# 多様な品質要求に対するトランスコードに基づくP2Pビデオ配信手法とその実環境での評価

柴田直樹，安本慶一†，森 将豪

滋賀大学情報管理学科 †奈良先端科学技術大学院大学情報科学研究科

**概要** 我々の研究グループでは，異なる品質で同じビデオの配信を要求する多数のユーザへのビデオ配信をP2Pネットワークにおける各ピアでトランスコードと中継を行うことで実現する方式MTcastを提案してきた．本稿では，PlanetLab上で動作するMTcastのプロトタイプを作成し，評価実験を行った結果を報告する．PlanetLab上の世界各国の20のノードを用いて，ビデオ配信時のスタートアップ遅延とノード離脱時の振る舞いについて調査した結果，実用上十分な性能が達成されることを確かめた．

# Transcode-based P2P video delivery method for diverse quality requests and its evaluation in real environment

Naoki Shibata, Keiichi Yasumoto† and Masaaki Mori

Shiga University †Nara Institute of Science and Technology

**Abstract** We have already proposed a P2P video delivery method called MTcast for simultaneously delivering video to users with different quality requirements. MTcast makes each user node transcode and forward video to other user nodes. In this paper, we report the results of performance evaluation of our method in the real internet environment. We have developed a prototype system and conducted experiments on 20 PlanetLab nodes. We evaluated startup delay and behavior when a user node abruptly leaves, and confirmed that MCast achieves practical performance in a real environment.

## 1 はじめに

計算処理能力，画面サイズ，ネットワークの利用可能帯域幅など環境の異なる多数のユーザ端末に対し，ビデオ配信を効率よく行う方法が希求されている．同一のビデオを異なる品質で複数ユーザに同時配信する既存の方法として，以下の手法がある．マルチバージョン法 [1] は，異なるビットレートの複数のビデオバージョンを用意し，ユーザからの要求に対し，制約を満たす最も品質の良いものを選んで配信する．オンライントランスコード法 [2] は，ビデオを中間ノード上で，要求された品質のビデオに変換して配信する．階層化マルチキャスト法 [3, 4] は，ビデオを階層符号化 [5] し独立したマルチキャストストリームとして配信する．ユーザは任意の個数の層を受信することでビデオを復号する．この方式は，ユーザ満足度を改善するためにはより多くの層数が必要となり，多くの層からのビデオ復号には大きな処理能力が必要になり，もし層の数が十分でなければ，要求品質と配信品質の差が大きくなる．オンライントランスコード法は，ユーザの要求に合わせた品質への変換が可能だが，トランスコードを行うために中間ノードで大きな計算パワーが必要となる．

また，P2Pネットワークを対象としたビデオ配信方式は多数研究されている．代表的なものにOMNI[6]やCoopNet[7]がある．OMNI (Overlay Multicast Network Infrastructure) では，各ユーザノードはサービスユーザであると同時にサービスプロバイダとしての役目を持ち，マルチキャスト木はビデオ配信サービスが木を通して全てのユーザノードに提供されるように構成される．また，ユーザノード分布やネットワーク条件の変化にも適応できる．CoopNetは，ビデオサーバの負荷がサーバの限界を超えたときのクライアント・サーバベースの配信方法を工夫している．この場合，ユーザノードはストリームデータをキャッシュし，それらを複数の異なる配送木を使ってユーザノードに配信する．これらは，ネットワーク条件の動的な変化に適応できるが，異なる品質要求を持つユーザに対するビデオ配信を扱っていない．文献 [9] では，P2Pネットワークにおいて異なった品質要求に対するビデオ配信をトランスコードにより行う上で，エンコード・デコードの途中過程における結果の一部を共有することで，トランスコードに必要な計算資源を削減する方法を提案している．

我々の研究グループでは，異なる品質で同じビデオの配信を要求する多数のユーザに対して，末端のユーザを除くすべてのユーザノードにトランスコードを行わせ中継させることで，効率よくビデオを同時配信する方式MTcast[8]を提案している．MTcastでは，各ユーザ (ビデオ受信者) はビットレートを要求品質として指定し，ビデオの配信を要求する．これらに基づいて，根がビデオコンテンツの送信源となるトランスコード木と呼ばれる配送木を，変形完全$n$分木として構成する．より高い品質要求を持つユーザノードは木の根近くに，より低い品質要求を持つユーザノードは葉近くに配置する．トランスコード木の各ノード (コンテンツ受信者) は，受信したビデオストリームを実時間トランスコードし，下流のノード (より低い品質を希望する受信者) に転送することで，制約・要求が異なる複数のユーザへのビデオ配信を効率よく実現する．

