，分散型で複数のマルチキャスト木を構築するような手法の考察が必要である．また，本研究で示したマルチキャスト木の性能はあくまでもシミュレーションにおいての結果であり，計算機ネットワークをグラフ問題に置き換えて簡単化した分，実環境との間にギャップがあるため，より詳細に検討する必要がある．そして，マルチキャストの宛先ノード数が万単位であるような環境を想定した場合，ノード全体へのメッセージの伝搬の時間と，MST によって得られるマルチキャストの性能向上とのトレードオフについて検討する必要がある．また，ノードの参加・離脱を考慮する対故障性の問題や，実際のデータ送信の中断や再開等の問題を検討する必要がある．

謝辞

本研究を進めるにあたって，御多忙の中，研究方針を明確に御指導してくださった松尾啓志教授に深く感謝致します．津邑公暁准教授，齋藤彰一准教授にはミーティングの進捗発表の際に，研究に関するアドバイスや，励ましの言葉をいただきました．有難うございます．御多忙の中，日頃から研究の進み具合を気に掛けていただき，的確な助言と激励をくださった松井俊浩助教に深く感謝致します．

また，常日頃から多くの助言，協力をして頂いた松尾・津邑研究室，齋藤研究室の皆様に深く感謝致します．特に，週に何度か研究室に泊まる中で，多くの励ましをくださり，精神的な支えになってくださった久野友紀氏に感謝致します．そして，研究内容の相談に熱心にのってくださり，的確にアドバイスしてくださった太田和宏氏に感謝致します．ことに大嵜惇也氏，川東勇輝氏，熊崎宏樹氏は，研究に関して私と共に悩んでくださり，様々なアイデアをくださいました．有難うございます．最後になりますが，数々の助言をくださるとともに，日頃からたわいもない会話で研究に対するやる気を向上させてくださった近藤勝彦氏，今井満寿巳氏，酒井衛氏，稲葉崇文氏に感謝致します．

参考文献

[1] Gheorghe Antonoiu and Pradip K. srimani. A Self-stabilizing Distributed Algorithm to construct an arbitrary spanning tree of a connected graph. *Computers and Mathematics with Applications*, Vol. 30, pp. 1–7, 1995.

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, Vol. 1, pp. 269–271, 1959.

[3] R. G. GALLAGER, P. A. HUMBLET, and P. M. SPIRA. A Distributed Algorithm for Minimum-Weight Spanning Trees. *ACM Transactions on Programming Languages and Systems*, Vol. 5, No. 1, pp. 66–77, January 1983.

[4] Sandeep K. S. Gupta and Pradip K. srimani. Self-stabilizing multicast protocols for adhoc networks. *J.Parallel Distib. Comput.*, Vol. 63, No. 1, pp. 87–96, 2003.

[5] Lisa Higham and Zhiying Liang. Self-stabilizing minimum spanning tree construction on message-passing networks. In *DISC*, pp. 194–208, 2001.

[6] Rauf Izmailov, Samrat Ganguly, and Nan Tu. Fast Parallel File Replication In Data Grid. In *Future of Grid Data Environments Workshop*, Mar 2004.

[7] Joseph. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathmatical Society*, Vol. 7, pp. 48–50, Feb 1956.

[8] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Vol. 36, No. 1, pp. 1389–1401, Dec 1957.

[9] Takahashi.K, Saito.H, Shibata.T, and Taura.K. A Stable Broadcast Algorithm. In *Cluster Computing and the Grid, 8th IEEE International Symposium*, pp. 392–400, May 2008.

[10] Reen-Cheng Wang, Su-Ling Wu, and Ruay-Shiung Chang. A Novel Data Grid Coherence Protocol Using Pipeline-based Aggressvive Copy Method. In *Grid and Pervasive Computing*, pp. 484–495, May 2007.