2003－DPS－115　(7)
2003／11／14

ノード障害に対する自律分散的回復を可能とするオーバレイ遅延最小木の構築アルゴリズム

ティルミー マリンダ　廣森聡仁　山口 弘純　東野 輝夫

大阪大学 大学院情報科学研究科

{thilmee,hiromori,h-yamagu,higashino}@ist.osaka-u.ac.jp

本稿では、次数制約を持つ遅延最小のスパニングツリーをオーバレイネットワーク上で動的に構築及び維持する分散型プロトコルを提案する．提案プロトコルでは，一定時間内に，複数のノードが同時又は連続して離脱あるいは故障する（予告なしで離脱する）ような場合でもスパニングツリーの回復を可能にする．ns-2 を利用したシミュレーション実験により頻繁に複数のノードの参加，離脱が起きた場合でも，既存の集中型の静的アルゴリズムと比較してツリーの最大遅延が妥当な大きさに保たれることが確認できた．

An Autonomous and Decentralized Protocol for Delay Sensitive Overlay Multicast Tree

Thilmee Malinda　Akihito Hiromori
Hirozumi Yamaguchi　Teruo Higashino

Graduate School of Information Science and Technology
Osaka University

In this paper, we present a protocol for dynamically maintaining a degree-bounded delay sensitive spanning tree in a decentralized way on overlay networks. The protocol aims at repairing the spanning tree autonomously even if multiple nodes' leave operations or failures (disappearances) occur simultaneously or continuously in a specified period. The simulation results using ns-2 have shown that the protocol could keep reasonable diameters compared with the existing centralized static algorithm even if many 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マルチユーザービデオチャット/会議システム等多数のインタラクティブなマルチメディアグループアプリケーションが登場している．これらのアプリケーションにおいては，各参加者自身の情報をを他の全参加者に効率良く発信するために参加者間のマルチキャストインフラが必要になる．これに対し，各参加者をマルチキャストスパニングツリーで接続する peer-to-peer マルチキャストが考えられる．

このようなインタラクティブなアプリケーションにおいては全体の遅延がアプリケーションのレスポンス性能に影響を与えるため，スパニングツリーの直径 (最大遅延) を最小化することが重要である．さらに，オーバーレイマルチキャストにおいて木の各ノードのリンク数 (次数) はそのノードのネットワークインタフェース容量を考慮し，ある一定値以下に抑える必要があると考えられる．

本稿では，次数制約を持つ遅延最小スパニングツリーをオーバーレイマルチキャストツリーとして構築するプロトコルを提案する．この木構築問題は *degree-bounded minimum diameter tree(DBMDT)* 問題として知られており NP 困難な問題である [1]．この問題に対しては，文献 [1] が Compact Tree(CT) アルゴリズムと呼ばれるヒューリスティックを提案している．しかし，CT アルゴリズムはネットワークの全情報を用いる集中型の静的アルゴリズムであり，セッション中にノードの参加/離脱が発生する問題には適してない．本稿では、最大 k ノード (k はプロトコルパラメータ) の同時又は連続した離脱の際にスパニングツリーを高速にかつ分散実行的に修復するプロトコルを提案する．

ns-2 を用いて行なった実験結果により，提案アルゴリズムは分散型でありながら、より少ない計算量と少ないトラフィック量で CT アルゴリズムとほぼ同程度の直径を達成できることが分った．

以下本稿の構成を示す．2 章で DBMDT 問題と提案プロトコルの概要を述べ，3 章でスパニングツリーを修復するための木情報の分散型収集方法を述べる．4 章で提案プロトコルがノードの参加/離脱に対しどのようにスパニングツリーの拡張/修復を行なうかを述べ，5 章において実験結果を述べる．最後に 6 章おいてまとめと今後の課題を述べる．

2 提案プロトコルの概要

2.1 DBMDT

DBMDT(*degree-bounded minimum diameter tree*) は以下のように定義されるスパニングツリーである．$G = (V, E)$ が完全無向グラフを表すとする．但し，V はノード集合，E はノード間のユニキャスト接続であるオーバーレイリンク集合 $(\vec{V} \times \vec{V})$ である．また，各ノード $v \in V$ の次数制約 (最大オーバーレイリンク数) を $d_{max}(v)$ で表すとし，E の各リンクには正の実数が遅延として与えられているとする．この時 DBMDT は，直径 (最大遅延) が最小で各ノード $v \in V$ の次数 $d(v)$ が $d(v) \leq d_{max}(v)$ を満たすスパニングツリー T をいう．

