オーバレイ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

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†

本論文では，ユーザノード間のユニキャスト接続を辺と見なしたオーバレイネットワーク上で，次数制約を持ちかつ最大遅延がなるべく小さい被覆木を動的に構築する分散型プロトコルMODEを提案する．MODEでは，複数のノードが同時または連続して離脱あるいは故障する（予告なしで離脱する）ような場合でも分散型の手続きによる被覆木の修復を可能とし，かつ最大遅延をなるべく小さく維持する．ns-2を利用したシミュレーション実験により，複数のノードの参加や離脱が生じる環境において，最大遅延が既存の集中型の静的アルゴリズムと比較し遜色ない値に保たれることが確認できた．

A Decentralized Protocol MODE for Minimum Delay Spanning Trees on Overlay Networks

Thilmee M. Baduge,† Akihito Hiromori,† Takaaki Umedu,† Hirozumi Yamaguchi† and Teruo Higashino†

In this paper, we present a protocol called MODE, for dynamically constructing degree-bounded minimum delay spanning trees in a decentralized way on overlay networks. The protocol repairs the spanning trees even if multiple nodes' leave operations or failures (disappearances) occur simultaneously or continuously. The simulation results using ns-2 have shown that MODE could keep reasonable maximum delay compared with the existing centralized static algorithm even if nodes' participations and disappearances occur frequently.

1. はじめに

近年のインターネットの急速な普及により，ビデオチャットなどによるインタラクティブなコミュニケーションをより大規模（たとえば数百人規模）で行いたいといった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，各ユーザが発信する情報を他の全ユーザに効率良く発信するため，ユーザ間のマルチキャスト配信を行うことが望ましい．これに対し，近年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プロトコル）がいくつか提案されてきている（文献1), 5)〜8) など）．

インタラクティブなアプリケーションにおいてはユーザ間の最大遅延がアプリケーション全体のレスポンス性能に影響するため，マルチキャスト木の最大遅延をなるべく小さくすることが重要である．さらにオーバレイネットワークにおけるマルチキャスト木の各ノードのすべてのリンクはすべてそのノードの物理ネットワークインタフェースを利用する．したがって，そのインタフェースの容量などを考慮し，ノードごとのリンク数（これを以下次数と呼ぶ）をある一定値以下に抑えることが望ましい．

本論文では，全ユーザノードの集合 V，遅延が分かっているユーザ間ユニキャスト（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）$G = (V, E)$ 上で，各ノード $v \in V$ に与えられた次数制約 $d_{max}(v)$ を満足し，かつなるべく小さい最大遅延を実現する被覆木を構築するプロトコルMODE（Minimum-delay Overlay tree construction by DEcentralized operation）を提案する．次数制約のもとで最大遅延を最小とする被覆木は *Degree-Bounded Minimum Diameter Tree*（*DBMDT*）と呼ばれ，その構築問題はNP困難である[1)]．MODEで

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

は，被覆木の効率的な情報収集を周期的に行い，ユーザノードの参加および離脱に対し，その情報を用いて高速かつ非集中型の手続きで DBMDT を構築および修復する．

ネットワークシミュレータ ns-2 を用い，200 ユーザノードを仮定したシミュレーション実験結果において，MODE はユーザノードの離脱に対し，十分短い処理時間（最悪でも約 1 秒以下）および少ない制御トラフィック量（ネットワーク全体で最大でも 200 kbps）で，既存の代表的な集中型ヒューリスティックアルゴリズム Compact Tree [1] の 1.4 倍程度の最大遅延に抑えることができた．

2. MODE プロトコルの概要と仮定

2.1 概　　要

MODE は周期的にその時点での被覆木の情報収集を短時間で行う．その情報を利用し，被覆木へ新たに参加するユーザノード（以下単にノード）に対し，被覆木の最大遅延をなるべく小さく保つような接続先ノードを決定する．また，被覆木 T からノードが離脱することにより生じる部分木間を接続することで T の構造をなるべく維持しながら新たな被覆木 T' を構成する．その際，T' の最大遅延がなるべく小さくなるように部分木間の接続箇所となるノードを選択する．具体的には，ノード離脱により生じる部分木群の中心ノードを部分木間の接続箇所とする．これにより，新たに構築された被覆木 T' の最大遅延を T のそれより小さくできる可能性がある．ここで，木の中心ノードとは，その木の最大遅延を実現する経路（最大遅延経路）上にあり，かつその経路の両端ノードからの遅延差が最も小さいノードと定義する．

たとえば **図 1** (a) の木 T において，太線で示された最大遅延経路 $x_1 - y_1 - y_2 - x_2$ 上のノード u が離脱した場合，MODE では，部分木間をそれらの中心ノード（四角で表す）を介して図 1 (b) のように接続し T' を構築する．中心ノードを部分木間の接続箇所とすることで，T' の最大遅延経路の構成要素となる可能性のある部分木間の接続ノードからの最大遅延を小さくできる（その部分木の最大遅延の 1/2）可能性があり，結果として最大遅延が小さくなる可能性がある．一方，図 1 (c) のように離脱ノードの隣接ノード間を単純に接続した T'' においては，部分木間の接続ノードからの最大遅延経路 $x1$ および $x2$ は T の最大遅延の構成要素であり，したがってこれらが最大遅延経路の構成要素の候補となる T'' はその最大遅延が T とほぼ同じとなり，その値を小さくできない可能性が高い．

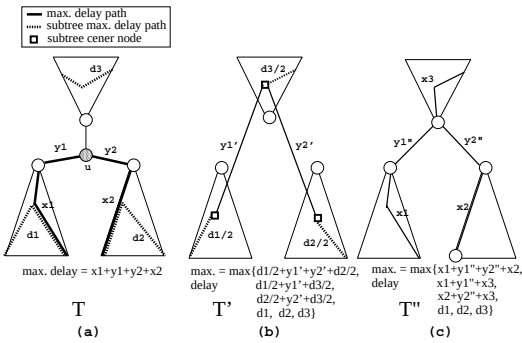

図 1 部分木間の接続方法の違いによる最大遅延の違い
Fig. 1 Different constructions by connecting sub-trees result in different maximum delay trees.

