情報処理学会論文誌　Vol. 50　No. 6　1549–1560 (June 2009)

推薦論文

安定したストリーム配信を実現するオーバレイマルチキャストプロトコルの設計とPlanetLab上での実証実験

池　田　和　史[†1]　ティルミー マーリンダ バデゥゲ[†1]
廣　森　聡　仁[†1]　山　口　弘　純[†1]　東　野　輝　夫[†1]

本稿では，インタラクティブアプリケーションにおける動画像配信などの用途を想定したオーバレイマルチキャストプロトコルを提案する．そのようなプロトコルへの性能要求として，インタラクティブ性を実現するために最大遅延をある閾値以下に制約できること，各ノードの動画像中継負荷を抑えるためにノードのリンク数（次数）を制限可能であること，ならびにノード離脱による配信中断が生じにくいことなどがあげられる．提案プロトコルでは，これらの実現を設計目標とし，複数のソースノードからの遅延制約，ならびに各ノードの次数制約を満たしながら，全ノードの受信安定度総和をなるべく高くするオーバレイマルチキャスト木を構築する．提案手法では，各ノードに対し，ノード安定度係数（そのノードのセッション滞留時間，転送能力など）というトポロジ非依存メトリックを定義し，あるノードについてそのすべての上流ノードの安定度の積をそのノードの受信安定度と定義する．さらに，シミュレーションと PlanetLab 上の実験による性能評価を行い，提案プロトコルの有効性を確認した．

An Application Layer Multicast Protocol for Stable Streaming and Its Evaluation on PlanetLab

Kazushi Ikeda,[†1] Thilmee M. Baduge,[†1]
Akihito Hiromori,[†1] Hirozumi Yamaguchi[†1]
and Teruo Higashino[†1]

In this manuscript, we propose a new protocol designed for interactive multimedia streaming applications. The protocol considers the heterogeneity of P2P multicast end-hosts and tries to minimize the negative impact (data outage) of end-hosts' un-announced departures. For this purpose, it concentrates on end-hosts' reliability (lifetime for instance) and constructs a shared tree called ms-DDBMSST (multiple-source Degree and Delay Bounded Maximum Stability Spanning Tree) as an overlay network that involves all the participants of the application, in a distributed manner. For a given set of nodes where some of them are senders, ms-DDBMSST is a spanning tree where the receive path stability of the entire tree is maximized while satisfying the delay-from-source constraint and degree constraint for each node. We believe that this is the first approach that defines ms-DDBMSST construction problem and presents a distributed protocol for the purpose. Our performance evaluation is based on experiments in both simulated networks and PlanetLab that strongly shows the efficiency and usefulness of the proposed protocol.

1. まえがき

インターネットの普及により，大規模なマルチメディアアプリケーション（たとえば，数千人規模で行う動画像配信やゲームなど）への需要が増加しつつある．そのようなアプリケーションにおいては，各ユーザへの情報配信を効率良く行うために，マルチキャスト配信を使用することが望ましい．オーバレイネットワークを用いず，ネットワーク層で実現されるマルチキャスト配信では，各ルータはマルチキャスト参加ノードへのマルチキャスト木を構築し，それに沿ってパケットを複製・転送するため，ネットワークの利用効率は高い反面，ネットワーク全体のルータがマルチキャスト通信に対応する必要があるなど運用上の課題も多い．一方，オーバレイネットワーク上のマルチキャストは，マルチキャスト参加ノードが，オーバレイネットワーク上にマルチキャスト木を構築し，それらのノード自身がパケットの複製・転送を行うことでマルチキャストを実現する．ネットワーク層のマルチキャストと比較し，ネットワークの利用効率は低いものの，ユニキャスト通信の組合せのみで実現できるなどの利点があり，中規模程度のグループ配信用途に向いている．これまでに，様々な用途に対し，オーバレイネットワーク上でマルチキャスト木を構築するプロトコル（アプリケーション層マルチキャスト（ALM）プロトコル）が提案されている[1)–7)]．

ALM では，各ユーザが他のユーザへのデータ中継を行うルータ的な役割も果たすため，

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
本稿の内容は 2007 年 10 月のマルチメディア通信と分散処理ワークショップにて報告され，DPS 研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

あるユーザが離脱するとそのユーザからデータ中継を受けていたユーザへの情報供給が中断される．したがって，動画像配信アプリケーションに ALM を適用する場合は，ユーザの離脱が他のユーザへ及ぼす影響をなるべく小さくするようにユーザノードを配置し，安定性の高い（各ノードの受信経路切断率が低い）マルチキャスト木を実現することが重要である．さらに，実時間インタラクション性を持つアプリケーション（たとえば講義システムなど）においては，ユーザ（視聴者）からの実時間フィードバックが求められ，各ユーザへの配信遅延をある閾値以下に抑えることが重要である．また，オーバレイネットワーク上のノードのネットワーク容量は限られており，ノードに接続するリンク数（以下，次数と呼ぶ）は，ある一定値以下に抑えられることが望ましい．

本稿では，このような設計要求に基づき，データ配信を行う複数ノード（複数ソース）からの遅延制約，ならびに各ノードの次数制約を満たしながら，全ノードの受信安定度総和をなるべく高くするようなオーバレイマルチキャスト木を構築する．以下，(i) 複数のソースが存在する環境でノード間の最大遅延と次数制限を満たしながら，ノード滞留率などによる受信安定度総和を最大化する問題の定義，(ii) その条件を満たすマルチキャスト木を分散処理で構築するプロトコルの提案，(iii) PlanetLab 上で行った実験による提案プロトコルの検証の 3 つの側面からこの問題を議論する．

2. 関連研究

近年，グループアプリケーションを対象とした ALM プロトコルが数多く提案されており，それらの研究は，遅延を重視したプロトコル，修復を重視したプロトコル，その他の QoS 要求を重視したプロトコルの 3 つに大きく分類することができる．遅延を重視したプロトコルは，ノード間の最大遅延パスにおける遅延を最小にすることを目的としたものであり，文献 1)，3)，6) のようにツリートポロジを採用する研究が多い．修復を重視したプロトコルでは，ノード離脱によるトポロジ切断の修復を研究対象としており，冗長パスによるトポロジ切断回避方法[2),8)] と，ノード離脱後に新たなパスを作成するトポロジ修復方法[9)–11)] が提案されている．また，この修復のための新たなパスについては，事前に計算しておく手法や離脱後に計算する手法が考案されている．その他の QoS 要求を重視したプロトコルとしては，文献 4)，5)，12) などがあげられ，これらの研究では帯域確保などを目的としている．

多くの既存 ALM プロトコルは，ノード間の遅延，ノードの次数などのトポロジ依存のメトリックを考慮するものであり，ノードの滞留時間や転送能力などのトポロジ非依存メトリックは長い間考慮されてなかった．文献 13) では，このトポロジ非依存メトリックを最初に扱い，実世界のトレースデータの解析に基づいたノードの滞留時間の特性と，それを利用したアルゴリズム（新しいノードを受け入れるピアノード候補として，滞留時間が長いノードを選択していくアルゴリズム）を提案している．文献 14) では，“優先度” というキーワードでトポロジ非依存メトリックを扱っている．この論文では，ノードの滞留時間とノード間のネットワーク帯域に注目し，木を構築する際にそれらをノード優先度に関連付けている．さらに，実世界のトレースデータを用いたシミュレーション実験に基づき，滞留時間優先の木構築方法と帯域優先の木構築方法の双方に対し，ノード離脱への耐性を評価した結果，後者の方法が耐性が高いという結論を得ている．また，文献 15) では，ノードの滞留時間を考慮した木構築分散アルゴリズムを提案している．