本稿では，文献 [8] で提案した方式を実際にインターネットで動作させることを目標に，システムの設計・実装，およびPlanetLab[11] 上での実験を行った結果を報告する．シミュレーションおよびPlanetLab上の評価実験を通して，提案手法が高い耐故障性とスケーラビリティ，短い配送開始遅延，および階層型マルチキャスト方式よりも高いユーザ満足度を達成していることを確認した．

## 2 MTcastの概要と実環境での動作に向けた拡張

この章では，文献 [8] で提案したMTcastの対象とする環境と，各種定義について述べ，MTcastのアルゴリズムの概要について説明する．また，実環境で効率よく動作させるため，物理トポロジを考慮したトランスコード木の構成方法を新たに提案する．

## 2.1 対象とする環境

MTcastは，利用可能帯域，処理能力などが異なる複数のユーザノードに対し，ビデオコンテンツを同時配信するための配信方式であり，以下のような環境を想定している．

**ネットワーク環境：** 複数ドメインに跨る広域ネットワーク
**ユーザ端末：** デスクトップPC，ラップトップPC，PDA等，インターネットに接続する様々な端末
**通信インフラ：** ブロードバンド（専用線，ADSL，CATV，etc），無線回線（無線LAN，Bluetooth，WiMax，etc）等，様々な伝送速度の媒体によりインターネット接続
**ユーザ数：** 500〜100,000程度
**対象コンテンツ：** ビデオ（録画済，生放送の両方）

MTcastでは，同一のビデオコンテンツを一斉に配信するサービスを想定しており，各ユーザが希望した時間に好みの配信を受けるオンデマンド型のサービスは対象としない．ユーザは放送開始後いつでも，現在放送中のシーンからビデオの受信を開始できる．

## 2.2 諸定義

ビデオ配信サーバを $s$，ユーザ数を $N$，ユーザノードの集合を $U = \{u_1, ..., u_N\}$ と表記する．各 $u_i$ に対して利用可能な上り（送信）帯域幅（ノードが送信に使用できる帯域幅）と，下り（受信）帯域幅（ノードが受信に使用できる帯域幅）は既知であるとし，これらを $u_i.upper\_bw$ と $u_i.lower\_bw$ で表す．各ユーザノード $u_i$ のビデオ要求品質を $u_i.q$ と表記する．$u_i.q$ はビットレートを表し，適切な画像サイズとフレームレートはビットレートから計算できるとする．ノード $u_i$ が品質 $q$ で表されるビデオを同時にトランスコード可能な本数を $u_i.n_{trans}(q)$，ノード $u_i$ で行われる品質 $q$ のビデオの最大同時転送数を $u_i.n_{link}(q)$ で表す．$u_i.n_{trans}(q)$ と $u_i.n_{link}(q)$ は，$u_i$ と $u_i.upper\_bw$ とビデオ品質より計算できる．

MTcastでは，$s$ を根，$U$ の各要素を節または葉とするオーバレイマルチキャスト木を構築する．今後，このマルチキャスト木を**トランスコード木**と呼ぶ．また，トランスコード木の葉ノード以外のノードを**内部ノード**と呼ぶ．

## 2.3 トランスコード木の構造

トランスコード木の内部ノードはビデオストリームを子ノードへ送信する．各内部ノードが持つ子ノードの数（リンク次数）を，原則として定数値 $n$ とする．2.4節で説明するように，$n$ の値はユーザノードの利用可能な資源数に依存して決める．根ノードから送信されたストリームが葉ノードで再生されるまでの遅延時間およびトランスコード数を削減するため，トランスコード木を変形完全 $n$ 分木状（後述の理由により，トランスコード木の根ノードの次数は $n$ ではなく $k$ である）に構成する．トランスコード木では，各ノード $u_i$ と $u_i$ の任意の子ノード $u_j$ に対して，$u_i.q \geq u_j.q$ が成り立つようにする．すなわち，トランスコード木の根ノード（動画配信サーバ）から葉ノードへ向かう任意のパス上のノード群の要求品質は降順となるようにトランスコード木を構築する．