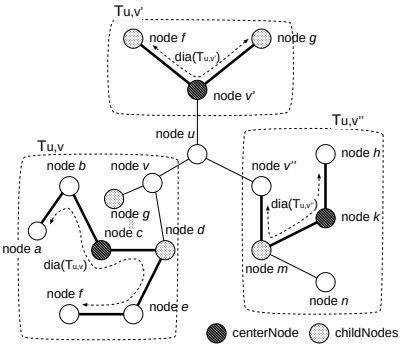

図 2 部分木の例
Fig. 2 Example sub-tree.

さらに，木の修復時間をなるべく短くするために，木の各ノード u は周期的な情報収集時に，隣接ノードが離脱した場合に生じる部分木の中心ノードなど修復に必要な情報をあらかじめ計算し保持しておく．以下 $T_{u,v}$ で，隣接する 2 ノード u，v においてノード u の離脱により生成される，ノード v を含む部分木を表す．**図 2** に各ノードが保持する情報の例を示す．簡単のため各リンクの遅延は等しいとする．部分木 $T_{u,v}$ の最大遅延経路は $[a, b, c, d, e, f]$ で，中心ノードはノード c である．ノード v は，ノード u の離脱に備え $T_{u,v}$，$T_{u,v'}$ および $T_{u,v''}$ の中心ノードの情報を情報収集時に計算し保持する．また，ノード v はノード d の離脱に備え $T_{d,v}$，$T_{d,e}$ および $T_{d,c}$ の中心ノードの情報を保持する．u が離脱した場合，その隣接ノードの 1 つが修復手続きを開始し，これらの中心ノード間に接続を指示することで新たに被覆木を構築する．これらの操作はすべて明示的な集中管理ノードを設定することなく非集中型で実行される．

2.2 仮　　定

MODE プロトコルではノードの離脱に対し以下を仮定する．

(G1) ノードの離脱は物理トポロジに影響を与えない．また，ノードは隣接ノードの離脱を十分短い時間で検出できるとする．
ノードは基本的にユーザのエンドシステムであると見なせるため，前者は妥当である．また，定期的なモニタ用パケットの送出やトランスポート層プロトコルからのシグナルなどを仮定すれば隣接ノードの離脱は検出可能であるため，後者の仮定も妥当であるといえる．

(G2) 木の最初のノードをルートノードとよび，ルートノードは離脱しないとする．木に参加するノードはルートノードのネットワークアドレスを知っているとする．
ルートノードはノードが木へ参加する際のいわゆる初期情報の役割を果たすとともに，定期的に情報収集の開始を指示する役割を果たすが，情報の集中管理は行わない．その意味でルートノードは最初のノードである必要はないが，簡単のためこのように仮定する．

(G3) MODE プロトコルで用いる制御メッセージはノードの離脱によって消失しない．
この仮定は各ノードがメッセージのキャッシングと応答メッセージなどを利用することで回避できるが，本論文では紙面の都合上その詳細な実装については省略し，議論の簡単のためこれを仮定する．

3. 木の情報収集

MODE プロトコルでは，ルートノードが定期的にその時点での木の情報収集を開始し，各ノードはその過程で隣接ノードの離脱の修復において必要となる情報を計算し保持しておく．本章ではこの情報収集方法について述べる．

3.1 ノードへの ID 割当て

まず，ルートノードは同期メッセージと呼ばれるメッセージを隣接ノードに送信し，情報収集状態に入る．その際，隣接ノードへの ID 割当ても同時に行う．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 $1,\ldots,d(r)$ のノード ID を同期メッセージにより指定する．同様に，あるノード v が隣接ノードからノード ID i を割り当てる同期メッセージを受信した場合，ノード v はその他の隣接ノードに $i \times d_{max}+1,\ldots,i \times d_{max}+d(v)-1$ のノード ID を割り当てる同期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．この ID は後述するノード参加時の手続き，およびノード離脱時の修復手続きにおいて利用される．

3.2 木情報収集

葉ノードは，その唯一の隣接ノードから同期メッセージを受信した場合，そのノードに情報収集メッセージを送信する．節ノード y は各隣接ノード x に対し，x 以外のすべての隣接ノードから情報収集メッセージを受信した時点で情報収集メッセージを送信する．ノード y は，そのすべての隣接ノードに情報収集メッセージを送信した時点で情報収集を完了し，情報収集状態から定常状態に移行する．葉ノードは隣接ノードから情報収集メッセージを受信した時点で定常状態に移行する．

以下，部分木 $T_{u,v}$ に対し，以下を部分木 $T_{u,v}$ の修復情報と呼ぶ．

- ノード v，およびその（u 以外の）隣接ノード群のネットワークアドレスとノード ID．以下 v の隣接ノード集合を $neighbor(v)$ で表す．
- $T_{u,v}$ の中心ノード（以下 $center(T_{u,v})$）のネットワークアドレスとノード ID．

前章で述べたように，MODE では，ノード u は定常状態に移行した時点で，隣接ノード v と v の隣接ノード w からなる (v,w) のすべての組に対し，$T_{v,w}$ の修復情報を保持していることを前提としている．

これを実現するため，ノード w から v への情報収集メッセージは以下の情報を含むようにしておく．

(1) $T_{v,w}$ の修復情報
(2) $T_{v,w}$ の v からの最大遅延経路（以下 $depth(T_{v,w})$）とその経路上の隣接ノード間の遅延．それらの遅延合計がこの経路の遅延となり，これを以下 $c(depth(T_{v,w}))$ で表す．
(3) $T_{v,w}$ の最大遅延経路（以下 $dia(T_{v,w})$）の遅延 $c(dia(T_{v,w}))$
(4) $T_{w,z}$ の修復情報（ただし，$z \in neighbor(w)-\{v\}$）