本提案は，インタラクティブマルチメディアアプリケーションでの利用を想定し，複数ソースノードの存在を考慮する点，ソースから各ノードへ遅延制約を設ける点，さらに PlanetLab 上で実験を行う点などについて既存のプロトコル研究と異なっている．

3. 問題定義

ユーザノードの集合 R，ユーザ間のユニキャスト（無向辺）集合 E からなる論理的な完全グラフネットワーク（オーバレイネットワーク）$G = (R, E)$ において，各ノード $r \in R$ に対し次数制約 $d_{max}(r)$ が，オーバレイリンク $(i, j) \in E$ に対しリンク遅延 $h(i, j)$ が与えられるものとする．ソースノードの集合を S（$S \subseteq R$），各ノードの安定度係数を $sc(r)$（$sc(r) \in [0, 1]$）でそれぞれ表す．ノードの安定度係数は，そのノードがオーバレイネットワーク上に滞留しデータ中継を行える期間やデータ中継を遅滞なく行うための配送能力など，データ中継をどの程度安定して行えるかの指標値である．ノードの安定度係数を決定する要因として，ノードの滞留時間や，計算機自身の能力，および計算機で動作する他のアプリケーションによる影響などが考えられる．ノードの滞留時間を考慮する理由として，オーバレイネットワークトポロジの安定度合いは，ノードの参加および離脱の頻度に大きく影響されることがあげられる．また，ノード自身がパケットの転送・複製を行うため，転送能力はそれらのノードの計算能力に大きく依存する．なお，安定度係数としてはノードに関するパラメータに限定するものではなく，リンクの安定性が懸念される際には，ノードに接続する各リンクの安定度を安定度係数として考慮するなど，ネットワークやアプリケーションの特性に即し，柔軟に設定することが好ましい．

ここで考える問題は，各ノード $r \in R$ の次数制約 $d_{max}(r)$ と S からの遅延制約 D_{max} を満たし，かつ全ノードの受信安定度総和（木の総受信安定度）がなるべく高い G の被覆木

T を構築する問題である．これを *multiple-source Degree and Delay Bounded Maximum Stability Spanning Tree*（*ms-DDBMSST*）と定義する．

T の総受信安定度とは，T の各ソースノードから他のノードへの経路の受信安定度の総和であり，ソースノード $s \in S$ から受信ノード $r \in R$ への経路の受信安定度（$stab_T(s,r)$ と表記）は，その経路上の各ノードの安定度係数の乗算とする．これを利用した T の受信安定度総和 $stab_T$ は以下のように表せる．

$$stab_T = \sum_{(s,r)\in S\times R} stab_T(s,r) \tag{1}$$

$$stab_T(s,r) = \begin{cases} 1 & (s = r) \\ sc(r')\cdot stab_T(s,r') & (s \neq r) \end{cases} \tag{2}$$

ここで，r' は s から r への経路上 r の上流隣接ノードである．

この ms-DDBMSST 構築問題（次数制約，遅延制約付き受信安定度最大の被覆木の構築問題）は NP 困難である．まず，この証明について議論を行う．G 上のいかなる木についても，与えられた次数制約，遅延制約，木の安定度を線形時間で判定できることは明らかである．そして，この問題は，ソースノードの数，各ノードの安定度係数を 1 に設定することにより，*Degree and Delay Bounded Spanning Tree* 構築問題と見なすことができる．このように，NP 困難な問題として知られている，*Degree and Delay Bounded Spanning Tree* 構築問題から，ms-DDBMSST 構築問題への変換は，線形時間で実行できるため，ms-DDBMSST 構築問題は NP 困難といえる．さらに，*Degree Bounded Maximum Stability Spanning Tree*（*DBMSST*）の構築問題を取り扱う．これは文献 13)–15) などで対象としている問題で，本稿で目的とする被覆木ではないが，この被覆木の構築方法と ms-DDBMSST 構築方法に共通部分があり，この問題の解説を通して，本稿で扱う被覆木問題を議論する．DBMSST 構築問題は，ms-DDBMSST 構築問題を簡略化したもので，ms-DDBMSST 構築問題の遅延制約を無限大，ソースノード数を 1 つとして取り扱う問題である．

3.1 DBMSST 構築問題

3.1.1 既 存 手 法

本項では DBMSST 構築問題について簡単に議論する．**図 1** に文献 14) で紹介されている簡単な DBMSST 構築方法を 2 つ示す．図 1 の上部に，これらの方法で木を構築する際に利用したノードの安定度係数と次数を示す．図 1 (a) では，安定度係数の高いノードを優先的に選択することで木を構築している．これを以下 s 優先と呼ぶ．一方，図 1 (b) では，

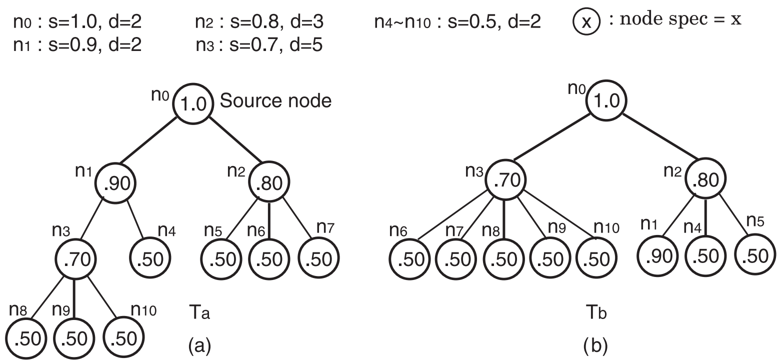

図 1　DBMSST 構築方法：(a) s 優先，(b) $s \cdot d$ 優先

Fig. 1　Two simple methods that construct DBMSST: (a) stability-first (s-first) and (b) stability-degree product-first ($s \cdot d$-first).

安定度係数と次数の乗算が高いものを優先的に選択することで木を構築しており，これを以下 $s \cdot d$ 優先と呼ぶ．いずれの構築方法においても，現在の木の中の最も高い受信安定度を提供できるノードに，選択したノードを接続する．T_a の受信安定度総和は次のように計算でき，$stab_{T_a} = \sum_{i=1}^{10} stab_{T_a}(n_0, n_i) = (1.0\cdot 1) + (1.0\cdot 1) + (0.9\cdot 1.0) + (0.9\cdot 1.0) + (0.8\cdot 1.0) + (0.8\cdot 1.0) + (0.8\cdot 1.0) + (0.7\cdot 0.9\cdot 1.0) + (0.7\cdot 0.9\cdot 1.0) + (0.7\cdot 0.9\cdot 1.0) = 8.09$ となる．同様に T_b の受信安定度総和 $stab_{T_b}$ は 7.9 となり，この例においては，s 優先構築方法がより高い受信安定度を実現していることが分かる．