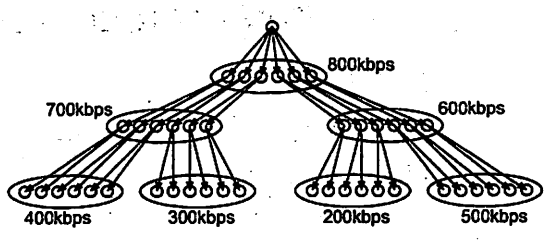


図 1: トランスコード木 $(n = 2, k = 6)$ による配信

ノードの故障・離脱に耐えビデオ配送開始遅延を短くするために，$U$ に属する $k$ 個のノードを束にして一つのグループとする．このグループのことを**レイヤ**と呼ぶ．$k$ はあらかじめ決められた定数である．同じレイヤ内のユーザノードには同一の品質を持つビデオを受信させることとする．この品質のことを**レイヤ品質**と呼ぶ．各レイヤに対してそのレイヤを管理する代表ノードが一つ選ばれる．トランスコード木上の全てのレイヤ間の親子関係は**レイヤ木**と呼ばれる．また，レイヤ木の最上位のレイヤを**根レイヤ**と呼ぶ．図1は $n = 2, k = 6$ のトランスコード木の例で，小さな円と大きな円はそれぞれノードとレイヤを表している．

## 2.4 トランスコード木の構築

MTcastでは，トランスコード木の計算をただ一つのノード $u_C$ で集中的に行う．$u_C$ の決め方は2.5節で説明する．$u_C$ にはビデオサーバ $s$ とビデオを要求しているユーザノード群 $U$ の情報を予め持たせる．トランスコード木構築アルゴリズムは以下の3つのステップから成る．

**ステップ1：** ユーザノードの集合 $U$ の各ノード $u$ に対して，ノード $u$ が一つまたはそれ以上のビデオのトランスコードを行うことができるか（下記不等式 (1)）と，$u$ が $(n+1)$ 本のビデオストリームを同時転送できる（不等式 (2)）か判定する．

$$u.n_{trans}(u.q) \geq 1 \quad (1)$$

$$u.n_{link}(u.q) \geq n + 1 \quad (2)$$

上記不等式の両方が成り立てば，$u$ を内部ノードの集合 $U_I$ に入れ，さもなくば葉ノードの集合 $U_L$ に入れる．但し，根ノード $s$ は $U_I$ に入れる．分割後に $|U_I| < |U|/n$ であれば，全てのユーザの要求品質を満足するにはネットワーク資源が不足している．この際には，不等式 (1) と (2) が成り立つように，$U_L$ 中のより大きな上り（送信）帯域を持つ $|U_L| - (n-1)/n|U|$ 個のノードの品質要求を減少させる．減少後それらのノードは $U_I$ に移される．上記の手続きにより $|U_I| > |U|/n$ が常に成り立つように調整する．

**ステップ2：** $U_I$ の要素を要求品質の降順にソートし，$k$ 個ずつ要素を束ねて一つの内部レイヤに割り当てる．各レイヤ毎に，最初と2番目のノードをレイヤの**代表ノード**，**副代表ノード**とする．各レイヤに属するノードの要求品質の平均値をレイヤ品質とする．葉ノードの集合 $U_L$ も同様にレイヤに割り当てる．

**ステップ3：** 内部レイヤをレイヤ品質の降順にソートし，それら内部レイヤの完全 $n$ 分木を構成する．内部レイヤを $n$ 分木に割り当てる順番は深さ優先探索の順とする（図2）．次いで，各葉レイヤ $L$ を，レイヤ品質が $L$ に最も近い内部レイヤに付け加える．もし $L$ のレイヤ品質が $L$ の親レイヤの品質を超えるならば，$L$ のレイヤ品質は $L$ の親レイヤの品質に変更する．最終的にトランスコード木は，内部ノードと葉ノードを，それぞれそれらの要求品質の降順に内部レイヤと葉レイヤに割り当てることにより得られる．

**木の次数 $n$ とレイヤの大きさ $k$ の決め方**

提案手法では，トランスコード木は変則的な完全 $n$ 分木として構成される．したがって，$n$ の値が大きくなると，木の高さ（すなわち，トランスコード数）は減少する．各ノードの要求上り（送信）帯域は $n$ の値に比例して増加するので，$n$ の値は各ノードの上り（送信）帯域の制限を考慮して決めなければならない．上記ステップ1において，どのノードも要求品質を下げたくない場合には，不等式 (2) を満たすノードの数が $|U|/n$ 以上になるように $n$ の値を決める必要がある．一方，$f$ 個のノードが一つのレイヤから同時に離脱する可能性があるならば，そのレイヤの残りの $k - f$ 個のノードによって，ビデオを $n \cdot k$ 個の子ノードへ伝送できなければならない．したがって，各レイヤにおい