これらのうち (1)，(2) および (3) はノード u が $T_{u,v}$ の修復情報を計算する際に利用する．また，(4) はノード v が必要とする修復情報である．以下，ノード v がノード u 以外の各隣接ノード w から情報収集メッセージを受け取った場合に，$T_{u,v}$ の修復情報をどのように計算するかについて述べる．

ここで，$depth(T_{u,v})$ は以下のように再帰的に定義できる．ここで “@” はノードリストの連結を表し，

$c(i,j)$ でノード i, j 間の遅延値を表す．

$$depth(T_{u,v}) = [v]@depth(T_{v,j})$$

ただし，j は $c(depth(T_{v,j}))+c(v,j)$ を最大とする v の（u 以外の）隣接ノードである．

次に，$center(T_{u,v})$を再帰的に定義する．$center(T_{u,v})$ は $T_{u,v}$ の最大遅延経路 $dia(T_{u,v})$ 上に存在するが，$dia(T_{u,v})$ は，(i) 遅延が最大である $dia(T_{v,k})$（$k \in neighbor(v) - \{u\}$），(ii) v からの 2 つの最大遅延経路を組み合わせた経路 $path_v = rev(depth(T_{v,x}))@[v]@depth(T_{v,y})$（“$rev$” は経路の逆順を返す関数）のうち，遅延が大きい方となる．これに基づき，$center(T_{u,v})$ は以下のように定義できる．

$$center(T_{u,v}) = \begin{cases} center(T_{v,k}) \\ \quad (c(dia(T_{v,k})) \geq c(path_v)) \\ path_v\text{の中心ノード} \\ \quad (c(dia(T_{v,k})) < c(path_v)) \end{cases}$$

以上より，$T_{v,w}$（$w \in neighbor(v) - \{u\}$）が情報収集メッセージにより与えられたもとで，ノード v はノード u への情報収集メッセージに $T_{u,v}$ の修復情報をはじめとする (1)，(2)，(3) および (4) の値を含ませることができる．また，必要な修復情報も得ることができる．

4. ノードの参加および離脱に対する処理

本章では，前章で述べた情報収集により部分木の修復情報が各ノードに保存されているとしたうえで，ノードの参加と離脱を処理する手続きを述べる．なお，説明を容易にするために 4.1 節〜4.4 節では以下を仮定する．

(N1) 情報収集状態にあるノードは離脱しない．
(N2) あるノードの離脱に対する修復手続き中は，その隣接ノードのうち少なくとも 1 つは離脱しない．
(N3) あるノードが修復手続きにおける修復マスタ，あるいは修復サブマスタ（これらについては後述する）として選択された場合，その修復手続き中はそのノードは離脱しない．

なお，4.5 節でこの仮定が成立しない場合について述べる．

4.1 参加手続き

新たに参加するノードは，各ノードの次数制約を満たしながら現在の木 T の中心ノード $center(T)$ に最も近くなるように接続させる方法をとる．

T に参加するノードは，まずルートノードに接続先問合せメッセージを送出する☆．ここで，任意のノードは，すべての隣接ノードからの情報収集メッセージに含まれる情報を用いて木全体の中心ノードが計算できることに着目し，情報収集終了後，ルートノードは T の中心ノード $center(T)$ をあらかじめ求めておき，(その生存を確認したうえで) 問合せへの応答として，そのネットワークアドレスを通知する．なお，中心ノードがすでに離脱した場合や，最初の情報収集が行われる前，あるいは情報収集中は T の中心ノードは計算できない．この場合，ルートノードは自身を仮の中心ノードと見なし，そのネットワークアドレスを通知する．

応答を受け取った参加ノードは，指定されたノードに対し接続要求を送信する．接続要求を受け取った中心ノードは自身の次数制約を考慮し，受け入れ可能な場合は参加ノードにその旨を通知し，加えて隣接ノード群に接続要求をブロードキャスト転送する☆☆．これを受け取った隣接ノード群も中心ノードと同様の動作を行う．ただし，受け入れ可能でない場合は参加ノードに何も通知しない．なお，接続要求を転送する際に中心ノードからの遅延を加算していくことで，各ノードは中心ノードからの遅延を知ることができるためこの値も参加ノードに通知する．参加ノードはこの遅延情報と応答したノードへの遅延（これは ping などで測定可能である）をもとに，接続後の自身が中心ノードからなるべく近くなるノードを接続ノードとして選択し，木に参加する．

なお，参加したノードは現在の木に関するいっさいの情報を持たないため，情報収集状態を完了し定常状態に移行するまでは，他ノードの参加手続きおよび修復手続きにいっさい関知しない．

ここで，接続要求の受け入れ可否の判断について，次数制約が満たされるまで受け入れるのではなく，各ノードにある程度の空き次数を持たせておくことが，その後の修復手続きの際に木の最大遅延を短くしやすいことがシミュレーション実験により分かっており，シミュレーション実験では次数制約の約 8 割を受け入れ上限としている．

4.2 ノード離脱に対する修復手続き

離脱ノードを v とする．v よりノード ID が小さい

☆ ルートノードは単に代表アドレスとしての役割を果たしているだけであり，この役割は任意の複数個のノードが分担して行うことも可能である．
☆☆ この際，適切な最大転送数制限を転送の際に指定しておくことで多数のノードが参加ノードに応答するのを防止することができる．

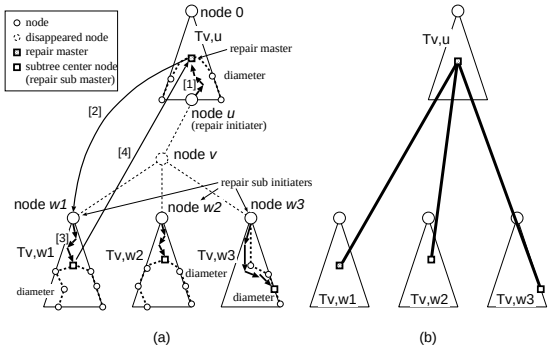

図 3 修復手続き（単一ノードの離脱）
Fig. 3 Repair procedure (single node disappearance).