3.1.2 集中構築方法

本項では，上記の 2 つの構築方法より優れている木構築方法を提案する．これは，ノードの安定度係数と次数に加えて，残ノード数（未接続のノード数）を考慮する方法で，接続先を選択するプロセスと，次に接続するノードを選択するプロセスの 2 つのプロセスから構成される．最初に，接続先選択プロセスによって，現在の木の中で最も受信安定度が高く，空次数があるノードを接続先として選択する．次に，接続するノードを選択するために，“推定受信安定度総和” を利用する．このプロセスでは，ノードを追加する各ステップにおいて，最終的に構築される被覆木の受信安定度総和の推定値を算出し，その値をもとに次に接続するノードを選択する．このように，既存手法の s 優先，$s \cdot d$ 優先の構築方法に比べ，この手法はより全体の受信安定度を改善する方針で被覆木を構築しているといえる．

以下，提案手法の詳細を説明する．それにあたり，T_{cur+u} は現在の木 T_{cur} にノード u を

追加したときの木，R_{rem} は T_{cur+u} に含まれないノード集合，$\hat{sc}$ は R_{rem} に含まれるノードの安定度係数の平均値，$\hat{d}$ は R_{rem} に含まれるノードの次数の平均値，R_{res} は T_{cur+u} に含まれるノード c のうち空次数があるノード集合，$outd(t)$ は木 t の総空次数（木全体の空次数）を表すものとする．一般的に，最適化された被覆木は比較的バランスの良い（左右にノード数の大きなバラツキがない）木になるため，T_{cur+u} 中の空次数を持つようなノードは，その下流に似たような部分木を持つと考えられる．簡単のため，以下，このような部分木を $subT$ と記す．これら部分木の受信安定度総和はほぼ同じであると仮定すると，それぞれの受信安定度総和は，以下のように推定される．R_{rem} 中のすべてのノードの安定度係数と次数は $sc(w) = \hat{sc}$ と $d(w) = \hat{d}$（$w \in R_{rem}$）で表せるものと仮定する．

$$\begin{aligned} stab_{subT} &= \sum_{source-receiver\ pair\ (s',r')\ on\ subT} stab_{subT}(s',r') \\ &= 1 + \hat{sc}\cdot\hat{d} + \hat{sc}^2\cdot\hat{d}^2 + \cdots + \hat{sc}^m\cdot\hat{d}^m \\ &= \frac{(\hat{sc}\cdot\hat{d})^{m+1} - 1}{\hat{sc}\cdot\hat{d} - 1} \end{aligned} \qquad (3)$$

ここで，$m = \log_{\hat{d}}|subT|$，$|subT| = \frac{|R|-1}{outd(T_{cur})+u}$（$|R|$ は全ノード数）である．

最後に，木 T にノード u を追加したときの受信安定度総和の推定値 $E_u[stab_T]$ は，以下のように計算できる．

$$E_u[stab_T] = stab_{T_{cur+u}} + \sum_{C} stab_{subT}\cdot stab_{T_{cur+u}}(s',r')\cdot d(r') \qquad (4)$$

C は T_{cur+u} 中のソースノードと受信ノードのペア (s',r')（$r' \in R_{res}$）を表している．提案手法では，木にノードを追加する度に R_{rem} 中の各ノードにおいて $E_u[stab_T]$ を計算し，それを最大とするノード u' を次に接続するノードとする．すべてのノードが木に接続されるまでこの手続きを繰り返し行う．

後述するシミュレーション実験において，既存手法である s 優先，$s\cdot d$ 優先の構築方法[13)–15)] と比較し，本手法が高い受信安定度総和を実現していることを示す．また，本稿の主目的である ms-DDBMSST 構築にもこの考えを採用している．

4. プロトコルの提案

木 T におけるソースノードからの遅延の最大値を $depth_T$ で表す．一般的に，$stab_T$ は各ノードの安定度係数と次数，$depth_T$ は各ノード間のリンク遅延と各ノードの次数にそれぞれ依存する．これは，ノードの安定度係数とノード間のリンク遅延のぞれぞれが，順に $stab_T$ および $depth_T$ を最適化するうえで重要な役割を果たすことを意味する．一方，ノードの滞留時間とリンク遅延の間に相関がないことが文献 13) などで知られている．同様に，ノードの転送能力とリンク遅延の間にも相関がないと考えられ，$stab_T$ と $depth_T$ は相関のないメトリックであるといえる．したがって，遅延最小木構築アルゴリズム[1)] の目的関数を，単純にこれらのメトリックスの線形和で置き換えるといった手法は利用できない．

$depth_T$ のみを最適化する問題は NP 困難であるため，本研究では $depth_T$ と $stab_T$ を考慮しながら，段階的に木を構築する手法を採用する．木の形は遅延に大きく依存するため，まず遅延を考慮して $depth_T$ を最適化し，その後 $stab_T$ を調整することで，木の大幅な組み換えを行うことなく，安定した木を構築する方針を採用している．具体的には，(a) $depth_T$ を最適化するための初期木構築を行い，引き続き，(b) $stab_T$ を最適化するための木改善処理を行う．$depth_T$ が D_{max} により制限されていることを考慮し，初期木構築に既存の遅延最小木構築アルゴリズム[1)] を適用する．これにより，$depth_T$ を極力小さくすることができ，その値は与えられた遅延制約 D_{max} より小さくなる可能性が高いと考えられる．次に，木改善処理を開始し，$stab_T$ が改善されるようにノードを移動する．この処理は，$depth_T$ に対して最適化していた初期木を $stab_T$ に対して最適化し直す目的で実行されるため，本処理によって $depth_T$ が増加してしまう可能性が高いことが容易に分かる．よって，本改善処理における各ノードの移動は，$depth_T$ への遅延制約が満たされるように行われることが重要である．なお，D_{max} に小さすぎる値を与えると初期木における $depth_T$ が D_{max} より大きくなる場合には木改善処理を実行しない．本稿では，D_{max} は実際のアプリケーションを想定して与えられ，初期木における $depth_T$ より大きいことを仮定する．

一般に，多くの参加ノードはセッション開始時刻より前に到着すると仮定できるため，初期木を集中アルゴリズムによって構築することは現実的である．初期木を構築し，セッションが開始された後には，初期木構築のような，木の全体的な再構築をともなう全体最適化は，受信者の動画受信中断を招いてしまうため不適切である．そのような場合には，分散処理により部分的に木の改善を行う．なお，セッション開始後のノード参加および離脱への配慮も重要であるが，これらに対する処理は既存手法[11)] を用いて実現するものとし，本稿では，主に初期木構築処理と木改善処理に関して議論する．木改善処理の概要を以下に述べる．あるノード x が，x をルートノードとする部分木の総受信安定度を増加させるため，x から下流へ K ホップ以内のノードからなる部分木（これを $RefT_{x,K}$ で表す）内でノードの組み換え操作（再構築）を行う．ただし，この再構築はそれにともなう再構築後の木全体の新しい遅延が S からの遅延制約 D_{max} を超えない場合のみ行う．この改善処理を葉ノー

