

## ユーザプリファレンスに基づく転送制御を行う アプリケーションレベルマルチキャストの一方

山口弘純 中村嘉隆 廣森聡仁 安本慶一<sup>†</sup> 東野輝夫 谷口健一

大阪大学 大学院基礎工学研究科 情報数理系専攻 <sup>†</sup> 滋賀大学 経済学部 情報管理学科

{h-yamagu, y-nakamr, hiromori, higashino, taniguchi}@ics.es.osaka-u.ac.jp

<sup>†</sup> yasumoto@biwako.shiga-u.ac.jp

本稿では、エンドホスト間でオーディオ及びビデオを長時間交換するグループアプリケーション向けのアプリケーションレベルマルチキャストプロトコル Emma を提案する。Emma では、エンド間ユニキャストからなるオーバレイネットワーク上に、オーディオ向けには遅延をメトリックとしたスパニングツリーをあらかじめ構築し、ビデオ向けには各エンドホストを根とする配送木を帯域とユーザプリファレンスをメトリックとしオンデマンドの継ぎ木により構築する。このもて、エンドホストは帯域制約を満たしかつユーザプリファレンスになるべく満足されるよう、オーバレイリンク上のビデオ転送を動的に制御する。Emma は完全な分散制御プロトコルとして実現される。簡単なシミュレーション実験により転送制御のオーバヘッドを測定している。

## A Protocol for Application Layer Multicast Based on Users' Preference

Hirozumi Yamaguchi, Yoshitaka Nakamura, Akihito Hiromori, Keiichi Yasumoto<sup>†</sup>,  
Teruo Higashino and Kenichi Taniguchi

Graduate School of Engineering Science, Osaka University <sup>†</sup> Faculty of Economics, Shiga University

In this paper, we propose a new application level multicast protocol called *Emma*, designed for audio/video multi-party applications. In *Emma*, a shared, spanning tree for audio transfer is constructed based on measured delay, while a source-based tree rooted at each end-host for video transfer is constructed based on bandwidth and user preference, on an overlay network where the end-hosts have direct (unicast) connections between them. Video packet forwarding is dynamically controlled in a distributed manner, so that user preference can be satisfied as much as possible. Simple experiments have been carried out to evaluate the overhead of the protocol operation.

### 1 はじめに

今後のインターネット端末や携帯端末の普及に伴い、ネットワークを介した比較的小規模のグループによるファイル交換やビデオ会議など、複数ユーザ間でグループ通信を行う需要が増加すると考えられる。しかし、そのようなグループコミュニケーションが多数確立されるようになると、各ユーザからのデータパケットを同グループの他のユーザに転送するコネクション処理を、限られた数のサーバが多数のグループに対し行うことは、サーバ側の帯域制限及びコネクション数制限などにより現実的ではない。これに対し IP マルチキャストなどを利用することも考えられるが、マルチキャストインフラストラクチャの普及、グローバルマルチキャストアドレスの割当てなどの諸問題により、そのようなグループコミュニケーションへの適用は現状では容易ではない。

近年、そのようなグループ間通信をアプリケーションレベルで実現するための方法に注目が集まっており、エンドホスト間で効率良くデータ転送を行うアプリケー

ションレベルマルチキャストの研究も多くなされている [3, 4, 5, 6, 7]。アプリケーションレベルマルチキャストはエンド間ユニキャストを仮想リンクとみなしたオーバレイネットワーク (図 1) 上でのマルチキャストであり、現在のインフラストラクチャでの実現が容易であること、ユニキャストトランスポートプロトコルの制御機能 (信頼性, セキュリティ, フロー制御, レート制御など) が容易に利用可能であるなど利点が多い。しかしその一方で、エンドホスト付近での帯域制約や実リンク上でのパケット重複など、オーバレイネットワーク上でのオーバヘッドを考慮した性能のよいプロトコルが要求される。