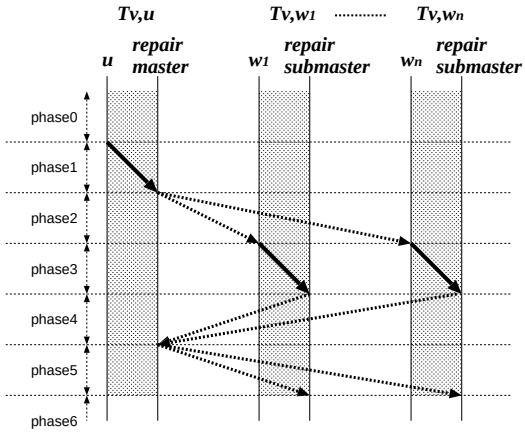

図 4 $R(v)$ のタイミングチャート
Fig. 4 Timing chart of $R(v)$.

隣接ノードを親ノードと呼び u で表し，v よりノード ID が大きい隣接ノード群を子ノードと呼び，その ID が小さい順に $w_j(1 \leq j \leq d(v)-1)$ で表す（$d(v)$ はノード v の現在の次数）．さらに，$R(v)$ はノード v の離脱に対する修復手続きを表す．

説明の簡単のため，本節ではまず情報収集後の最初の単一ノード離脱について述べる．

図 3 に単一ノード離脱の場合の修復手続きを示す．ノード v が離脱した場合，その親ノード u〔これを $R(v)$ の開始ノード（repair initiator）と呼ぶ〕がそれを検出し，$T_{v,u}$ の中心ノード〔$R(v)$ の修復マスタ（repair master）と呼ぶ〕に対し，すべての T_{v,w_j} の修復情報をノード ID を用い木上で修復マスタに転送する☆（図 3 (a) 参照）．この手順をフェーズ 1（**図 4** 参照）と呼ぶ．

部分木の修復情報を受信した修復マスタは，各 w_j（$R(v)$ の開始サブノード（repair sub-initiator）と呼ぶ）に自身のネットワークアドレスを通知し（フェーズ 2），w_j は T_{v,w_j} の中心ノード（$R(v)$ の修復サブマスタ（repair sub-master）と呼ぶ）に向けて修復マスタのアドレスをアルゴリズミックルーティングにより転送する（フェーズ 3）．これにより，各部分木 T_{v,w_j} の修復サブマスタが修復マスタ（もしくはその代替ノード）のアドレスを知ることができる．修復サブマスタは修復マスタまたはその代替ノードに対し修復接続要求を行い，4.1 節の参加手続きと同様に接続する（フェーズ 4）．なお，T_{v,w_j} の修復サブマスタに空き次数がない場合は，自身に最も近い隣接ノードに依頼する．

☆ この転送方法はアルゴリズミックルーティングと呼ばれ，3.1 節のノード ID 割当て方法に基づき，各ノードは目的ノード ID と自身のノード ID から転送すべき隣接ノード ID を一意に計算できる方法である．詳細は文献 10) を参照されたい．

以上により，部分木の中心ノード間が修復マスタを介して接続され，最大遅延がなるべく抑えられた木が構築できる．

なお，修復マスタと修復サブマスタは新たに構築されたリンクを介して互いの隣接ノード情報を交換しておく（フェーズ 5）．

4.3 複数ノードの離脱時の処理

同時並行的に発生する複数ノードの離脱に対し 4.2 節の修復手続きを単純に適用した場合はいくつかの問題が生じる．以下ではその場合の問題とその問題解決のための手続きの拡張について述べる．

(E1) v の開始ノード（v の親ノード）が離脱している場合，修復手続き $R(v)$ が開始できない．これに対し，v のすべての隣接ノードは，ノード v の離脱を検知した場合に互いの生存確認を行い，最もノード ID が小さいノード w が $R(v)$ の開始ノードとなり，その他の隣接ノードは開始サブノードとなるようにする．仮定 (N2) より，隣接ノードは少なくとも 1 つ存在するため，開始マスタは必ず決定できる．この場合，$T_{v,w}$ の中心ノードが修復マスタとなり，$T_{v,u}$ を除く残りの部分木の修復サブマスタと接続する．同様に，開始サブノード w_j が離脱している場合，修復マスタが部分木 T_{v,w_j} の修復サブマスタに対し自身のアドレスを通知できない．これに対し，修復マスタはもし w_j の子ノードのうち，アルゴリズミックルーティングに基づき w_j から T_{v,w_j} の中心ノードへの転送先隣接ノードが離脱していなければそれに通知し，そうでなければ任意の子ノードのいずれかに通知する．

(E2) $R(v)$ の開始ノードから修復マスタへの経路上

のノードが離脱していた場合，修復マスタへの通知が行えないため，メッセージ転送の過程で発見された離脱ノードの直前のノードが修復マスタとなる．同様に，修復サブマスタへの経路上のノードが離脱していた場合，メッセージ転送の過程で発見された離脱ノードの直前のノードが修復サブマスタとなる．

(E3) 直近の情報収集以降の修復手続きにより生成された枝 (v, w) が，ノード v の離脱により切断された場合，ノード v の隣接ノードは部分木 $T_{v,w}$ の修復情報を持たないため，修復できない．これに対し，部分木 $T_{v,w}$ については $R(v)$ を適用せず，v の隣接ノードのうち ID 最小のノードが $R(v)$ とは独立に w（または w の隣接ノード）に接続して局所的に修復する．また，直近の情報収集以降の他の修復手続き $R(v')$ はそのような枝 (v, w) を無視するようにする．

4.4 離脱手続きの正当性

本節では，前節の拡張 (E1)，(E2) および (E3) 後の離脱手続き $R(v)$ の正当性を証明することで MODE の正当性を述べる．ここで MODE が正当であるとは，ループあるいは孤立部分木を生じないことと定義する．