2.2 プロトコル概要

提案プロトコルでは収集フェーズと定常フェーズとの 2 つのフェーズが交互に繰り返される．収集フェーズは比較的短く (実験では 2 秒以内)，定常フェーズは比較的長い (1 分) ものとする．各収集フェーズではノード v に隣接している各ノードが，ノード v の離脱の場合に孤立する部分木の情報を収集する．従って，次の定常フェーズでノード v が離脱した場合，隣接ノードが孤立した部分木の情報を既に把握しているので瞬時にこれらの部分木を再接続できる．なお，再接続の際にはなるべく部分木の中心間を接続することで直径の短縮を試みる．

2.3 フォールトモデル

ノードの離脱に対して以下のような仮定をおく．

（1） ノードの離脱は物理ネットワークに影響を与えない．また，各ノードは隣接しているノードの離脱を瞬時に検出できる．

（2） 初期ノードは離脱しない．セッションに参加する新たなノードはこのノードのネットワークアドレスを知っている．

　これは初期ノードが集中管理ノードの役割を果たす事を意味するのではなく，新ノードの参加において必要な代表ノードの役割を果たすだけである．

（3） 制御メッセージが消滅するようなノードの離脱は生じない．

以下の仮定はプロトコルの定常フェーズにおけるもので，プロトコルの正当性を容易に述べるためのものである．しかし，これらの仮定が成立しなくても木が破壊される (ループや孤立部分木が発生する) ことなく，正しく動作するように設計されている．(4.3 節を参照)

（4） 任意の定常フェーズで，節ノード v の離脱時にともなう修復手続きが終了するまで少なくとも 1 つの v の隣接ノードは離脱しない．

（5） 任意の定常フェーズで，あるノードが修復マスタ，あるいは修復マスタへの接続候補ノードとして選択された場合，修復操作が終了するまでそのノードは離脱しない．

3 木の情報収集

本節では，各収集フェーズにおける提案プロトコルの動きを説明する．

3.1 ノード ID 割り当て

提案プロトコルでは木の各ノードに一意な ID を割り当てる．これらの ID は各収集フェーズの開始時にリフレッシュされる．このため一意な ID 割り当て方法が提案されている文献 [7] のアルゴリズミックルーティングを利用する．

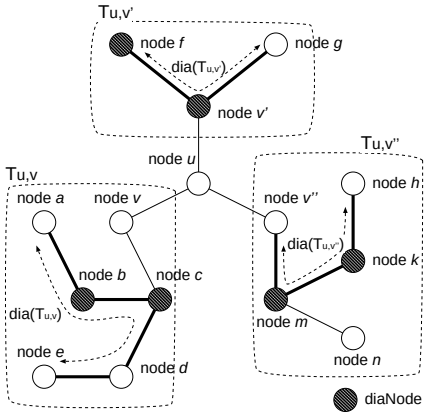

図 1: 部分木の例

前節で述べたように，初期ノードが離脱しないと仮定し (仮定 2)，このノードにノード ID 0 を割り当て，ルートノードと呼ぶ．ルートノードが定期的に同期メッセージを木にブロードキャストすることで収集フェーズが開始される．ルートノード (a とする) が隣接しているノードにメッセージを送る時，$1, \ldots, d(a)$ のノード ID をそれらに割り当てる．同様に，ノード v が隣接ノードから同期メッセージを受信し，ノード ID n が割り当てられたら，ノード v が残りの隣接ノードにメッセージを送る時に $n \times d_{max} + 1, \ldots, n \times d_{max} + d(v) - 1$ までのノード ID を割り当てる（ここで d_{max} は全ノードの次数制約の最大値を示す）．最終的に木の全てのノードが一意なノード ID を持つことになる．

各ノード v に対しそのノード ID より小さなノード ID を持つ v のノードを v の親ノード，より大きなノード ID を持つ v のノードを v の子ノードと呼ぶ．