ドから親ノードに向けてボトムアップ的に行うことにより，各ノードの次数制約ならびに S からの遅延制約を満たしながら，木の総受信安定度をなるべく高くする．

4.1　初期木構築処理

ソースノード集合 S からの最大遅延（$depth_T$）が最小になるように初期木を構築する．お互いのオーバレイリンク遅延が比較的に短くなるように，すべてのソースノードが配置されている場合には，それらを単一グループとして扱い，ルートノードと見なすことができ，単純に従来の最小遅延被覆木構築アルゴリズム[1)] が適用可能である．しかし，一般的にはソースノードがお互いに離れている場合もあるため，それらを単一グループとして扱うと余分な遅延が発生し，$depth_T$ が大きくなる可能性がある．よって，従来の最小遅延被覆木構築アルゴリズムを複数ソースノード用に拡張した下記のアルゴリズムを適用する．T_{node}, T_{edge} はそれぞれ初期木のノードと辺，R は T_{node} 以外のノードを表す．このアルゴリズムでは，S の中から 1 つのソースノード（s_0）をランダムに選択し，それを初期木とする．

$T_{node} \Leftarrow \{s_0\}$, $T_{edge} \Leftarrow \emptyset$, $S \Leftarrow s_0$, $R \Leftarrow R - s_0$;
while ($R \neq \emptyset$)
　find $r \in R \backslash T_{node}$ and $r' \in T_{node}$ that minimize $depth_{(T_{node} \cup \{r\}, T_{edge} \cup \{(r', r)\})}$
　$T_{node} \Leftarrow T_{node} \cup \{r\}$;
　$T_{edge} \Leftarrow T_{edge} \cup \{(r', r)\}$;
　$R \Leftarrow R \backslash \{r\}$;
　if (r is a source node) then $S \Leftarrow S \cup \{r\}$;
endwhile;
return (T_{node}, T_{edge});

4.2　安定度改善処理

4.2.1　情 報 収 集

ルートノード（s_0）は，情報メッセージと呼ばれるメッセージを子ノードに送信し，初期木の構築が終了したことを通知する．情報メッセージには，(i) s_0 から他のソースノードへの各パス上のノード ID のリスト，(ii) s_0 から他のソースノードへの遅延値，(iii) 最大遅延制約値（D_{max}），(iv) 後述する安定度改善処理の対象となる部分木の高さ（K）が含まれる．D_{max} と K はアプリケーションに依存する定数であり，ソースノードが指定する．情報メッセージを受信した各ノード v は，(v) ソースノード S から自身への遅延（以下，$D(s_0, v)$ で表す），(vi) ソースノード S から自身へのホップ数（以下，$H(s_0, v)$ で表す）をそのメッセージに追加して自身の子ノードに転送する．これにより，各ノードは (i) から (vi) までの情報を獲得できる．

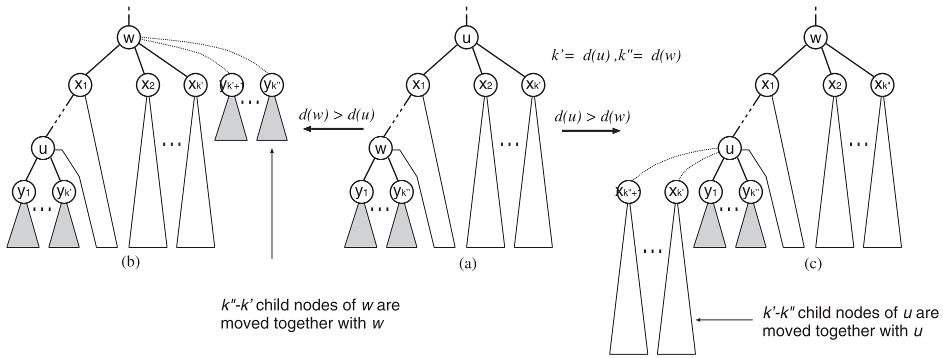

図 2　入れ換え候補選択方法の概要：(a) 初期木，(b) $d(w) > d(u)$ 時に u，w を入れ換えた後の木，(c) $d(w) < d(u)$ 時に u，w を入れ換えた後の木

Fig. 2　Node replacement: (a) the initial tree, (b) the tree swapping u for w ($d(w) > d(u)$), (c) the tree swapping u for w ($d(w) < d(u)$).

情報メッセージの送受信と同時に，ノードへの ID 割当てを行う．ルートノード r は自身にノード ID 0 を割り当て，隣接ノードには $1, \ldots, d(r)$（$d(x)$ はノード x の次数を表す）のノード ID を情報メッセージにより指定する．あるノード v が隣接ノードからノード ID i を割り当てる情報メッセージを受信した場合，ノード v は自身にノード ID i を設定し，また，ノード v の隣接ノードに対し，$i \times d_{max} + 1, \ldots, i \times d_{max} + d(v) - 1$ のノード ID を割り当て，情報メッセージを送信する．ここで d_{max} は全ノードの次数制約の最大値を示す．これにより，各ノードに一意な ID が割り当てられる．

4.2.2　ノード入れ換え

各ノードは自身の子ノードから入れ換え候補ノードを選択する[⋆1]．その際，**図 2** のように，ノード u は部分木 $RefT_{u,K}$ の総受信安定度が増加する場合にのみ，子孫ノード w を自身との入れ換え候補として選択する．u，w の入れ換えによって得られる部分木の総受信安定度増分は，下記のように表現できる．

⋆1 本稿では $K = 1$ の場合のみを議論し，$K > 1$ については今後の課題とする．

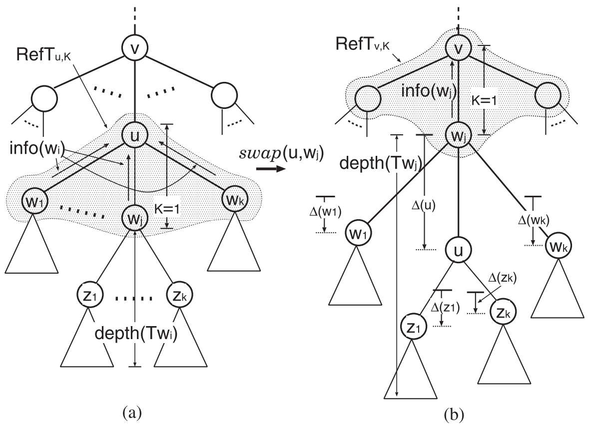

図 3　$RefT_{u,K}$ の改善の様子：(a) 改善前，(b) 改善後
Fig. 3　Improving $RefT_{u,K}$ (a) before swapping (b) after swapping.

$$\Delta_{u,w} = \begin{cases} \sum_{n=1}^{d(u)} \Big(sc(w) - sc(u)\Big) \cdot stab_{T_{x_n}} \\ + \sum_{n=d(u)+1}^{d(w)} \Big(sc(w) - sc(u) \cdot R \cdot sc(w)\Big) \cdot stab_{T_{y_n}} \\ \qquad \text{(if } d(w) \geq d(u),\ \text{Fig. 2 (b))} \\ \sum_{n=1}^{d(w)} \Big(sc(w) - sc(u)\Big) \cdot stab_{T_{x_n}} \\ + \sum_{n=d(w)+1}^{d(u)} \Big(sc(u) \cdot R \cdot sc(w) - sc(w)\Big) \cdot stab_{T_{x_n}} \\ \qquad \text{(if } d(w) < d(u),\ \text{Fig. 2 (c))} \end{cases} \tag{5}$$

ここで，$R = sc(v) \cdots sc(x_1)$（v が w の親ノードである）であり，T_{x_1} は例外的に子孫部分木である $T_{y_1} \cdots T_{y_{d(u)}}$ 以外の部分木を表すものとする．