以下，v' の離脱が v の離脱と同時あるいはそれ以降に生じるものとする．v' の離脱タイミングは $R(v)$ のいずれかのフェーズ（フェーズ 0 からフェーズ 6）に分類できる（図 4 参照）．なお，フェーズ 0 は v の離脱後から $R(v)$ が開始されるまでの時間，フェーズ 6 はフェーズ 5 以降とする．また，v' は $R(v)$ における以下の役割のいずれかに分類される．

(1) $R(v)$ の開始ノードまたは開始サブノード
(2) $R(v)$ の開始ノードから修復マスタまでの経路上のノード（修復マスタ含む），または $R(v)$ の開始サブノードから修復サブマスタまでの経路上のノード（修復サブマスタ含む）
(3) それ以外のノード

このもとで，v' の離脱が $R(v)$ におけるフェーズと役割のどの組合せにおいても，$R(v)$ が正当であることを示す．また，その際の $R(v')$ の正当性を述べることで 2 ノードの修復手続き $R(v)$ と $R(v')$ の正当性を述べる．この証明は容易に任意の n ノードの修復手続きに拡張することが可能である．

まず，(1) の場合において，v' が $R(v)$ の開始ノードである場合を考える（**図 5** (a)）．図 4 より v' がフェーズ 1 以降に離脱した場合，v' の離脱は $R(v)$ に影響を与えず，$R(v)$ と $R(v')$ は独立と見なすことができる（この場合については後述する）．したがって，v' のフェーズ 0 での離脱を考える．仮定 (N2) より，少なくとも 1 つの v の子ノードが存在していることが保証されるため，拡張 (E1) よりいずれかの子ノードが開始ノードとなり（図 5 (a)），$T_{v,v'}$ を除く部分木が接続される（図 5 (b)）．なお，$R(v')$ においては，v は開始サブノードに相当するが，これについては以下の，v' が $R(v)$ における開始サブノードである場合の修復方法の議論で正当性を証明できる．さらに $R(v')$ は $T_{v',v}$ を 1 つの部分木と見なして実行されるため，$R(v)$ と $R(v')$ の適用後も正しく 1 つの木になる（図 5 (b)）．

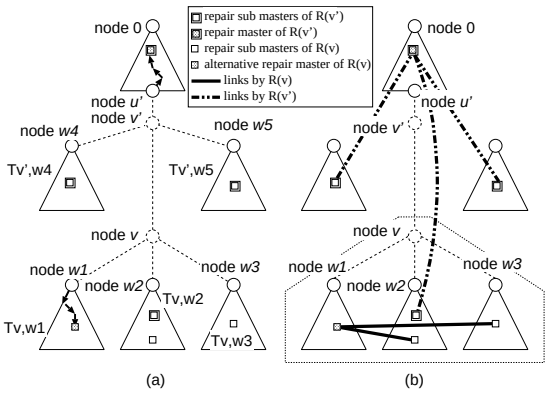

図 5 v' が $R(v)$ の開始ノードである場合
Fig. 5 v' is initiator of $R(v)$.

同様に (1) の場合において，v' が $R(v)$ の開始サブノードである場合は，図 4 よりフェーズ 2 以前の離脱を考えればよい．この場合，修復マスタに渡される部分木の修復情報には v' の隣接ノードの情報も含まれるため，拡張 (E1) より修復マスタは v' が離脱していてもその子ノードに通知することができ，$T_{v,v'}$ との接続を行うことができる．この際，$R(v')$ においては，v が開始ノードであるが，この場合は前述の v' が $R(v)$ の開始ノードである場合の修復方法の証明により正当性が証明されており，$R(v')$ により $T_{v',w_j}(1 \le j \le d(v') - 1)$ が互いに接続され，$T_{v,v'}$ が 1 つの部分木になる．よって，$R(v)$ により $T_{v,v'}$ が $T_{v',v}$ と接続された後正しく 1 つの木になる．

次に，(2) の場合において，v' が $R(v)$ の開始ノードから修復マスタまでの経路上のノード（修復マスタ含む）である場合を考える（**図 6** (a)）．v' がフェーズ 3 以降に離脱した場合，図 4 より v' は $R(v)$ に影響を与えず $R(v)$ と $R(v')$ は独立と見なすことができるため，フェーズ 2 以前での離脱を考える．この場合，$R(v)$ はアルゴリズミックルーティングを行うため，v' の離脱を v' の直前のノード（ノード w' とする）が

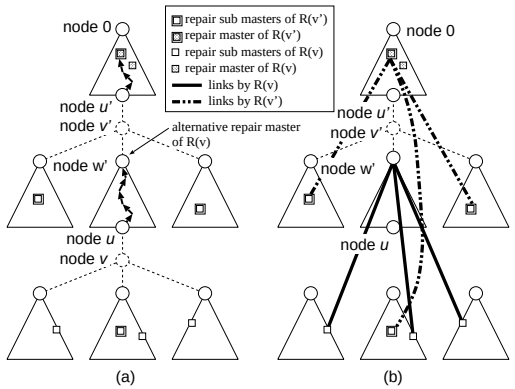

図 6 v' が $R(v)$ の開始ノードと修復マスタ間の経路上である場合
Fig. 6 v' is on the path from initiator to repair master of $R(v)$.