3.2 部分木情報収集

以下 $T_{u,v}$ で，隣接している 2 つのノード u，v に対しノード 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．

同期メッセージのブロードキャストの後，各ノード u の全隣接ノードが，u の離脱により生成される部分木の情報を収集する．この情報を部分木情報と呼ぶ．この情報を利用すると，ノード u が離脱した場合に孤立した部分木を接続する修復作業を，隣接するノードのいずれかにより開始することができる．ここで，修復された木の直径をなるべく短くするために孤立した部分木をそれらの “中心ノード” で接続する．中心ノードとは部分木の直径となる経路 (以下直径経路) 上に存在し，直径経路の両端からの遅延の差が最小であるノードをいう．例えば，図 1 の $T_{u,v}$ において直径経路は [node a, node b, node c, node d, node e] で，中心ノードはノード c である (但し，各リンクの遅延は等しいと仮定する)．

そして図 1 の例では，ノード v がノード u の離脱に備えて $T_{u,v}$, $T_{u,v'}$ と $T_{u,v''}$ の部分木情報，そしてノード c の離脱に備えて $T_{c,v}$, $T_{c,b}$ と $T_{c,d}$ の部分木情報を保持する．$T_{u,v}$ の部分木情報は以下からなる．

- ノード v のネットワークアドレス (例：IP アドレス) とノード ID
- $dia(T_{u,v})$:$T_{u,v}$ の直径
- $diaNode(T_{u,v})$:中心ノード付近にある $(k+1)$ ノードの集合．各ノード $z \in diaNode(T_{u,v})$ に対し，空き次数 $d_{res}(z) = d_{max}(z) - d(z)$, ネットワークアドレス，及びノード ID も含まれる．

例えば，図 1 の部分木 $T_{u,v}$ に対して，

- $dia(T_{u,v}) = 4$
- $diaNode(T_{u,v}) = [c(1), b(2)]([c(1), d(2)]$ であっても良い) ($x(q)$ はノード x の空き次数が q であることを表している)

となる．ここで，$k = 1$，各ノード x に対し最大次数 $d_{max}(x) = 4$ であり，各リンクの遅延は等しいと仮定している．

葉ノード z が同期メッセージを受信した場合，親ノード y に情報収集メッセージを送信する．ノード y は情報収集メッセージを受信する度に，そのメッセージを送信したノード以外の全ての隣接するノードに情報収集メッセージを送信する．

各ノード v は $T_{u,v}$ の部分木情報を収集し，計算した情報をノード u への情報収集メッセージに含める．ノード v が u 以外の全ての隣接するノードから情報収集メッセージを受信したら $T_{u,v}$ の部分木情報が計算できる．このように部分木情報を再帰的に計算するために，各 $T_{u,v}$ ごと以下の補助パラメータを用いる．

- $depth(T_{u,v})$:$T_{u,v}$ の v からの最大遅延
- $depthNode(T_{u,v})$:$T_{u,v}$ の最大遅延経路を表すの v からのノードリスト．各ノード $z \in depthNode(T_{u,v})$ に対し，空き次数 $d_{res}(z) = d_{max} - d(z)$, ネットワークアドレス，及びノード ID も含む．

これら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す．

$$depth(T_{u,v}) = \max_{1 \le j \le d(v)-1} \{depth(T_{v,w_j}) + c(v, w_j)\}$$

$depth(T_{u,v})$ を最大にする w_j に対して，

$$depthNode(T_{u,v}) = [v]@depthNode(T_{v,w_j})$$

$dia(T_{u,v})$ は以下のように再帰的に定義できる．

$$dia(T_{u,v}) = \max_{1 \le j \le d(v)-1} \{dia(T_{v,w_j}), jointdepth\}$$

但し，

$$jointdepth = \max_{1 \le x,y \le d(v)-1} \{depth(T_{v,w_x}) + c(v, w_x) + depth(T_{v,w_y}) + c(v, w_y)\}$$

ここで， w_x と w_y は $w_1, \ldots, w_{d(v)-1}$ 中の 2 つの異なるノードを表す．$T_{u,v}$ の直径は (i) 部分木の直径と (ii) 1 番目と 2 番目に長い $depth$ の和のいずれか大きい方となる．