アプリケーションレベルマルチキャストに関する従来の研究は、それらのオーバヘッドをネイティブマルチキャストと比較しても十分許容できる程度に抑え、かつエンドホストの離脱などに伴うオーバレイネットワークの流動性への適応を目標としたものがほとんどである。しかし、我々がターゲットとしている電子ビデオ会議などのコミュニケーションシステムでは、(1) 例えば数十台のエンドホストがそれぞれ同時並行かつ継続的

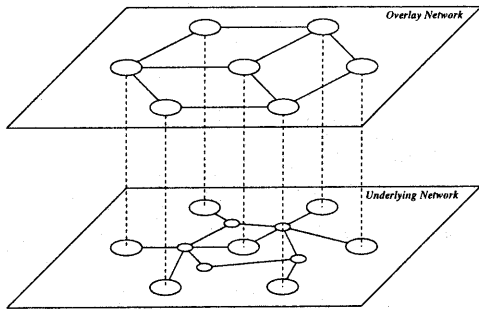


図 1: オーバレイネットワーク

に他のエンドホストにビデオデータを送出し、(2) 各エンドホストが扱える（受信または転送できる）ビデオ数は、処理能力及びエンドホスト付近の帯域を考慮した場合には一定の制限があり、(3) 各エンドホストが受信したいビデオには偏りがある（プリファレンスがある）、といった特徴及び制約がある。したがって、オーバレイマルチキャストの基本要件を満足しつつ、エンドホストのプリファレンス及び帯域を考慮したマルチキャスト配信木の構築及び維持が分散制御で行えることが望ましいといえる。唯一、文献 [3] では文献 [4] のプロトコル Narada を会議アプリケーション向けに設計しているが、この方法においては、各ノードから送信されるビデオは同時には一つであると仮定しており、各ノードを根とする帯域優先の経路木が単純に（ソースからのフラッディングに基づき）構成されるのみである。したがって、上述のアプリケーションに適用した場合も、エンドホストのプリファレンスをを十分満足させることは容易ではない。

本稿では、エンドホスト間でオーディオ及びビデオを実時間交換するようなコミュニケーションシステム向けの、アプリケーションレベルマルチキャストプロトコル Emma (End-user Multicast for Multi-party Applications) を提案し、簡単なシミュレーション実験による性能評価を行ったため、それを報告する。

## 2 プロトコルの設計方針

我々は、オーバレイネットワークの特性、及び目的アプリケーションの特性から以下の方針で Emma を設計した。以下、オーバレイネットワークを構成するエンドホストを単にノードとよぶ。

[オーディオ配信制御機能とノードの相互接続性維持機能]

音声やサムネイル画像、共有ホワイトボードへの書き込みデータなど、全体として利用する帯域が（非継続的、あるいは排他的な特性により）比較的小さいが、遅延には敏感なメディア（以下、このタイプのメディア

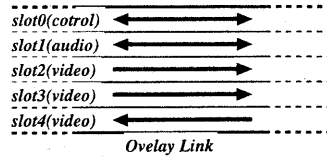


図 2: オーバレイリンクのスロット分割

は単にオーディオとよぶ) 向けに、遅延をメトリックとした共有スパニングツリーを、ノードがアプリケーション（セッション）に参加する際に継ぎ木方式で構築する。その際、木上のノード間の最大遅延はセッションの許容遅延以下となるよう接続先を選択する。また、この木を利用し、ノード間の相互接続性を保証する。各ノードはこの木を元に、一ノードの離脱や障害に対してもノード間の最大遅延とノード間の相互接続性を保証できるような代替経路を計算しておく。

[ビデオ配信制御機能]

電子会議における各参加者の映像やマルチサイト会議の各会場の中継画像など、遅延と帯域を要するメディア [3]（本稿ではこのタイプのメディアは単にビデオとよぶ）向けに、遅延と空き帯域をメトリックとし、各ノードを根とするソーススペースの配信木を、ノードの受信要求をトリガとしたオンデマンドの継ぎ木方式で構築する。オンデマンドとすることにより、状況に応じた適当な経路を選択できる可能性がある反面、受信要求のフラッディングが必要となるなど経路選択時にかかるオーバヘッドも大きい。Emma では、各ソースノードが遅延をメトリックにしたフラッディングを行い、各隣接ノードごとソースノードへの遅延を記録しておき、受信要求メッセージの転送先はその遅延情報と空き帯域をもとに決定する。