発見できる（図 6 (a)）．拡張 (E2) より，この場合はノード w' が修復マスタとなり，$T_{v',w'}$ を 1 つの部分木に修復する（図 6 (b)）．一方，$R(v')$ は $T_{v',w'}$ を 1 つの部分木と見なし，v' の離脱により生じた部分木を接続し修復する（図 6 (b)）．

同様に (2) の場合において，v' が $R(v)$ の開始サブノードから修復サブマスタまでの経路上のノード（修復サブマスタ含む）である場合も，v' はフェーズ 3 以前に離脱したと見なすことができ，同様の議論が成り立つ．なお，v' が修復マスタ（またはサブマスタ）である場合，仮定 (N3) よりフェーズ 2（またはフェーズ 4）からフェーズ 4（またはフェーズ 5）での離脱は生じない．フェーズ 5（またはフェーズ 6）以降は，$R(v)$ と $R(v')$ は独立と見なすことができるため後述する．以上により木全体が正しく修復される．

(3) の場合においてはどのフェーズでの離脱においても v' は $R(v)$ と無関係であり，$R(v)$ と $R(v')$ は独立と見なすことができる．

最後に，上述で $R(v)$ と $R(v')$ が独立と見なせる場合について述べる．これは (1) v' が開始ノード（開始サブノード）である場合のフェーズ 1（フェーズ 3）以降の離脱，(2) $R(v)$ の開始ノード（開始サブノード）から修復マスタ（修復サブマスタ）までの経路上のノードのフェーズ 1（フェーズ 3）以降の離脱，または修復マスタ（修復サブマスタ）のフェーズ 5（フェーズ 6）以降の離脱が相当する．これらにおいてはいずれも v' は $R(v)$ に影響を与えないことは明らかであるが，$R(v)$ により，いくつかのノードにおいてその隣接ノードのいくつかがすでに離脱した状態，また新たな隣接ノードが $R(v)$ により追加された状態で $R(v')$ が実行される．離脱した隣接ノードについては上述の v' の $R(v)$ に対する役割分類による正当性証明を v と v' を入れ換えて適用できる．また，追加された隣接ノードについては拡張 (E3) により，$R(v')$ を適用せず，特別な局所的修復（単純な隣接ノード間修復）が行われる．以上により，v' のすべての役割および離脱タイミングにおける $R(v)$ および $R(v')$ の正当性が証明された．

4.5 例外的なノード離脱に対する修復手続き

本節では，本章の冒頭で述べた仮定 (N1)，(N2) および (N3) のいずれかが成り立たない場合，または最初の情報収集以前に生じるノード離脱に対する修復手続きについて述べる．

まず，仮定 (N1) が成り立たないとする．なお，そのもとにおいても，議論の簡単のため，同期メッセージは必ず正しく全ノードに到着するものとする☆．

情報収集状態ノードが離脱した場合，そのすべての隣接ノードは情報収集状態である．これは離脱したノードに情報収集メッセージを送信できないか，または離脱したノードからの情報収集メッセージを受信できないことから分かる．この場合の修復方法を説明する．まず，離脱したノード v のすべての隣接ノードは，ノード v 以下の部分木がはじめから存在しなかったと見なして情報収集メッセージの送受信を続行し，定常状態に移行する．次に，v の各子ノード w は自身の部分木を保ったまま T のルートノードに参加接続する．これにより，この手続きは情報収集の完了後の修復手続きと同じと見なされ，正当性は容易に保証できる．

仮定 (N2)，(N3) が成り立たない場合には，修復手続きで役割を果たすノードが存在しないことにより部分木が孤立したまま修復されない可能性がある．そのような孤立部分木に存在する各ノードは同期メッセージのタイムアウトなどにより自身が孤立した部分木の存在すると判断できるため，その部分木で最もノード ID の小さいノードがルートノードに接続することで木を修復できる．しかし，ルートノードによる同期メッセージの送信間隔によっては，孤立する時間が長くなり望ましくない．これに対しては，ルートノードが短い間隔で定期的にプローブメッセージを送ることで，孤立部分木の検出間隔を短くすることも考えられる．

☆ そうでない場合，いくつかの部分木が同期メッセージを受け取れない場合があるが，この場合は情報収集の最大間隔のタイムアウトを利用することで同期メッセージの未到着を判断し，新たに参加手続きを実行するものとする．

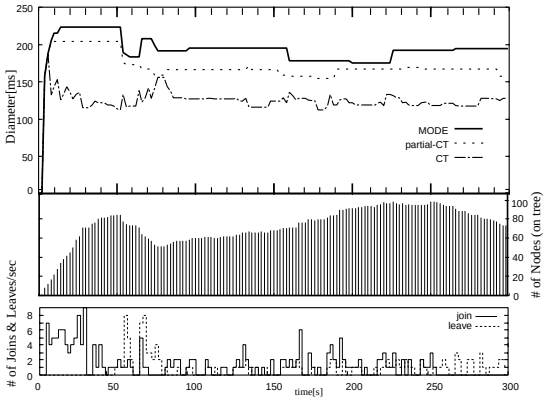

図 7 最大遅延の時間による変動
Fig. 7 Maximum delay variation.

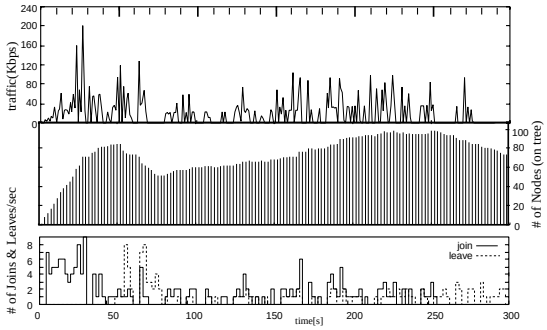

図 8 制御トラフィック総量の時間による変動
Fig. 8 Total control traific.