最後に $\mathrm{diaNode}(T_{u,v})$ は以下のように定義できる．

$$diaNode(T_{u,v}) = \begin{cases} diaNode(T_{v,w_j}) \ \ (\text{if } dia(T_{u,v}) = dia(T_{v,w_j})) \\ \text{“}jointpathlist\text{”} \text{ の中心ノード付近にある} \\ \text{(k+1) 個のノード} \ \ (\text{if } dia(T_{v,w}) = jointdepth) \end{cases}$$

但し，

$$jointpathlist = \mathrm{rev}(depthNode(T_{v,w_x}))@[v]@depthNode(T_{v,w_y})$$

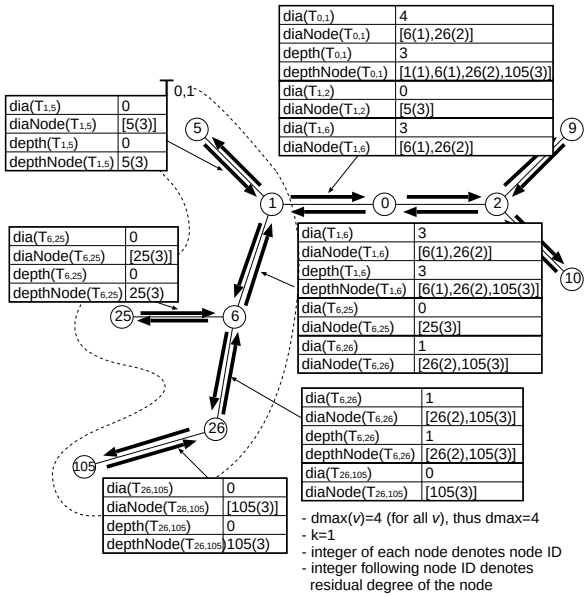

図 2: 例：部分木情報の交換例

ここで，“rev” はリストの逆順にしたものを表す関数である．

$T_{u,v}$ の直径がその部分木 $T_{v,w'}$ の直径に等しい時，$\mathrm{diaNode}(T_{u,v})$ は $\mathrm{diaNode}(T_{v,w'})$ と同じものとなる．一方 v からの 1 番目と 2 番目の depth の和が $T_{u,v}$ の直径になる時には，新直径上のノードの中から $(k+1)$ 個の候補ノードが選択される．

v から u への情報収集メッセージは以下の情報を含む．

- $T_{u,v}$ の部分木情報
- $\mathrm{depth}(T_{u,v})$ と $\mathrm{depthNode}(T_{u,v})$
- 各 v の隣接ノード w に対して $T_{v,w}$ の部分木情報 (但し，$w \neq u$)

従って，ノード u が全ての隣接するノードから情報収集メッセージを受信 (送信) し終えた時点で，ノード u は各 $T_{u,w}$ の部分木情報を持つことになる．ここで，v は u の隣接しているノードで w は v の隣接しているノードである．図 2 に情報収集メッセージが交換される様子を示す．200 ノード程度のノードの数で行なった実験において木全体の部分木情報収集にかかる時間は 2 秒以内であった．

3.3 収集フェーズでの離脱

各情報収集フェーズは比較的短い時間で完了するが，この間も幾つかのノードの離脱は起こり得る．

ノードの離脱が発生した場合必ず，孤立した部分木が生成される．ここでノード離脱のタイミングによって，(i) 孤立した部分木 t は既に同期メッセージを受信し，そのノード ID が更新されている，(ii) 同期メッセージを未受信である，との 2 状態に場合分けできる．いずれの場合においても，t の 1 つのノードのみがスパニングツリー T に再結合する．ここでの再結合方法は 4.2 節で説明する方法と同じである．また，定常フェーズでの孤立は修復作業で解決できるため，ここでは収集フェーズで起きた孤立に関してのみ議論を行なう．(i) において，離脱したノードの周辺ノードが情報収集メッセージの交換を終了していなかった場合，その離脱が収集フェーズで