木全体の遅延制約が満たされるように，$RefT_{u,K}$ 内のノード入れ換えを実行する必要がある．このために，各部分木に遅延増分上限を割り当て，ノード入れ換えにともなう部分木の遅延上昇が遅延増分上限以内で収まるような交換候補を選択することで，木全体の遅延制約を満たす．ノード入れ換えに関わるノードを x とし，各 x に割り当てられる遅延増分上限 $\delta(x)_{max}$ を下記の式で求める．たとえば，**図 3** (a) の u，w_j の入れ換えにおいては，u，w_j（u の子ノード），および z（w_j の子ノード）を x で表すことができる．

$$\delta(x)_{max} = \{D_{max} - D(s_f, x) - depth(T_x)\}/H(s_f, x)(x \neq z) \tag{6}$$

$$\delta(z)_{max} = \delta(w_j)_{max} \tag{7}$$

$depth(T_x)$ は x をルートノードとする部分木の最大遅延，$H(s_f, x)$ は s_f から x への経路上のホップ数である．この式は x 上で利用可能な最大遅延増分を s_f までの経路上の各ノードへ均等に分割するものである．

最後に，ノード u がこれらの計算に基づき，各 x の遅延増分 $\delta(x)$ が $\delta(x)_{max}$ 以内であり，かつ $\Delta_{u,w}$ を最大にする子ノード w を入れ換え候補として選択する．

$$\delta(x) = D(s_f, x)_{new} - D(s_f, x)_{prev} \tag{8}$$

$$D(s_f, w_j)_{new} = D(s_f, u)_{prev} - D(v, u) + D(v, w_j) \tag{9}$$

$$D(s_f, u)_{new} = D(s_f, w_j)_{new} + D(w_j, u) \tag{10}$$

$$D(s_f, w)_{new} = \begin{cases} D(s_f, w_j)_{new} + D(w_j, w) \text{ if } d(u) \geq d(w_j) \\ \qquad (w \text{ が } u \text{ に付随して移動しない}) \\ D(s_f, u)_{new} + D(u, w) \\ \qquad \text{else } (w \text{ が } u \text{ に付随して移動する}) \end{cases} \tag{11}$$

$$D(s_f, z)_{new} = \begin{cases} D(s_f, u)_{new} + D(u, z) \text{ if } d(w_j) \geq d(u) \\ \qquad (z \text{ が } w_j \text{ に付随して移動しない}) \\ D(s_f, w_j)_{new} + D(w_j, z) \\ \qquad \text{else } (z \text{ が } w_j \text{ に付随して移動する}) \end{cases} \tag{12}$$

ノード入れ換えが終了した時点で，w_j が自身の新しい親ノード v に，(i) $sc(w_j)$，(ii) $depth(T_{w_j})$，(iii) $D(w_j, v)$ の情報を含めた収集メッセージを送信する．ノード入れ換えが実現されなかった場合には，同様のメッセージが u によって v へ送信される．v がすべての子ノードから収集メッセージを受信した時点で $RefT_{v,K}$ の改善処理を開始する．この改善処理を木のルートノード s_0 に至るまで繰り返し行う．

4.2.3　複数ソースへの対応

入れ換えに関るノードがソースノード間の経路上に位置していない（入れ換え対象となるノードを根とする部分木内にソースノードが存在しない）場合に限り，複数ソースノードの存在を意識せずに上記のノード入れ換え処理が実行できる．これは，このような部分木内のノード入れ換えがあったとしても，すべてのソースノードは部分木から見て上流にいるため，あたかも 1 つのソースであると見なすことができるからである．しかし，改善処理が上流に伝搬されるにともない，この条件を満たさないノードが現れるため，特別な処理が必要となる．本稿ではこのようなノードを不規則ノードと呼ぶ．

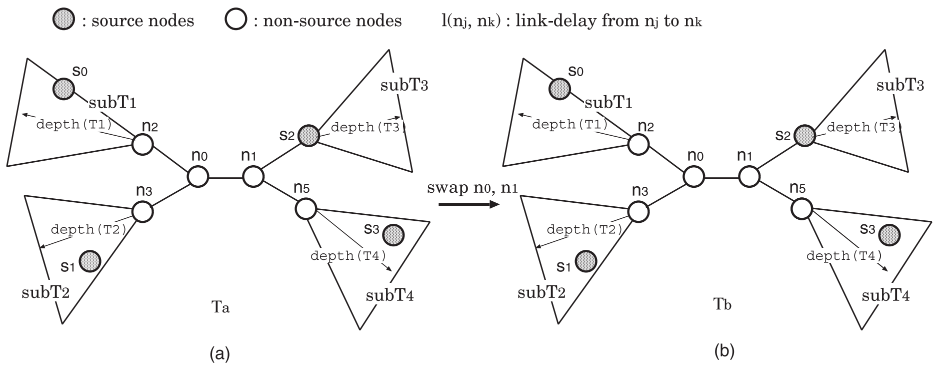

図 4　複数ソースへの対応：ソースノード間の経路上のノードの入れ換え．(a) 交換前，(b) 交換後
Fig. 4　Dealing with multiple sources: Exchanging nodes on paths between source nodes. (a) before exchange, (b) after exchange.

以降，不規則ノードに対する入れ換え処理を**図 4** をもとに述べる．ノード n_0，n_1 はそれらを交換することで，木全体の総受信安定度が上昇し，かつ遅延制約が満たされるときにのみ交換する．なお，ここで得られる総受信安定度増分 Δ_{n_0,n_1} は以下のように計算できる．

$$\Delta_{n_0,n_1} = stab_{T_b} - stab_{T_a} \tag{13}$$

$$stab_{T_a} = \sum_{0\le k\le 4}\sum_{r\in R} stab_{T_a}(s_k, r) \tag{14}$$

$$\sum_{r\in R} stab_{T_a}(s_0, r) = sc(s_0)\cdot sc(n_2)\cdot sc(n_1)\cdot sc(n_0) \cdot (stab_{subT_2} + sc(n_1)\cdot(stab_{subT_3} + stab_{subT_4})) \tag{15}$$

$\sum_{r\in R} stab_{T_a}(s_i, r)$（$i = 1, 2, 3$）も同様に表現でき，それによって $stab_{T_a}$ が求まる．$stab_{T_b}$ も同様の方法で計算でき，これらを用いて求まる Δ_{n_0,n_1} によって n_0，n_1 を入れ換えるか否かを判断できる．ただし，この入れ換えは T_b のソースからの最大遅延増分が遅延制約以内であるか否かを検討してから行うものであり，この値は $depth(T_{subT_k})$（$k = 0, 1, 2, 3$）と n_0，n_1 周辺のリンク遅延を用いることで容易に計算できる．

ここで，不規則ノードは最大遅延と受信安定度の面からお互いに影響し合うため，それらの入れ換えを異なる場所で同時に行うのは不可能であることに注目されたい．よって，改善処理が不規則ノード上で行われる前に，そのノードがルートノードに改善処理の許可を得る仕組みを導入し，同時実行を避けるようにする．ルートノードは，すべての通常ノード（非不規則ノード）の改善処理が終了した時点から，これら不規則ノードの改善処理を順に許可する．各不規則ノードは改善処理の開始前と終了後に自身に関する情報をルートノードに送信することで，ルートノードは改善処理に必要な情報を提供することが可能になる[⋆1]．