5. 性 能 評 価

5.1 シミュレーション設定

ns-2 を用いて MODE を実装し，シミュレーションによりその性能を評価した．実験では 200 のユーザノードを仮定した．また，参加ノード間の遅延（ユニキャスト遅延）を 10 ミリ秒〜200 ミリ秒にした．

ビデオ会議やグループウェアなどのインタラクティブなアプリケーションを考慮し，ユーザのセッションへの参加および離脱は以下のシナリオに従った．ここで，ルートノードによる同期メッセージの送信間隔は 60 秒とし，実験開始時点を時刻 0 としたとき，時刻 30 秒で最初の同期メッセージを送信するようにした．また，全ノードの次数制約を 5 に設定した．

- 各ノードはセッション中に 1 回参加する．
- 時刻 0 秒でのノード数は 1 である．
- 時刻 0 秒から 30 秒において，約 60 ノードが参加するが，いずれのノードも離脱しない．
- 時刻 30 秒から 270 秒では，時刻 30 秒，90 秒，150 秒，210 秒，270 秒で情報収集が開始される．それぞれの情報収集完了後の 60 秒間に，平均約 35 ノードが参加し，約 30 ノードが離脱する．
- 時刻 270 秒から 300 秒では，いずれのノードも参加せず，約 40 ノードが離脱する．

図 7 と**図 8** の下部にノード数の時間変化を参加/離脱回数とともに示す．

比較対象として，文献 1) で提案されている集中型アルゴリズムである CT アルゴリズムを実装した．これは，$G = (V, V \times V)$ に対し，ランダムに選択したある 1 ノードを初期木とし，現在の木 $T = (V_T, E_T)$ に属する全ノードから $V - V_T$ の全ノードに対する遅延値を考慮し，最も小さい最大遅延を実現するノード v と辺 (v, v') を加えた $(V_T \cup \{v\}, E_T \cup \{(v, v')\})$ （$v' \in V_T$ かつ v の空き次数は 1 以上）を新たな T とし，これを $V_T = V$ となるまで繰り返す集中型の静的グリーディアルゴリズムである．

CT アルゴリズムは最適値に近い最大遅延を持つ被覆木を構築できるため，提案プロトコルでの木の最大遅延の妥当性を確認するための指標として利用した．また，CT アルゴリズムは静的なアルゴリズムであるため，MODE と比較するために，以下の 2 通りで実装した．(1) ノードの参加または離脱ごと現在の木を破壊し，新しいノード集合による被覆木を CT アルゴリズムで新たに構築する．これを単に CT と表記する．CT では最大遅延はつねに適切に保たれるかわりに，頻繁なセッションの分断や，再構築による多くのリンク切断および生成が発生する．また，被覆木を計算するための集中管理ノードの存在が必要となる．(2) ノードの参加に対し，既存の木を保持したまま，最大遅延を最小に抑える接続先ノードを CT アルゴリズムを用いて計算する．また，ノードの離脱に対し，生じた部分木群から最大遅延を最も小さくできる接続点となるノードを CT アルゴリズムに従い計算し，部分木群を相互接続して木を生成する．これを partial-CT と表記する．この方法は，CT より最大遅延が増加する可能性がある一方，リンク切断および生成数は CT より少なく抑えられる．ただし，CT 同様に集中管理ノードの存在が必要となる．

5.2 実 験 結 果

［ユーザ数変動にともなう最大遅延の変動］

生成した被覆木の，セッションを通した最大遅延の変化を図 7 に示す．

図より，初期段階（時刻 0 秒〜60 秒）のノード参加時には MODE における木の最大遅延が約 225 ms であるが，離脱の発生に従い最大遅延が小さくなっており，参加/離脱が繰り返される 60 秒以降は，比較的に

表 1 最大遅延と修復手続きのリンク張り替え数
Table 1 Comparison of max. delay and link generation/dropping.

	最大遅延				リンク
	平均値			最悪値	張替数
	初期	中間	最終		平均値
CT	136	137	132	187	58.3
Partial-CT	184	164	161	204	7.5
MODE	208	188	184	224	2.9

安定して最大遅延を維持していることが分かる．また，セッションを通して，MODE での最大遅延は CT には及ばないが，partial-CT での最大遅延の変動にほぼ追随し，かつほぼ同程度の値を維持しており，MODE は有効な分散再構築方法であるといえる．

［最大遅延とリンク張り替え数］

1 セッションを通した最大遅延の平均および最悪値，修復手続きにおける切断リンク数と生成リンク数の和（リンク張り替え数）の平均値を測定し，これを異なるネットワーク遅延において 10 セッション試行したうえでの平均値を，セッションの初期期間，中間期間と最終期間に分類して**表 1** に示す．ここで，初期期間は時刻 0 秒から 50 秒で，ノードの参加が多い状態であり，中間期間は 50 秒から 240 秒で，比較的に定常な状態，最終期間は 240 秒から 300 秒で，ノード離脱が多く行われる状態を指す．表 1 より，MODE での最大遅延の平均値が初期期間において CT の約 1.52 倍，partial-CT の約 1.13 倍であるが，定常状態である中間期間においては，CT の約 1.37 倍，partial-CT の約 1.14 倍となり，最終期間においても CT の約 1.39 倍，partial-CT の約 1.14 倍に抑えられていることが分かる．また，最悪値においては CT の 1.19 倍，partial-CT の 1.09 倍であり，十分妥当な値であるといえる．平均リンク張り替え数に関しては CT に比べて約 0.05 倍，partial-CT に比べ 0.38 倍である．これらから MODE は非集中型でありながら，代表的な集中型アルゴリズムに比べて，遜色のない最大遅延が得られており，かつリンク張り替え数もきわめて少ないことが確認できた．

［制御メッセージによるトラフィック量］

セッションを通じて送受信された MODE の制御メッセージによるネットワーク全体でのトラフィック量の時間変化を図 8 に示す．MODE は時刻 30 秒付近での情報収集時に最大 200 Kbps を生じているが，これはネットワーク全体での制御トラフィックであり，被覆木の 1 リンク平均はおおよそ 3 Kbps であった．各ノードの次数はたかだか 5 であることから，各物理ネットワークリンクではたかだかこの 5 倍程度のトラフィックで抑えられる．これは現在一般化している DSL 接続におけるエンドシステムの帯域（数 Mbps）を考慮しても十分低いと考えられる．

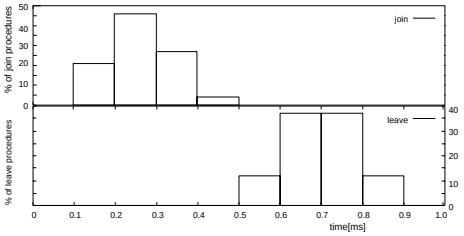

図 9 参加/修復手続きに要する時間の分布
Fig. 9 Distribution of execution time of join/repair procedures.