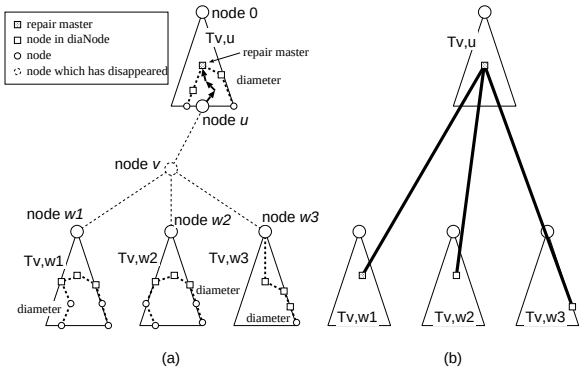

図 3: 修復手続き (単一ノードの離脱)

起きたと判断できる．そして，t のルートノードで T と再結合できる．ここで，t のルートノードは更新されている ID に基づいて選択される．一方，(ii) の t のノードにおいては，離脱が収集フェーズで起きたのか，あるいは以前の定常フェーズで起きたかの判断ができない．ここで，全てのノードがルートノードの同期メッセージの発生間隔を知っていると仮定する．これにより，離脱が定常フェーズで発生した場合には，修復作業により木が修復され，同期メッセージが t の各ノードに届くことになる．そうでなければ離脱が収集フェーズで起きたと判断できる．この場合，t の中でリーダーノードを選択し (t は スパニングツリーであるのでリーダ選択は容易である)，そのノードが T との再結合を行なう．

4 DBMDT 構築プロトコル

本節では，DBMDT 構築プロトコルについて述べる．プロトコルは定常フェーズにおける以下の手続きからなる．

4.1 参加手続き

現在の木Tに参加するノード (参加ノード) は，まずルートノードに接続可能かどうかを問い合わせる．もし可能であれば，新ノードはルートノードに接続される．もし，ルートノードに空き次数がない，あるいはルートノードとの遅延がある閾値より大きい場合，ルートノードが隣接しているノードの内 1 つを無作為に選択しそれを参加のノードに知らせ，参加ノードはこのノードに同様の問い合わせを行なう．これを繰り返すことで T に接続することができる．

このようなアプローチをとるのは，参加手続きをできるだけ単純にし，参加要求を素早く処理させるためである．この参加プロセスは木の直径を拡大せず，かつ十分な高速性を持っていることを後節で示す．

4.2 修復手続き

離脱したノード v の親ノードを u ，v の j 番目の子ノードを $w_j(1 \leq j \leq d(v)-1)$ とする．但し，ここで $d(v)$ はノード v の現在の次数を表す．

簡単のために，まず離脱するノードは 1 つ (そのノードを v で表す) とし，その後複数ノードの離脱について述べる．

図 3 に離脱するノード数が 1 の場合の修復プロセスを示す．ノード v が離脱した時，親ノードであるノード u が

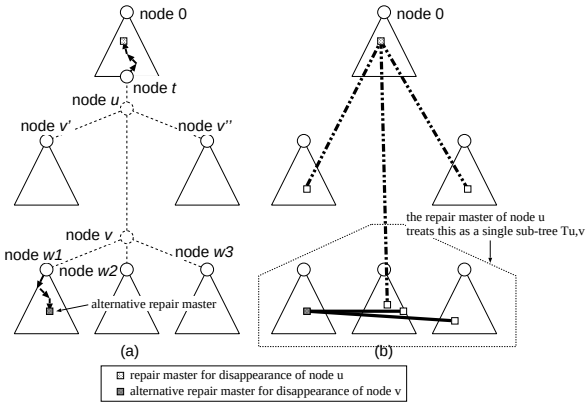

図 4: Case1(親ノード u が離脱した場合)

$diaNode(T_{v,u})$ の中心ノードに最も近く，かつ空き次数が少なくとも 1 であるノードを選択し修復プロセスを開始させる．このように選択されたノードを修復マスタと呼ぶ．ノード u は $T_{v,u}$ と $T_{v,w_j}(1 \leq j \leq d(v)-1)$ の部分木を修復マスタに送る (図 3(a) 参照)．ノード u は修復マスタの ID を利用し，文献 [7] のアルゴリズミックルーティングに従いこれらの部分木情報を送信する．修復マスタは各部分木 T_{v,w_j} の $diaNode(T_{v,w_j})(1 \leq j \leq d(v)-1)$ から中心ノードに最も近く，かつ空き次数が少なくとも 1 であるノードを選択し，そのノードにオーバーレイリンクを張る．もし，全ての部分木を修復する前に修復マスタの次数が上限に到達したら，修復マスタに隣接しているノードに残り部分木の修復を依頼する．