5. 評価実験

評価実験として，シミュレーション実験と PlanetLab 上での実験の 2 種類を行った．シミュレーション実験では，提案手法の定常的な性質を示すために，一般的なネットワーク環境のもとで，最大遅延による影響，ノード入れ換え手法による影響，ソースノード数による影響について評価を行った．一方，PlanetLab における実験は，多種多様なノードが存在する PlanetLab 上で実際にストリーム配信を行い，不安定な環境における提案手法の性能について評価を行った．この 2 つの評価実験から提案方式の有用性について議論を行う．

5.1 シミュレーション実験

構築されたマルチキャスト木上でストリーム配信を行ったときに，各ノードがどの程度安定してストリームを受信できるかを表す総受信安定度 $stab_T$ を用いて，提案プロトコルの評価を行った．シミュレーション実験では，オーバレイマルチキャストが対象とする，中規模のネットワークを想定し，最大 1,000 オーバレイノードを用いた．また，エンドホストが一様に配置されているネットワークを想定し，ノード間のオーバレイリンク遅延は，平均 100 [ms]，分散 20 [ms] となる標準正規分布に従うものとした．積極的にマルチキャスト配信に貢献するようなノードは限られているため，高い転送能力を提供するノードは少なく，ほとんどのノードは低い転送能力を提供する環境となるよう，ノードの次数はパレート分布に従うものとした．ここでは，GNU 科学演算ライブラリ（GSL：GNU Scientific Library）において，$p(x) = (a * b^a)/x^{(a+1)}$（$x \ge b$）で定義されているパレート分布を用い，$a = 0.6$，$b = 20$ のパレード分布をノード次数に設定した．また，文献 14) にあるように，ノード滞留時間もパレート分布に従うとされている．本実験では，$a = 1.2$，$b = 1$ となるパレート分布から導出されたノード滞留時間をノード安定度係数とした．

また，対象とするアプリケーションとしては，遠隔地の数人のパネリストによるパネル討論の映像配信などを想定している．このアプリケーションでは，各パネリストの映像ストリームがマルチキャスト配信される．そのため，最大遅延制約としてはたかだか 1 秒程度，ソースノード数としてはたかだか 5 程度が想定される．このような環境を対象とし，最大遅

⋆1 一般的に不規則ノードの数は少ないため，ここでのロック機構は長時間を要するものではない．

表 1　異なる遅延制約における提案プロトコルの振舞い
Table 1　Protocol Performance at various delay bounds.

ノード数		100	200	500	1,000
$D_{max}=$ 500 [ms]	$depth_T$ [ms]	480	472	481	463
	$stab_T$	0.46	0.46	0.38	0.34
$D_{max}=$ 1 [s]	$depth_T$ [ms]	751	879	901	934
	$stab_T$	0.55	0.53	0.48	0.43
$D_{max}=$ 1.5 [s]	$depth_T$ [ms]	1,207	1,324	1,372	1,402
	$stab_T$	0.62	0.62	0.58	0.52

表 2　異なるソースノード数におけるプロトコルの振舞い
Table 2　Protocol Performance with different number of source nodes.

ソース数	1	2	3	4	5
$stab_T$	0.65	0.63	0.62	0.58	0.56
$depth_T$ [ms]	1,427	1,398	1,390	1,452	1,421

延制約，ソースノード数，ノード入れ換えの数の影響を調査した．

[最大遅延制約による影響]

まず，提案プロトコルの最大遅延制約（D_{max}）による影響を調べた．本プロトコルは与えられた D_{max} 以内で $stab_T$ を改善するため，D_{max} が小さいと得られる $stab_T$ の改善幅も小さくなる．**表 1** にこの様子を示す．ノード数が 500 の場合においては，D_{max} を 500 ms，1 s，1.5 s と増加させると，安定度係数も 0.38，0.48，0.58 と増加しており約 1.5 倍になっている．このように，D_{max} が上昇するに従って，$stab_T$ も上昇しており，また，ノード数の増加に従い，$stab_T$ が下がっていることが分かる．これは，一般的にノード数に比例して初期木の最大遅延制約の数も増えるため，$stab_T$ の改善に利用できる遅延の余裕が少なくなるためと考えられる．

[ソースノード数による影響]

ソースノード数によるプロトコルへの影響を評価した結果（異なるソースノード数に対する $stab_T$，および $depth_T$ の値）を**表 2** に示す．D_{max} は 1.5 [s] に，オーバレイノード数は 500 に設定した．これらの結果からソースノード数の増加に従い，$stab_T$ は下がる傾向にあることが分かる．これは，ソースノード数が増加するにつれ，各ノードの移動時の遅延制約がより厳しくなるからである．

[ノード入れ換え手法による影響]

従来手法である $s \cdot d$ 優先，および s 優先のノード入れ換え方法と提案プロトコルの性能を比較した．これらの既存手法では順に $s \cdot d$ または s が高いノードを無条件に木の上流に移動させる．これに対し，提案手法は，木全体の総受信安定度が上昇する場合にのみ，ある下流ノードを上流に移動させる．$s \cdot d$ 優先，および s 優先の手法は，ノード滞留時間を対象とした既存研究[14),15)] で利用されている．ここで，提案プロトコルにおいて D_{max} を十分に大きく（$= 2$ [s]）することで自由にノード移動を可能にしたうえで，さらに，ソースノード数を 1 つにすることで，遅延制約，および複数ソースを扱ってない既存手法と同じ環境を実現した．**表 3** に実験結果を示す．オーバレイリンク遅延のばらつきが少ない環境（分散 20 ms）においては，他の手法と比較し，提案手法は高い $stab_T$ のマルチキャスト木を構築できているが，その差は小さいものとなっている．一方，遅延のばらつきが大きい環境（分散 50 ms）において，他の手法の $stab_T$ は下がっているものの，提案手法は，遅延のばらつきが少ない環境と同等の $stab_T$ を達成しており，変化に富んだ環境においても有用であるといえる．また，他の手法と比較とすると，提案手法のノード入れ換え回数は若干大きい値となっているが，高い安定度のマルチキャスト木を構築しており，安定的なマルチキャスト通信を実現しているといえる．

表 3　ノード入れ換え手法の性能比較
Table 3　Protocol Performance with different swapping policies.

	遅延（平均 100 ms 分散 20 ms）			遅延（平均 100 ms 分散 50 ms）		
	$stab_T$	$depth_T$ [ms]	ノード入れ換え数	$stab_T$	$depth_T$ [ms]	ノード入れ換え数
提案	0.64	1,238	843	0.64	1,348	932
$s \cdot d$ 優先	0.62	1,165	818	0.55	1,283	917
s 優先	0.61	1,330	827	0.52	1,324	923

5.2　PlanetLab 上での実験

実環境である PlanetLab 上での実験により，提案プロトコルの性能を評価した．実験には我々の研究グループが提案しているミドルウェア[16)] を利用した．このミドルウェアは，様々な ALM プロトコルの実装および PlanetLab 上での性能評価実験を支援する．提案プロトコルとグリーディな遅延最小木を構築するアルゴリズムを，ミドルウェア上に実装し，マルチメディアストリーム配信時の性能を比較した．

5.2.1　実 験 環 境

実験環境の詳細を以下に示す．PlanetLab には世界各地から端末が提供されており，実験に使用した端末は各地域から提供されている端末数に応じて均一に選んだ．実験でゃ総数 320 の端末を利用した．それらの CPU 能力は Pentium 3（1.2 GHz）から Pentium 4

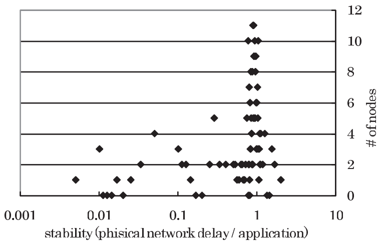

図 5　PlanetLab ノードの安定度分布
Fig. 5　PlanetLab stability distribution.