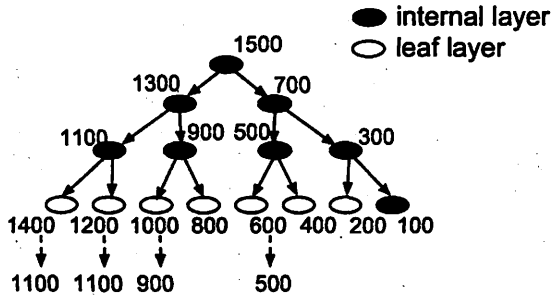


図 2: 深さ優先探索順のレイヤ木の構成

て最大 $f$ 個の同時故障（離脱）からの回復を可能にするには，以下の 2 つの不等式が満たされなければならない．$n$ と $f$ の値が決まれば，以下の不等式により $k$ の値が決まる．

$$(k-f)u.n_{link}(q) \geq n \cdot k \quad (3)$$

$$(k-f)u.n_{trans} \geq \lceil \frac{k}{u.n_{link}(q)} \rceil n \quad (4)$$

## 2.5 MTcast の動作

**(1) スタートアップ時の手順**

ビデオ配送の開始時刻を $t$ とする．各ユーザは，時刻 $t-\delta$ 以前にビデオサーバ $s$ にビデオ配送要求を送信する．時刻 $t-\delta$ に，サーバ $s$ は 2.4 節で説明したアルゴリズムによりトランスコード木 $T$ を計算する．$\delta$ は定数であり，トランスコード木の計算と，必要な情報を全ノードに配布するための時間である．また，サーバ $s$ は，次回にトランスコード木を計算するノード $u_C$ を決める ($s$ 自信が毎回計算ノードになっても良い)．$u_C$ は，十分な下り（受信）帯域を持つレイヤの代表ノードから選択する．次に，サーバ $s$ は，全ノードに以下の情報 $I$ または $I'$ を配布する．これらは，トランスコード木にそって配布する．

- 各レイヤの代表ノードに配布する情報 $I$
  トランスコード木 $T$ に含まれる全ノードとその親子関係，対応するレイヤ木の親子関係と各レイヤに含まれるノード，代表ノードと副代表ノード，およびレイヤ品質，次にトランスコード木の構築を行うノード $u_C$ と木の再構築予定時刻 $t_r$
- 代表ノード以外のノードに配布する情報 $I'$
  所属するレイヤの代表ノード，子ノード，親レイヤの代表ノードと副代表ノード，$u_C$ と $t_r$

**(2) 新規配信要求とノード故障に対する対処**

前述の通り，内部レイヤ内の各ノードは，ビデオストリームを 1 本分転送するための上り（送信）帯域幅の余裕を持っている．ビデオの配送開始時刻 $t$ 後にビデオ配送を要求したユーザノード $u_{new}$ は，この帯域幅を使用する．$u_{new}$ にストリームを送信する親ノード $u_f$ のノード接続可能数 (次数) は，一時的に $n+1$ となる．内部ノード $u_f$ は，既に子ノードに伝送しているビデオストリームを $u_{new}$ へ伝送するため，新たなトランスコードは必要ない．

あるレイヤ中の 1 個またはそれ以上のノードが故障・離脱した場合，トランスコード木内の故障・離脱ノードの子ノード（孤児ノードと呼ぶ）およびその子孫ノードはビデオストリームを受信できなくなる．提案手法では，孤児ノードはデータ $I'$ を受信しており，親ノードが所属するレイヤの代表ノードを知っているため，この代表ノードに依頼することで，代替親ノードを発見し即座に切り替えることができる．代替ノードが孤児ノードに即座にビデオストリームを転送できるよう，新規ノードの場合と同様に余剰帯域幅を利用する．親ノードが切り替わる間シームレスな再生