この手続きによって部分木の中心付近にあるノード間が接続され，修復後の木の直径は離脱前の直径より小さく，あるいは等しくなる．

次に k ノードの離脱が発生した場合を下のように場合分けして議論する．

Case1:ノード v が離脱し，その修復手続きを開始させる前に親ノード u も離脱した場合 (図 4)

2.3 節の仮定 4 によってこの時少なくとも 1 つの子ノードが存在していることが保証される．従って，ノード v が離脱した場合，v の各子ノードが，v の親ノードが存在しているかどうか，そして自分より小さい ID を持つ v の他の子ノードが存在しているかどうかをチェックする．そして，最も小さい ID を持つ子ノード w_x が $diaNode(T_{v,w_x})$ から代替修復マスタを選択する (図 4(a))．

代替修復マスタは修復マスタの代わりに $T_{v,w_j}(1 \leq j \leq d(v)-1)$ を接続する．そして，u の修復マスタから部分木 $\bigcup_j T_{v,w_j}$ の 1 つのノードにリンクを張る．従って，代替修復マスタが $T_{v,w_j}(1 \leq j \leq d(v)-1)$ のみを接続することが求められる．

この手続きは v_{i-1} が v_i の親であるようなノード列 $v_1, v_2, \ldots, v_h$ の同時離脱の場合にも容易に拡張できる．v_{i-1} の代替修復マスタが v_i も含め v_{i-1} の子ノードをルートとしている部分木を接続する．v_i の離脱のため v_i をルートとする部分木が切断されているが，これが v_i の代替修復マスタにより修復される．この操作を v_1〜v_h まで繰り返すことによって，孤立した全ての部分木が接続される．ここで，仮定 2 に従い木のルートノードが離脱することがなく，従って v_1 の親ノードが既に存在す

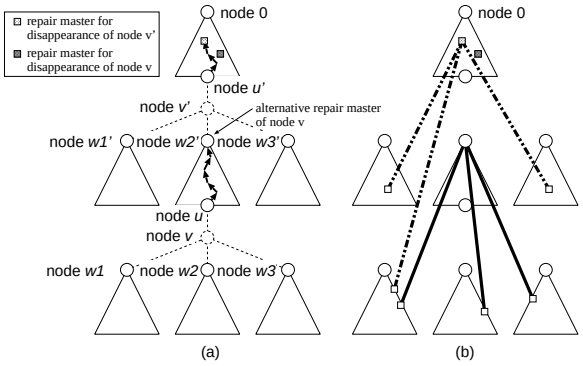

図 5: Case2(ノード u から修復マスタへのルート上のノードが離脱した場合)

ることに注意されたい．

Case2 :ノード u から修復マスタへのルート上にあるノード v' が離脱した場合 (図 5)

この時 v 'の直前にあるノードが代替修復マスタになる．例えば，図 5(a) においてノード w'_j が代替修復マスタになる．

Case3:修復マスタがリンクを張ろうとしたノードが離脱した場合

この場合の修復は容易である．どの $T_{v,w}$ の $diaNode(T_{v,w})$ にも $k+1$ ノードが存在する，あるいは $T_{v,w}$ のノード数は $k+1$ 以下である．言い変えれば，k ノードが離脱しても，$diaNode(T_{v,w})$ にまだ 1 つノードが存在している．あるいは木全体が消えており修復する必要はない．

Case4:v の修復手続きで v の祖先又は子孫ではないノード v'' が離脱した場合

この場合，両修復手続きは互いに影響しないため，独立に修復することができる．

Case5:ノード v の修復手続きの終了後，新たなノード離脱が生じた場合

孤立した部分木が修復された後，同一定常フェーズで起きる木の他のノードの離脱のために引き続いて実行される修復手続きでは以前修復された部分木は利用されない．なぜなら，ノード v' の離脱が発生し，その親ノード u' が以前修復された部分木に属するノード w を修復マスタとして選択した場合，制御メッセージは w に転送される途中でノード v の離脱により切断されたリンクに到達した場合，そのノードが *case2* のような代替修復マスタになるためである．