(3.4 GHz)，また，メモリ容量は 512 MB から 3.6 GB と多岐にわたっている．実験環境は Linux OS と JRE1.6 を利用した．

PlanetLab ノードの安定さを評価するために，ping コマンドを実行したときの OS レベルでの RTT（round trip time）と，Java プログラム上で ping アプリケーションを実行したときのアプリケーションレベルでの RTT を比較した．OS レベルでの ping は他のアプリケーションよりも優先的に実行され，ネットワーク遅延のみを評価するのに対し，アプリケーションレベルでの ping はネットワーク遅延に加え，ノードの状態やパケットの生成，受信イベントの待ち時間など，実際にアプリケーションでパケットを送信した際に遅延を発生させる様々な要因が含まれる．ここでは RTT（OS level)/RTT（Application level）を安定度とし，PlanetLab 上の各ノードにおいて，他の全ノードからの安定度の平均を評価した．使用した PlanetLab ノードの安定度の分布を**図 5** に示す．安定度が 1 に近いノードほど安定しているといえる．実験では半数以上のノードは安定度は 0.8–1.2 に含まれるのに対し，一部のノードは非常に小さな安定度を持ち，動作が不安定であることが分かる．PlanetLab 上の実験では，各ノードの安定度を提案手法におけるノードの安定度係数とする．

5.2.2　実験シナリオ

提案プロトコルと遅延最小木を構築するグリーディアルゴリズムについて，ストリーム配信時の平均ジッタと平均利用帯域を，(1) トポロジサイズが変化した場合，(2) 全体に占める安定ノード（安定度が 0.65 以上であるノード）の割合が変化した場合について，それぞれ評価した．

実験 (1) のシナリオを示す．初めに 20 ノードをプロトコルに従った方法でセッションに参加させ，トポロジを形成する．初めに参加したノードをソースノードとし，ソースノードから 500 kbps のストリームを配信する（マルチメディア配信アプリケーションなどを想定）．その後，プロトコルに従ってストリームをマルチキャストし，全ノードへ配信すると同時に，各ノードでのストリームのジッタと利用帯域を評価する．続いて，20 ノードを新たに参加させ，40 ノードが参加している状態を作り，同様にソースノードからストリームを配信する．同様に，20 ノードずつ増加させ，200 ノードに到達するまで，ストリームの配信性能を評価した．実験 (2) ではノード数を 200 に固定し，その中で図 5 から求められた安定ノードの割合を 100%から 75%まで 5%ずつ変化させ，各割合でプロトコルに従ってトポロジを構築し，ソースノードから 500 kbps のストリームを配信し，ジッタと利用帯域を評価した．それぞれのプロトコルにおいて PlanetLab 上で受信したストリームの品質を http://www-higashi.ist.osaka-u.ac.jp/software/stable_multicast.html より確認できる．

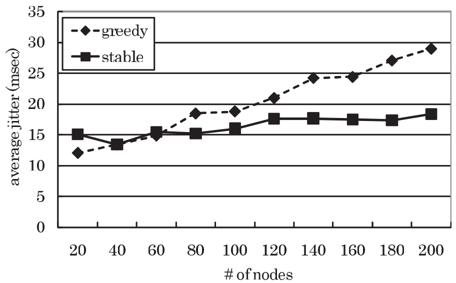

図 6　ノード数とストリームのジッタ
Fig. 6　Average jitter vs. number of nodes.

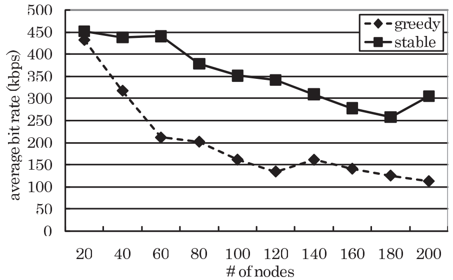

図 7　ノード数と各ノードの平均利用帯域
Fig. 7　Average bandwidth vs. number of nodes.

5.2.3　実 験 結 果

図 6，**図 7** に各プロトコルにおける平均ジッタと平均利用帯域を示す．グリーディアル

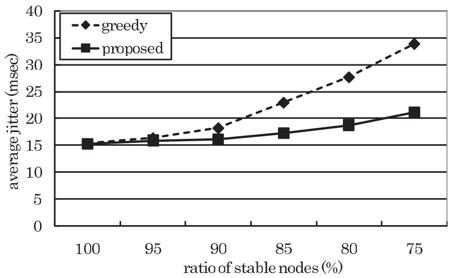

図 8　安定ノードの割合とストリームのジッタ
Fig. 8　Average jitter vs. ratio of stable nodes.

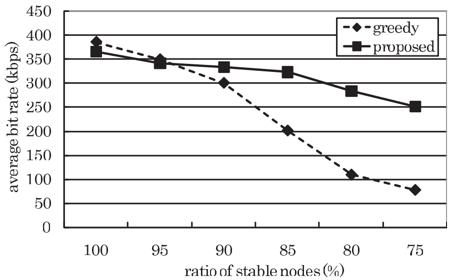

図 9　安定ノードの割合と各ノードの平均利用帯域
Fig. 9　Average bandwidth vs. ratio of stable nodes.

ゴリズムにおいてはノード数に比例しジッタの上昇と帯域の低下が激しくなっているのに対し，提案プロトコルにおいてはそれらの振舞いがより緩やかになっていることが容易に分かる．また，グリーディアルゴリズムではソースノード近くに存在しうる不安定なノードが木全体の性能を悪化させているが，提案プロトコルではこのような不安定なノードを木の比較的に下流の位置に移動させるため，より品質の高いストリーミングを提供できている．さらに，**図 8**，**図 9** より，グリーディアルゴリズムが不安定ノードの割合に大きく影響され，ストリーミングの品質が落ちていることが分かる．これらの結果より，提案プロトコルはノードの安定係数を考慮することでより高品質なストリーミングを実現しているといえる．

6. ま と め

本稿では，エンドユーザ間のユニキャスト接続からなる論理的なオーバレイネットワーク上でマルチキャスト木（被覆木）を構築するプロトコルを提案した．提案手法では，インタラクティブ性を実現するために，最大遅延がある程度抑制されること，各ノードの動画中継負荷を抑えるためにノードのリンク数を制限可能であること，ならびにノード離脱による動画中断が生じにくいことを設計目標とし，複数のソースノードからの遅延制約と各ノードの次数制約を満たしながら，全ノードの総受信安定度をなるべく高くするオーバレイマルチキャスト木を構築する．シミュレーション実験によって，提案プロトコルは従来手法より高い総受信安定度を実現可能であることが確認できた．さらに，PlanetLab 上の実験によって，遅延制約のみを考慮する既存グリーディープロトコルより，提案プロトコルは高品質なストリーミングを実現していることを確認した．それぞれのプロトコルにおいて得られたストリームファイルを，http://www-higashi.ist.osaka-u.ac.jp/software/stable_multicast.html よりダウンロードできる．