なお、ビデオ配信においてはそれぞれがある程度の帯域を恒常的に消費するため、オーバレイリンク上に空き帯域が十分にない場合も多い。その場合はアプリケーション（とそのユーザ）依存のメトリックであるプリファレンス値に基づき、既存のメディアの配信停止によるプリファレンス値損失をなるべく小さくしながら空き帯域を確保できる可能性のあるリンクを選択する。

[ノード負荷削減]

アプリケーションレベルマルチキャストでは制御メッセージ数、接続するオーバレイリンク数（コネクション数）、転送ビデオ数などがノード負荷に影響を与えると考えられる。Emma では、一定の周期時間でノードを緩やかに同期させ、周期時間内に一オーバレイリンク上に送信される制御メッセージを集約することで制御メッセージ数の削減を図ると共に、周期時間の調整による制御トラフィックの調整を可能とする。また、各ノードのコネクション管理を容易にするため、任意の

2ノード間のオーバーレイリンクは高々1本の双方向ユニキャストチャンネルのみであるとし、各オーバーレイリンクをスロットと呼ばれる仮想的な分割単位で管理する。一つのスロットの利用帯域はあらかじめ固定（例えば64kbps）であるとし、スロット0は制御メッセージ用双方向チャンネル、スロット1はオーディオ用双方向チャンネル、スロット2以降はビデオ用単方向チャンネルであるとする（図2）。これにより、実ネットワークでのコネクション数を減らし、Emmaが利用可能な帯域の制限、空き帯域の予測などが容易になる。なお、ノード*i*は(*i, j*)の帯域及び遅延(RTT)の測定がpathchar[2]やpingなどにより可能であるとする。また、一ノードあたりのオーバーレイリンクは高々3程度とし、セッション参加時のRTTブループ及び接続先ノードとのネゴシエーションにより決定する。

### 3 プロトコル記述

Emmaでは、制御メッセージをCTRL, AUDIO, VIDEOの3タイプに分類し、さらに各タイプをいくつかのサブタイプに分類する。以下、各メッセージを、“タイプ/サブタイプ”で分類する。なお、メッセージタイプの一覧を表1に、利用する表記の一覧を表2に示す。

#### 3.1 オーバレイリンク確立

ノードはセッション参加時に、ロビーサーバの役割を持つ適当なサーバ（WWWサーバなど）に自身のIPアドレスを登録し、そのサーバにすでに登録しているノード（すでにセッションに参加しているノード）のIPアドレスリストを入手する。それらのノードへの遅延や帯域を計測し、遅延が小さい数個のノードに対し、リンクスロット数のネゴシエーションパラメータを含むCTRL/JoinREQメッセージ（トランスポートレベルのコネクション設立要求）を送出し、ネゴシエーション結果CTRL/JoinREPに基づくスロット数のオーバーレイリンクを構築する。

なおCTRL/JoinREPを送出するノードは各ノードに共通の周期シーケンス番号、周期時間、現在の周期時間終了までの残り時間（オフセット）を含ませる。受け取ったノードは以降その周期時間に従い周期シーケンス番号を増加させる。これにより、各ノードがおおよそ同じタイミングで周期シーケンス番号を同じ値に更新する。これは、タイプが“VIDEO”である制御メッセージ（後述）を集約するために利用する。