［参加および修復手続きの所要時間］

1 セッションを通じて実行された参加/離脱手続きそれぞれに要した時間を測定し，その分布を**図 9** に示す．これにより，すべての参加手続きが 0.5 秒以内で，また，すべての離脱手続きが 0.9 秒以内に終了していることが分かる．また，これらの手続きの実行時間の期待値はそれぞれ 0.26 と 0.70 であり，いずれも十分高速に実行されていることが分かる．

6. 関 連 研 究

オーバレイマルチキャストの分野での多くの研究は，木の安定性や，帯域と遅延の制約下での効率の良い木の実現などを目標としている．たとえば，HBM[5] は，ノードの離脱に備えてバックアップリンクを作成する集中型アルゴリズムを提案している．また，ALMI[6] は，遅延最小被覆木構築を集中的に行うアルゴリズムを提案している．Narada[7] は各ソースノードごとに最小遅延経路木をメッシュ型オーバレイネットワーク上に構築する．NICE[8] は階層型トポロジを対象とし，リーダノードが論理サブドメインを構築し，ノード管理のオーバヘッドを軽減している．

MODE プロトコルは，まず自律分散的な被覆木再構築を目指している点で上述の既存研究とは異なる．提案されている多くのプロトコルでは集中型アプローチがとられている．これらにおいては，非集中型アプローチはプロトコルの複雑さを増すことから非効率であるとされている場合も少なくない．しかし，MODE は非集中型でありながら，中間ノードでの情報集約を再帰的に行いながらメッセージ交換を行うため，きわめて少ない制御トラフィック量で必要な情報が収集できる．また，木の最適化は集中型のアプローチでのみしか考慮されていない（たとえば ALMI[6] など）が，MODE ではなるべく簡潔な分散操作で，複数のノー

ドが同時に離脱するような場合も，最大遅延がなるべく小さくなるような木が正しく生成されるように設計されている．

文献 1) は DBMDT 問題を対象としているが，1 章で述べたように集中型のヒューリスティックアルゴリズムである．我々の知る範囲では，最近の研究で，文献 9) が OMNI と呼ばれる DBMDT 分散構築プロトコルを提案しているのみである．ただし，OMNI のアプローチは MODE と異なり，SA に基づく局所的なリンク張り替えを繰り返し行うため，木の安定性などの面で望ましくない場合もある．OMNI と MODE の詳細な性能や特性の比較を行うことが我々の今後の課題の 1 つである．

7. まとめと今後の課題

本論文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，次数制約を持ち最大遅延がなるべく小さい被覆木を構築する分散型のプロトコル MODE を提案し，その性能評価を行った．MODE はセッション中でのエンドユーザの離脱に対しても動的かつ部分的に木を再構成し，分散型の設計であるにもかかわらず，既存の静的な集中型アルゴリズムと比較し遜色ない最大遅延を実現できることが分かった．

MODE を実現するミドルウェアを実装することが今後の課題である．

参 考 文 献

1) Shi, S., Turner, J. and Waldvogel, M.: Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks, *Proc. Network and Operating System Support for Digital Audio and Video* (*NOSSDAV'01*), pp.83–91, (2001).
2) Graham, R.L. and Hell, P.: On the History of the Minimum Spanning Tree Problem, *IEEE Annals of the History of Computing*, Vol.7, No.1, pp.43–57 (1985).
3) Royer, E.M. and Perkins, C.E.: Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, IETF Internet Draft, draft-ietf-manet-maodv-00.txt (2000).
4) Paul, S., Sabnani, K.K., Lin, J.C.-H. and Bhattacharyya, S.: Reliable Multicast Transport Protocol (RMTP), *IEEE Journal of Selected Areas in Communications*, Vol.15, No.3, pp.407–421 (1997).
5) Roca, V. and El-Sayed, A.: A Host-Based Multicast (HBM) Solution for Group Communications, *Proc. 2001 IEEE Int. Conf. on Networking* (*ICN'01*) (2001).
6) 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).
7) Chu, Y.-H., Rao, S.G. and Zhang, H.: A Case for End System Multicast, *Proc. ACM SIGMETRICS*, pp.1–12 (2000).
8) Banerjee, S., Bhattacharjee, B. and Kommareddy, C.: Scalable Application Layer Multicast, *Proc. ACM SIGCOMM 2002*, pp.205–217 (2002).
9) 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* (2003).
10) Huang, P. and Heidemann, J.: Minimizing Routing State for Light-Weight Network Simulation, *Proc. Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2001).

(平成 16 年 5 月 21 日受付)
(平成 16 年 9 月 3 日採録)

ティルミー マーリンダ バデゥゲ
(学生会員)
平成 15 年神戸大学工学部情報知能工学科卒業．同年大阪大学大学院情報科学研究科博士前期課程入学．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 11 年大阪大学基礎工学部情報科学科中退．同年同大学大学院基礎工学研究科博士前期課程修了．平成 16 年同大学院基礎工学研究科博士後期課程修了．工学博士．平成 15 年より日本学術振興会特別研究員．通信プロトコル/ネットワークシミュレーションに関する研究に従事．

梅津　高朗（正会員）

平成 13 年大阪大学大学院基礎工学研究科情報数理系専攻博士前期課程修了．同年同大学院博士後期課程進学．平成 14 年同大学院博士後期課程退学後，同大学院情報科学研究科助手．プロトコル合成法の応用やアドホックネットワーク用ミドルウェアの研究に従事．

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 年同大学大学院博士後期課程修了．工学博士．同年オタワ大学客員研究員．現在，大阪大学大学院情報科学研究科助手．分散システムや通信プロトコルの設計および実装に関する研究に従事．

東野　輝夫（正会員）

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．