結果的に，既に修復されている部分木のノードが次の収集フェーズまで他のノードの修復手続きに参加せず，連続して実行される修復手続きは互いに独立である．言いかえれば，各修復手続きは収集フェーズで更新されている情報のみを利用し，元の場所から移動された部分木を無視する．さらに，定常フェーズで木に新たに参加したノードも無視する．

4.3 不意の離脱

離脱が仮定 4，5 に従わなかった場合には，修復過程で役割を果たすノードが存在しないことにより孤立した部分木が生じ得る．このように孤立した部分木に存在するノードは収集フェーズでの同期メッセージタイムアウトによって自身が孤立していることが分る．従って，次の収集フェーズの後，このように孤立した部分木は修復されるが，この場合孤立している時間が長いため望ましくない．

1 つの対策として，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が考えられる．これにより，孤立した部分木のルートノードが，自身が孤立したことが分り，木のルートノードに問い合わせを行なった後再接続することができる．

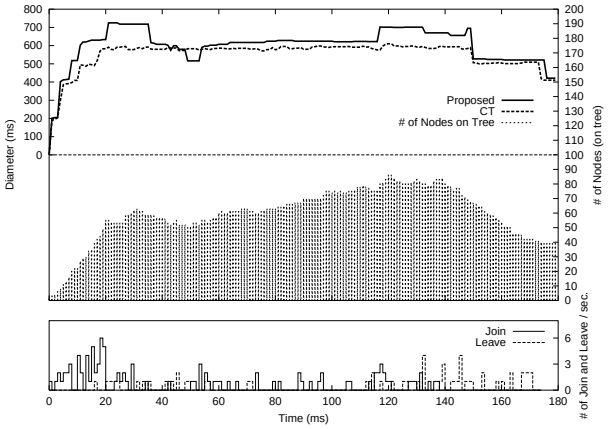

図 6: 直径の時間変化

5 性能評価

5.1 シミュレーション設定

ns-2 を用いて提案プロトコルを実装し，シミュレーションにより性能を評価した．実験において約 400 物理ノードを生成し，その内 200 ノード程度をオーバレイ参加ノードして選択した．また，エンド間の遅延を 80ms〜120ms(平均値 100ms) にした．

ビデオ会議やグループウェア等のインタラクティブなアプリケーションを考慮し，リアルタイムセッションをシミュレートする以下のシナリオを設定した．ここで，収集フェーズと定常フェーズにかかる時間を順に 1.5 秒と 58.5 秒にし，k を 5(全ノード数の 2.5%) に，次数制約を 5 に設定した．

- セッションの継続時間は 180 秒である．
- 各ノードはセッション中 1 回のみ参加する．
- 最初の 30 秒間で約 60 ノードがセッションに参加する．
- 時刻 30 秒〜140 秒間では，ノードの参加と共に離脱も発生する．時刻 60 秒と 120 秒で 2 つの収集フェーズが実行される．
- 時刻 140 秒以降は，新たなノードの参加は起こらず，約 40 ノードの離脱が生じる．
- 結果として各定常フェーズで起こる離脱数は k を大きく上回る．

図 6 と図 7 にノード数の時間変化を参加/離脱回数と共に示す．なお，このシナリオにおいて離脱回数が k を大きく上回っている場合でも孤立した部分木は生成されなかった．

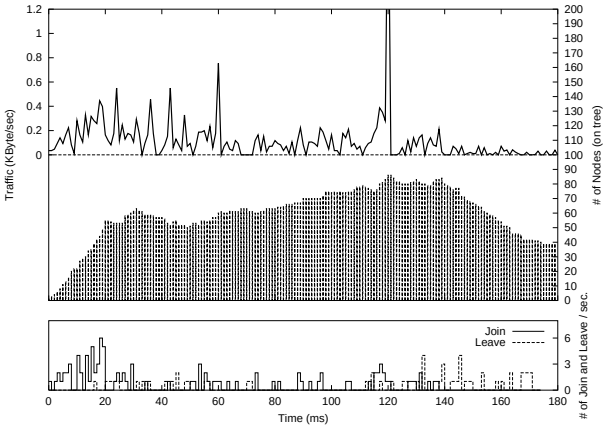

図 7: 制御トラフィック

5.2 比較対象：CT アルゴリズム

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．これは最小遅延のリンクを初期リンクとし，遅延の小さいリンクをグリーディに加えていく集中型の静的アルゴリズムである．