#### 3.2 オーディオ転送制御とノード間相互接続性維持

オーディオ用の共有木は、ノード間の最大遅延がセッションの許容遅延以下である張る木とする。これはCTRL/JoinREPを受け取りオーバーレイリンクを構築

表1: ノード *u* から *v* への制御メッセージ

Type	Subtype	Layer	Semantics*
CTRL	JoinREQ	Underlying	オーバーレイリンクネゴシエーションパラメータ
	JoinREP	Underlying	ネゴシエーション結果、周期シーケンス番号、周期時間、オフセット
	Leave	Overlay	セッション離脱
AUDIO	Advertise	Overlay	各ノードからのDA公告
	JoinREQ	Overlay	オーディオ受信要求
	JoinREP	Overlay	要求結果
VIDEO	Advertise	Overlay	各ノードからのDV公告
	JoinREQ	Overlay	ビデオ受信要求、要求プリファレンス、プリファレンス値損失計算パラメータ
	JoinREP	Overlay	要求結果
		Overlay	受信継続通知、プリファレンス値
	Keep	Overlay	受信停止要求
	Leave	Overlay	受信停止要求

\* 受信ノードアドレスはデフォルトで含まれるとする

表2: 表記

表記	意味
$(i, j)$	ノード <i>i, j</i> 間のオーバーレイリンク
$V_s$	ノード <i>s</i> のビデオ
$pref(s, v)$	$V_s$ に対するノード <i>v</i> のプリファレンス値
$d(i, j)$	オーバーレイリンク ( <i>i, j</i> ) の (計測などによる) 遅延
$DA(v)$	オーディオ用共有木のノード <i>v</i> での最大遅延
$DV_s(v)$	ノード <i>s</i> からノード <i>v</i> までの最小遅延
$T_s$	$V_s$ の配送木
$T_s(v)$	$V_s$ の配送木の、 <i>v</i> を根とする部分木 ( $T_s(s) = T_s$ )
$\uparrow_s(v)$	$T_s$ について、ノード <i>v</i> の親ノード
$\downarrow_s(v)$	$V_s$ の配送木について、ノード <i>v</i> の子ノード集合
$\downarrow_s(v)^{[Q]}$	$V_s$ の要求木 ( $V_s$ の VIDEO/JoinREQ により形成される木) について、ノード <i>v</i> の子ノード集合

したノードが、すでに存在する木に継ぎ木する形で以下のように構築できる。

セッションに参加したノード *v* はオーバーレイリンクを構築したノード *u* から、現在の共有木の *u* までのノード間最大遅延である  $DA(u)$  を AUDIO/Advertise メッセージとして受け取り、 $DA(u) + d(u, v)$  なるべく小さい *u* に対し AUDIO/JoinREQ メッセージを発行し、リンク (*u, v*) のスロット1を既存のオーディオ共有木に接続する。*v* は  $DA(v) = DA(u) + d(u, v)$  とする。

また、一ノードの離脱に対してもこの張る木を維持できるように、接続したノード *u* を“親”とした場合の、*u* の親ノード (*u'* とする) に対し CTRL/JoinREQ によりオーバーレイリンクを構築しておき、これを代替経路とする。このように仮の親子関係を定める (この親子関係はデータ転送には無関係) ことで、1ノードの離脱や障害に対してもノード間最大遅延及び相互接続性を保証できる (初めにセッションに参加したノードは親ノードをもたないため、2番目に参加したノードを親ノードとみなす)。ノード *u* の離脱や障害を検知したノード *v* は、*u'* に AUDIO/JoinREQ を発行し、張る

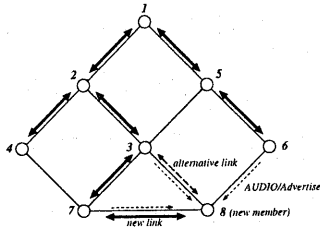


図 3: オーディオ用共有木

木を維持する。

図 3 に例を示す。ノードの数字はセッションに参加した順を示す。ノード 8 がノード 7 とのリンクを共有木に接続したとすれば、ノード 8 はノード 7 の親であるノード 3 との間にオーバーレイリンクを構築してこれを代替経路をみなし、ノード 7 の障害時にはノード 3 に接続を変更する。