今後の方針としては，提案手法を利用したマルチメディアアプリケーションを簡単に開発できるよう，文献 16) で提案されているような P2P ミドルウェア上に提案手法を実装する．本稿では，プレゼンテーションアプリケーションを想定し，許容遅延を 2 秒，ノード安定度係数をノード滞留時間と設定したが，アプリケーションごとで適切な設定は異なると考えられる．提案方式を利用する際，アプリケーションごとで最適なマルチキャスト木を構築できるよう，ネットワークの規模に即した遅延特性や，安定度に用いるパラメータ候補などについて検討を行う．さらに，マルチキャスト木構築アルゴリズムの見直しを行う．具体的には，アプリケーション利用時に生じる通信中断をなるべく回避できるよう，ノードの追加や離脱時や，安定度係数の時間的変化を考慮した木改善アルゴリズムについて取り組む．

参 考 文 献

1) Shi, S. and Turner, J.: Routing in Overlay Multicast Networks, *Proc. IEEE INFOCOM 2002*, pp.1200–1208 (2002).
2) Roca, V. and El-Sayed, A.: A Host-Based Multicast (HBM) Solution for Group Communications, *Proc. IEEE International Conference on Networking* (*ICN '01*), Lecture Notes in Computer Science, No.2093, pp.610–619, Springer-Verlag (2001).
3) Pendarakis, D., Shi, S., Verma, D. and Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure, *Proc. 3rd USENIX Symp. on Internet Technologies and Systems* (*USITS*), pp.49–60 (2001).
4) Chu, Y.H., Rao, S.G. and Zhang, H.: A Case for End System Multicast, *Proc. ACM SIGMETRICS*, pp.1–12 (2000).
5) Banerjee, S., Bhattacharjee, B. and Kommareddy, C.: Scalable application layer

multicast, *Proc. ACM SIGCOMM 2002*, pp.205–217 (2002).
6) Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B. and Khuller, S.: Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications, *Proc. IEEE INFOCOM 2003*, pp.1521–1531 (2003).
7) ティルミー・マーリンダ・バデゥゲ，廣森聡仁，梅津高朗，山口弘純，東野輝夫：ネットワーク上で遅延最小木を動的に構築する分散型プロトコル MODE の提案と評価，情報処理学会論文誌，Vol.46, No.2, pp.483–492 (2005).
8) Francis, P.: Yoid: Extending the Internet Multicast Architecture (2002). http://www.isi.edu/div7/yoid/
9) Yang, M. and Fei, Z.: A proactive approach to reconstructing overlay multicast trees, *Proc. IEEE INFOCOM 2004*, pp.2743–2753 (2004).
10) Yamaguchi, H., Hiromori, A., Higashino, T. and Taniguchi, K.: An Autonomous and Decentralized Protocol for Delay Sensitive Overlay Multicast Tree, *Proc. 24th IEEE Int. Conf. on Distributed Computing Systems* (*ICDCS2004*), pp.662–669 (2004).
11) Baduge, T.M., Hiromori, A., Yamaguchi, H. and Higashino, T.: Design and Implemetation of Overlay Multicast Protocol for Multimedia Streaming, *Proc. 34th IEEE Int. Conf. on Parallel Processing* (*ICPP2005*), pp.41–48 (2005).
12) Nakamura, Y., Yamaguchi, H., Hiromori, A., Yasumoto, K., Higashino, T. and Taniguchi, K.: On Designing End-user Multicast for Multiple Video Sources, *Proc. 2003 IEEE Int. Conf. on Multimedia and Expo* (*ICME2003*), pp.III497–500 (2003).
13) Sripanidkulchai, K., Ganjam, A., Maggs, B. and Zhang, H.: The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points, *Proc. ACM SIGCOMM 2004*, pp.107–120 (2004).
14) Bishop, M., Rao, S. and Sripanidkulchai, K.: Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments, *Proc. IEEE INFOCOM 2006*, pp.1–13 (2006).
15) Tan, G. and Jarvis, S.A.: Improving the Fault Resilience of Overlay Multicast for Media Streaming, *IEEE Trans. Parallel and Distributed Systems*, Vol.18, No.6, pp.721–734 (2007).
16) Ikeda, K., Baduge, T.M., Umedu, T., Yamaguchi, H. and Higashino, T.: A Middleware for Implementation and Evaluation of Application Layer Multicast Protocols in Real Environments, *Proc. 17th International workshop on Network and Operating Systems Support for Digital Audio & Video* (*NOSSDAV 2007*), pp.125–130 (2007). Tools are provided: http://www-higashi.ist.osaka-u.ac.jp/software/ALM/middlewareAPI/

(平成 20 年 6 月 3 日受付)
(平成 21 年 3 月 6 日採録)

推　薦　文

本稿は，複数のソースノードからの遅延制約と各ノードの次数制約を満たしながら，全ノードの送受信安定度を可能な限り高くする，オーバレイマルチキャストプロトコルを提案している．提案プロトコルでは，ノード安定度係数という概念を導入することにより，動画配信などのインタラクティブアプリケーションにおける品質制御を実現している．ノードの安定度を考慮した手法の新規性は高く，また本稿では，シミュレーション実験，および PlanetLab 上の実験により提案プロトコルの有効性が示されており，推薦に値する．

（マルチメディア通信と分散処理研究会主査　串田高幸）

池田　和史（正会員）

平成 18 年大阪大学基礎工学部情報科学科飛び級のため中退．平成 20 年同大学大学院情報科学研究科博士前期課程修了．同年 KDDI 株式会社入社，現在 KDDI 研究所勤務．在学中，オーバレイネットワーク等の研究に従事．

ティルミー マーリンダ バデゥゲ（正会員）

平成 15 年神戸大学工学部情報知能工学科卒業．平成 17 年大阪大学大学院情報科学研究科博士前期課程修了．平成 20 年同博士後期課程修了．博士（情報科学）．同年パナソニック株式会社本社 R&D 部門入社．在学中，マルチキャスト通信等の研究に従事．

廣森　聡仁（正会員）

平成 16 年大阪大学大学院基礎工学研究科博士後期課程修了．平成 17 年株式会社エヌ・ティ・ティ・ドコモ入社．平成 20 年より大阪大学大学院情報科学研究科助教．博士（工学）．モバイルアプリケーションやモバイルネットワークの設計および性能評価に関する研究に従事．IEEE 会員．

山口　弘純（正会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院基礎工学研究科博士後期課程修了．同年オタワ大学客員研究員．平成 11 年大阪大学大学院基礎工学研究科助手．平成 14 年同大学院情報科学研究科助手．平成 19 年より同研究科准教授．博士（工学）．分散システムや無線通信プロトコルの設計および実装に関する研究に従事．IEEE，電子情報通信学会各会員．

東野　輝夫（フェロー）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院基礎工学研究科博士後期課程修了．同年同大学助手．現在，同大学大学院情報科学研究科教授．工学博士．分散システム，通信プロトコル，モバイルコンピューティング等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．本会フェロー．