CT アルゴリズムは最適値に近い直径を持つスパニングツリーを構築するため，提案プロトコルでの木の直径の最適性のベンチマークとして利用した．また，CT アルゴリズムにおける参加及び離脱を以下のように実装した．(i) 新たな参加ノードには木全体の情報を利用し直径を最小にする接続先ノードを選択する．(ii) ノードの離脱に対しに直径が最小になるように木全体を再構築する．

5.3 実験結果

1) 直径：秒単位で測定した直径を図 6 に示す．直径の平均値が CT アルゴリズムの 1.09 倍，最悪時でも 1.25 倍であり，妥当な直径が得られている．また，直径の時間変動も CT アルゴリズムのものと同じ振舞をしており，提案プロトコルは分散型でありながらも最適値に近い直径値を維持しているといえる．

2) 修復コスト：削除あるいは追加されたオーバーレイリンク数の総和を修復コストと定義する．CT アルゴリズムは木全体が再構築されるため修復コストは大きくなる．

	提案	CT
総修復コスト	434	7283
適用された修復手続き数	149	149
平均修復コスト	2.9	48.9

表 1: 修復コストの比較

表 1 により，提案プロトコルがコストの面で優れていることが分かる．

3) トラフィック：木の制御トラフィック量の測定結果を図 7 に示す．これにより提案プロトコルは収集フェーズで最大トラフィックの 1KByte/sec(8kpbs) を生じている (時刻６０秒と１２０秒でのピーク値) が，これは十分に小さいといえる．

4) 手続きの所要時間：最後に参加/離脱手続きが要する時間を測定し，その分布を図 8 に示す．これにより全参加/離脱手続きが 0.8 秒以内に終了していることが分る．さらに，参加，離脱手続きの所要時間の期待値はそれぞれ 0.47 と 0.49 であり，両手続きが十分高速に行なわれているといえる．

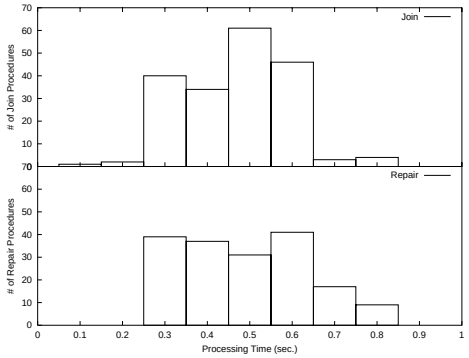

図 8: 参加/離脱に要する時間の分布

6 まとめと今後の課題

本稿では次数制約を持つオーバレイマルチキャスト遅延最小木の k ノードの同時又は連続した参加/離脱の発生 (ある期間内に) に対し，分散的に修復を行なうプロトコルを提案した．

提案プロトコル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] S. Shi, J. Turner and M. Waldvogel, “Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks,” *Proc. of Network and Operating System Support for Digital Audio and Video (NOSSDAV'01)*, pp. 83–91, 2001.

[2] R. L. Graham and P. Hell, “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] E. M. Royer and C. E. Perkins, “Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing,” *IETF Internet Draft*, draft-ietf-manet-maodv-00.txt, 2000.

[4] S. Paul, K. K. Sabnani, J. C.-H. Lin and S. Bhattacharyya, “Reliable Multicast Transport Protocol (RMTP),” *IEEE Journal of Selected Areas in Communications*, Vol. 15, No. 3, pp. 407–421, 1997.

[5] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, “ALMI: An Application Level Multicast Infrastructure,” *Proc. of 3rd USENIX Symp. on Internet Technologies and Systems (USITS)*, pp. 49–60, 2001.

[6] P. Francis, “Yoid: Extending the Internet Multicast Architecture,” *Unrefereed Report*, 2002. *http://www.isi.edu/div7/yoid/*

[7] P. Huang and J. Heidemann, “Minimizing Routing State for Light-Weight Network Simulation,” *Proc. of Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2001.