### 3.3 ビデオ転送制御

[VIDEO/Advertise メッセージによる遅延時間計算] ビデオの配信木はノードの受信要求に応じてオンデマンドの継ぎ木方式で構築される。Emma では配送木間の競合をできるだけ避けるために、受信要求メッセージは、セッションの許容遅延を満たしながらなるべく空き帯域のあるリンク上を（途中のノードの受信要求と集約されながら）ソースノードに向けて転送される。この際、ソースノードへの経路情報を全く保持しない状況では各ノードからの受信要求のフラディングに頼ることになり現実的でないことから、ビデオを送信する各ノード  $s$  は、経路選択のメトリックとなる遅延計算のために、 $V_s$  についての VIDEO/Advertise メッセージを適当な周期時間間隔で各隣接ノードに送信する。  $u$  から  $V_s$  についてのメッセージを受信したノード  $v$  は、  $u$  と  $V_s$  ごと、メッセージに含まれる  $s$  から  $u$  までの最小遅延  $DV_s(u)$  を記録する。また、もし  $DV_s(v) > DV_s(u) + d(u, v)$  であれば  $DV_s(v) = DV_s(u) + d(u, v)$  とし、隣接ノードに送信する。これにより、受信要求メッセージを転送する際に遅延をメトリックとしたリンク選択が可能となる。

[VIDEO/Keep メッセージによるプリファレンス値の集約] Emma ではノード  $v$  のビデオ  $V_s$  に対するプリファレンスの程度を整数値の大きさと表現するとし、以下  $pref(s, v)$  (非負整数値) で表す。プリファレンスは受信要求時、リンクに空き帯域がない場合に優先して配送するビデオを決定するためのメトリックである。

各ノード  $v$  はその決定を受信要求時にすみやかに行えるよう、受信中のメディア  $V_s$  に対し、  $pref(s, v)$  を

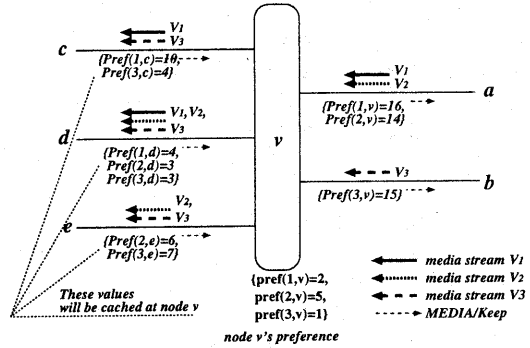


図 4: VIDEO/Keep によるプリファレンス値集約

含む VIDEO/Keep メッセージを、適当な時間周期間隔ごと上流ノード  $u = \uparrow_s(v)$  に送出する。また  $V_s$  を子ノードに転送している場合はそれらのノードからの VIDEO/Keep メッセージを受信し、  $V_s$  の部分配送木  $T_s(v)$  上の  $V_s$  に対するプリファレンス値総和（以下  $Pref(s, v)$  で表す）を計算し、それを含むメッセージを次の周期時間に上流ノード  $u$  へ転送する。なお、その計算は、  $Pref(s, v)$  が

$$Pref(s, v) = pref(s, v) + \sum_{w \in \downarrow_s(v)} Pref(s, w)$$

で再帰的に定義できるため、各ノード  $w$  は送出する VIDEO/Keep メッセージに計算した  $Pref(s, w)$  を含ませることで実現できる。ノード  $v$  は子ノード  $w$  の  $Pref(s, w)$  を保持しておくとする。なお、ノード  $v$  は、ある子ノード  $w$  から VIDEO/Keep メッセージを受信した周期に他の子ノード  $w' \in \downarrow_s(v)$  からの VIDEO/Keep メッセージが到着しなければ、保持していた集約プリファレンス値  $Pref(s, w')$  を用いて  $Pref(s, v)$  を計算する。