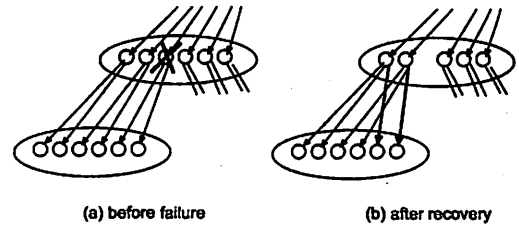


図 3: 故障ノード発生からのリカバリ

が可能とするため，バッファしておいたビデオデータを再生する．ノード故障時の親ノード切替の例を図 3 に示す．

**(3) トランスコード木の再構築**

トランスコード木は定期的に再構築することで，各ストリームの次数が $n$ 以下に調整し，消費された余剰上り帯域幅を再確保する．以下のステップでトランスコード木を再構成する．再構成後のトランスコード木に切り替える時刻 $t_r$ は全てのノードに予め配布しておく．まず，全てのノードから，トランスコード木に沿って，時刻 $t_r$ 後に有効となる品質要求を収集する．次に，2.4 節で述べた手順によりトランスコード木を計算し，全ノードに対して新しいトランスコード木と，次にトランスコード木を計算するノード $u'_c$ の情報を配布する．

時刻 $t_r$ に，全ノードはストリームの受信を一旦中止し，新しい木に沿ったビデオストリームの配信を開始する．新しいトランスコード木に沿って伝送されたビデオストリームは，トランスコードの処理遅延，オーバレイリンクの通信遅延により，ある時間 (**切替遅延時間** と呼ぶ) 遅れて各ノードに受信される．再生の途切れを防ぐため，各ノードでは切替遅延時間以上のある時間分のストリームデータを常にバッファしておく．トランスコード木の次の再構築時までに，各ノードのバッファは少なくとも切替遅延時間分のビデオデータで満たされねばならない．そのため，ビデオストリームを元々のビットレートより若干速く伝送する．

## 2.6 実環境に適用するための拡張

2.4 節で述べたトランスコード木の構築法は，インターネット上の AS(Autonomous System) 内の帯域と AS 間のバックボーン帯域を区別していないため，比較的貴重なバックボーン帯域を消耗してしまう可能性がある．本稿では，MTcast の優位性を保ったまま，新たにバックボーン帯域を節約する方法について述べる．

本方式で使用するバックボーン帯域は，実際の物理ネットワーク上のトポロジに依存する．基本方針として，2.4 のステップ 3 において，トランスコード木における親子関係を物理的なトポロジを考慮して決定する．

子レイヤに属するノードの集合を $C$，親レイヤに属するノードの集合を $P$ とする．子レイヤに属する各ノード $c \in C$，親レイヤに属する各ノード $p \in P$ に対し，物理ネットワーク上のリンクの集合 $L(c,p)$，および，利用可能帯域 $bw(c,p)$ を，traceroute，pathload 等のツールを用いて計測する．ラスト 1 ホップを除いたバックボーン帯域の節約が目的であるので，$c,p$ のラスト 1 ホップのリンクは $L(c,p)$ から除いておく．

次に，各物理リンクの利用可能帯域の推定値と共有度数を求める．リンク $l \in L(c,p)$ の利用可能帯域の推定値 $m_l$ と共有度数 $d_l$ は以下のように定義される．

$$m_l = Min\{\, bw(c,p) \mid l \in L(c,p),\ c \in C,\ p \in P\}$$
$$d_l = |\{\, c \mid l \in L(c,p),\ c \in C,\ p \in P\}|$$

以上で求めた各物理リンクの利用可能帯域の推定値および共有度数に基づき，各ノード $c \in C$ の親ノードを次のように決定する．

全ての$p \in P$に対し，$c$が，希望品質のストリームを受信可能かどうかを，各リンクの利用可能帯域の推定値を用いて調べる．ストリームを配信可能なノードが一つだけの場合は，そのノードを親ノードとして決定する．また，ストリームを配信可能なノードが複数存在する場合は，$L(c,p)$に属するリンクのうち，共有度数当たりの利用可能帯域の最小値を求め，これを最大化するノードを$c$の親ノードとして決定する．このようなノードが複数ある場合，最もホップ数の近いノードを親ノードとする．$c$の親ノード$p'$が決定した後は，各物理リンクに対し$c$が増加させた共有度数を無効にし，$L(c,p')$に属する各リンクの利用可能帯域の推定値から，$c$の要求品質分の帯域を引く．

# 3 プロトタイプシステムの実装

MTcastの性能を，インターネット上で評価することを目的に，システムの設計およびプロトタイプの実装を行った．本章では，その概要を説明する．

## 3.1 実装の方針

プロトタイプを実装する上で，以下の2点を重視してソフトウェアの設計を行った．(i) 実装を可能な限り容易にすること．(ii) 提案方式の変更に対し，プロトタイプの変更を最小限に抑え，またできる限り提案方式以外の類似方式の実装にも使用可能なこと．上記2点を満たした上で，できるだけ性能を犠牲にしないように工夫を行った．以下で説明するモジュラーデザインを採用することにより，ネットワーク上の各ノードで動くプロセスを汎用のコードとし，中央サーバのコードを変更するだけで，各ノードの機能を変更できるようにした．