$v$  が同じ親ノードから複数のビデオを受信している場合は、それらに関する VIDEO/Keep メッセージは 1 つのメッセージにまとめて送出する。これにより、1 オーバレイリンク上に一周期時間あたりに送出されるメッセージを一つにできる。図 4 に VIDEO/Keep メッセージによるプリファレンス値集約の例を示す。ノード  $v$  はノード  $a$  から  $V_1, V_2$  を、  $b$  から  $V_3$  を受信し、  $c$  には  $V_1, V_3$  を、  $d$  には  $V_1, V_2, V_3$  を、  $e$  には  $V_2, V_3$  をそれぞれ転送している。  $v$  はそれらの子ノードから受け取る VIDEO/Keep に含まれる、集約されたプリファレンス値に自身のプリファレンス値を加算し、  $Pref(1, v)$ 、  $Pref(2, v)$  及び  $Pref(3, v)$  を計算する（この図の場合、それぞれ 16, 14, 15）。  $v$  は  $a$  への VIDEO/Keep には  $Pref(1, v)$ 、  $Pref(2, v)$  を、  $b$  への VIDEO/Keep には  $Pref(3, v)$  を含めて送信する。

定められた周期時間の間 VIDEO/Keep を受信しない子ノード  $w''$  に対しては、 $v$  は保持している  $Pref(s, w'')$  の値をクリアし、 $w''$  への配送を停止する。

**[VIDEO/JoinREQ メッセージによる受信要求]** ノード  $v$  は  $V_s$  の受信要求を行う場合、VIDEO/Advertise メッセージにより記録されている  $s$  への遅延時間がセッションの許容遅延以下でありかつオーバーレイリンクに空き帯域 (スロット) がある隣接ノード  $u$  を一つ選んで適当な周期時間ごと VIDEO/JoinREQ メッセージを送信する。また、 $V_s$  についての VIDEO/JoinREQ を一つ以上の隣接ノード  $w$  から受け取った場合、それらのメッセージに含まれる要求をまとめて送信する。

ただし、いずれのリンクにも空き帯域がない場合、各隣接ノード  $u$  と、 $(u, v)$  上を  $v$  へ配送されている各メディア  $V_i$  について、VIDEO/JoinREQ が転送されてきた各リンク上 (これらのリンクで構成される木を  $V_s$  の要求木とよぶ) に少なくとも 1 つの空きスロットを確保するために、 $(u, v)$  上での  $V_i$  の配送停止を含む既存のビデオの配送停止方法のうち、最もプリファレンス値損失が小さいような配送停止方法の損失 (これを  $loss_s(i, v)$  で表す) を計算する。  $loss_s(i, v)$  は

$$\begin{aligned} loss_s(i, v) &= pref(i, v) \\ &+ \sum_{w \in \downarrow_i(v) - \downarrow_s(v) \cap Q} Pref(i, w) \\ &+ \sum_{w \in \downarrow_s(v) \cap Q \cap \downarrow_i(v)} loss_s(i, w) \\ &+ \sum_{w \in \downarrow_s(v) \cap Q - \downarrow_i(v)} \left\{ \min_{V_j \in V(v, w)} loss_s(j, w) \right\} \end{aligned}$$

で定義できる。なお、 $V(u, v)$  で、 $(u, v)$  上を  $u$  から  $v$  へ配送されているメディアの集合を表す。 $(u, v)$  上のメディア  $V_i$  の配送を停止することで要求木上に空きスロットを確保するためには、(a) ノード  $v$  の  $V_s$  に対するプリファレンス値損失、(b)  $V_s$  を要求していないにも関わらず  $v$  が転送を行わなくなることで  $V_i$  を受信できなくなる  $v$  の全下流ノードの、 $V_i$  に対するプリファレンス値損失総和、(c)  $V_s$  を要求している  $v$  の全下流ノードのうち、 $V_i$  も受信しているノードのプリファレンス値損失総和、(d)  $V_s$  を要求している  $v$  の全下流ノードのうち、 $V_i$  は受信していないノードの、 $V_i$  以外のメディア受信を停止する場合のプリファレンス値損失総和の最小、の総和が損失となる。 $loss_s(i, v)$  の第 1 項から第 4 項は上述の (a) から (d) にそれぞれ対応する。 $loss_s(i, v)$  の式より、ノード  $w$  が  $v$  に送信する VIDEO/JoinREQ に  $Pref(s, w)$  及び  $loss_s(j, w)$  ( $\forall V_j \in V(v, w)$ ) が含まれていれば、ノード  $v$  は  $loss_s(i, v)$  を計算することができる。

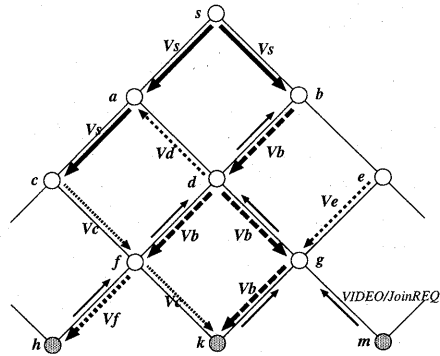


図 5: VIDEO/JoinREQ の転送例

ここで、ノード  $v$  は (一定の遅延を満足する) 隣接ノードそれぞれに対し上記の計算を行い、 $loss_s(i, v)$  が最小であるような  $V_i$  の親ノードを VIDEO/JoinREQ の転送先に決定する。

図 5 はノード  $h, k, m$  が送出した  $V_s$  に対する VIDEO/JoinREQ メッセージが転送される様子を示している。簡単のため、一オーバーレイ用のビデオ用スロットは 1 であるとし、遅延は考慮していない。例えばノード  $k$  は VIDEO/JoinREQ の送出先としてノード  $f$ 、ノード  $g$  があるが、この場合  $k$  は  $loss_s(c, k) = pref(c, k)$ 、 $loss_s(b, k) = pref(b, k)$  であるため単純に自身のプリファレンス値で決定できる。今、 $pref(b, k) < pref(c, k)$  であるとすれば  $g$  が送出先として選択される。同様にして集約された VIDEO/JoinREQ が図のようにノード  $d$  に木上に集約されたとする。ノード  $d$  は  $V_s$  の  $d$  以下での配送停止を含みかつリンク  $b-d, d-f, d-g, f-h, g-k, g-m$  からなる  $V_s$  の要求木上に空きスロットを作るための最小損失プリファレンス値を以下で計算する。

$$loss_s(b, d) = pref(b, d) + loss_s(b, f) + loss_s(b, g)$$

ただし、

$$loss_s(b, f) = pref(b, f) + loss_s(f, h)$$

$$loss_s(b, g) = pref(b, g) + loss_s(b, k) + loss_s(\epsilon, m)$$

であり、 $loss_s(\epsilon, m)$  は空きスロットがあるリンクのプリファレンス値損失 (すなわち 0) を便宜上表している。

**[VIDEO/JoinREP メッセージによる配送ビデオ変更]** VIDEO/JoinREQ を受け取ったノード  $v$  上に  $V_s$  がすでに配送されている ( $v = s$  の場合も含む) 場合、 $v$  は VIDEO/JoinREQ を受け取ったノード  $w \in \downarrow_s(v) \cap Q$  に配送しているビデオ  $V_i$  のうち  $loss_s(i, w)$  が最小で

あるものについて,

$$\text{loss}_s(i, w) < \text{Pref}(s, w)$$

なら,  $V_i$  の  $V_s$  への変更命令を, そうでなければ変更不可通知を含む VIDEO/JoinREP メッセージを  $w$  に向けて送信する. VIDEO/JoinREP を受け取ったノードは  $V_i$  の代わりに  $V_s$  を配信し要求木上を下流に転送する. ただし,  $\Psi_i(w) = \emptyset$  であるノード ( $V_i$  の配送木の葉ノード)  $w$  は VIDEO/JoinREP に含まれる  $V_i$  から  $V_s$  への変更命令を,  $V_j$  から  $V_s$  への変更命令に置き換えて転送する (ただし  $V_j$  は  $\text{loss}_s(j, w')$  ( $w'$  は  $w$  の子ノード) が最小であるビデオ). これにより, 要求木上に  $V_s$  の送信が行われ, かつプリファレンス損失を選択した要求木において) 最小とできる.

一方, 変更不可命令を含む VIDEO/JoinREP メッセージは各ノードの損失計算値をクリアしながら転送される.

## 4 シミュレーション実験

ネットワークシミュレータ ns-2 [1] 上に Emma を実装し, 簡単な性能評価を行った.

ノード数 30 のうちエンドホストが 10 であるランダムなトポロジを持つネットワーク上で, 各エンドホストを順次セッションに参加させ, ネットワークが定常状態に到達した後, 各エンドホストが定期的にランダムに決定したプリファレンス値に基づいてメディア受信を要求する状況をつくり出した. このもとで, 定常状態移行後の制御メッセージによるトラフィック量と, VIDEO/JoinREQ 送信からメディア変更までの時間 (要求応答時間) を測定した. なお, 1 周期時間は 0.5 秒, 制御メッセージサイズは約 200B とした. シミュレーション実験は合計 10 回行った. その結果, 定常時の制御トラフィックはオーバーレイリンクあたりおおよそ 0.2kbps, 要求応答時間は最悪時でも 1.6 秒であった. なお, 一般にアプリケーションマルチキャストは, 異なるオーバーレイリンクが同じ実リンクを利用する可能性もあるため, ネイティブマルチキャストと比較しネットワーク利用効率が低下する. したがって, それが十分妥当なレベルであることを示すための指標として, 実リンク上の重複率 (リンクストレス) が文献 [4] であげられている. 本実験におけるリンクストレスは高々 3 であったため, 実リンクでのトラフィックは高々 0.6kbps であり, 十分許容できる値であるといえる.

今後はノード負荷及び実遅延の検証をより一般のネットワーク上 (エンドホスト数は数十程度) で行うことで Emma がユニキャスト (リンクストレス最大, ノード負荷最大, 遅延最小) と比較しどの程度の効率向上を図れているかを調べる. またノードの離脱に対しどの程度の適応性を実現できるかを調べ, よりエンドホストの流動性に適したプロトコルとする予定である. さ

らに, VIDEO/JoinREQ の発行順にビデオ配送を決定し, 動的には変更を行わない単純な方法と比較しどの程度のプリファレンス向上が達成できるかも測定する予定である.

## 5 おわりに

本稿では, 各エンドホストが他のエンドホストにオーディオ及びビデオを配信するコミュニケーションシステム向けに, ユーザプリファレンスを考慮したアプリケーションレベルマルチキャスト Emma を提案した.

今後の課題は, より詳細なシミュレーション実験による性能評価, Emma に基づくグループコミュニケーションシステム向けのミドルウェアの開発, そのもとでの会議アプリケーションの実装, Emma のワイヤレスアドホックマルチキャストへの応用などがあげられる.

## 参考文献

- [1] Berkeley MASH Research Group, University of California, "The Network Simulator ns-2," <http://www-mash.ce.berkeley.edu/ns/>, 2000.
- [2] V. Jacobson, "pathchar," 1997. Tools are available at Lawrence Berkeley National Laboratory FTP site: <ftp://ftp.ee.lbl.gov/pathchar/>.
- [3] Y.-H. Chu, S. G. Rao, S. Seshan and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," *Proc. of ACM SIGCOMM*, 2001.
- [4] Y.-H. Chu, S. G. Rao and H. Zhang, "A Case for End System Multicast," *Proc. of ACM SIGMETRICS*, 2000.
- [5] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek and R. Morris, "The Case for Resilient Overlay Networks," *Proc. of 8th Annual Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.
- [6] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," *Proc. of 3rd Usenix Symp. on Internet Technologies & Systems*, 2001.
- [7] R. Cohen and G. Kaempfer, "A Unicast-based Approach for Streaming Multicast," *Proc. of IEEE INFOCOM 2001*, 2001.
- [8] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network," *Proc. of ACM SIGCOMM 2001*, 2001.
- [9] F. Baccelli, D. Kofman and J.L. Rougier, "Self Organizing Hierarchical Multicast Trees And Their Optimization," *Proc. of IEEE INFOCOM 2001*, 2001.