## 3.2 モジュラーデザイン

ネットワーク上の各ノードの構成を柔軟に変更し，各種設定の下での性能の評価を容易にするために，各ノードで動くプログラムを，バッファやトランスコーダなどのソフトウェアモジュールの集合として構成した．各ノードで動作するプロセスは，中央サーバからの指示に従ってこれらのモジュールのインスタンスを生成し，接続する．また，中央サーバから各モジュールへの指示の中継を行う．この仕組みを実現するため，プロトタイプはJava言語を用いて製作し，中央サーバと各ノードで動くプログラムとの間の通信はRMI(Remote Method Invocation)を使用して行うこととした．ソフトウェアモジュールとして，トランスコーダ，ビデオの表示部，ネットワークからのデータの受信部および送信部，バッファ，ユーザインタフェース，設定ファイルの読み出し部等を用意した．これらはそれぞれ一つのクラスに対応するようにした．また，基本的に各モジュールは一つのスレッドとして動作するようにした．これにより，イベントドリブンのコードを書くよりも容易にコーディングが行えるようにした．

中央サーバの指示により各ノードのプロセス上で生成されたインスタンスは，ユニークなIDをキーとして，Hashtableに登録し，IDにより，参照およびメソッド呼び出しができるようにした．また，提案システムを実装する上で，各モジュールの状態の変化（接続が切れた，バッファされたデータ量が決められた値以上になった，等）に応じて，各種の処理を行う必要があるが，このために，中央サーバにキューを用意し，モジュールの状態が変化すると，モジュールはRMIにより中央サーバのキューに状態の変化を知らせるメッセージを入れるようにした．中央サーバは，順にキューからメッセージを取り出し，必要な処理を行う．

## 3.3 ビデオストリームを扱うための工夫

モジュール間のデータの送受信として，Javaクラスライブラリの InputStream/OutputStream のような仕組みを用いることもできるが，処理の途中でピクチャサイズを変更し，トランスコーダ等を再起動するような処理の実現が煩雑になる．プロトタイプシステムでは，モジュール間のデータの送受信に独自のクラスを用いた．これは，データ書き込み時に明示的にデータの切れ目を宣言し，読み込み側は宣言されたデータの切れ目までを通常と同じように読み込むことができるものであり，その後は一度EOFまで読んだ後と同じようにreadメソッドが$-2$を返し，続いて次のデータをまた読み込むことができる．これにより，各インスタンスを再生成することなしにデータの切れ目を容易に処理できるようにした．

JMF(Java Media Framework)は，各モジュールに配置されたバッファのため，遅延が非常に大きくなる．プロトタイプシステムは，遅延を可能な限り少なくするため，各モジュールにできるだけバッファを配置しないように構成した．プロトタイプシステムでは，各モジュールが基本的に一つのスレッドに対応しているが，書き込み側がwriteメソッドを呼び出すと，読み込み側がそのデータを全て読み出すまで，書き込み側が一旦ブロックするように構成した．これにより，writeメソッドが呼び出された時点でスレッドの切り替えが起こり，読み出し側のスレッドが有効になるため，少ない遅延での処理が可能になった．

## 3.4 多数のノードで評価するための工夫

評価実験をPlanetLab上の数十のノード上で実施するために，以下で述べる工夫を行った．PlanetLabのノードの状況は，刻々と変化し，ある日まで使用できたノードがその次の日には使えなくなったり，あるいは，その逆が毎日のように起こる．このような状況の変化に対処するため，自動的に各ノードの現在の状況を把握し，実験に必要な環境を設定するためのスクリプト群を作成した．このスクリプトは呼び出し元のノードで実行されるものと，各ノードで実行されるものに分けられる．呼び出し元のノードで実行されるスクリプトは，その他のスクリプトおよび必要なファイルを各ノードにscpコマンドによりコピーし，実行する．この作業は，ノード毎に並列に実行される．各ノードで実行されるスクリプトは，必要なファイルがノードにあるか調べ，必要に応じてjavaの実行環境(jre)などをwgetコマンドによりダウンロードし，インストールする．実際にこのスクリプトを使用したところ，奈良先端大のwebサーバより，PlanetLab上の20のノードにjreをダウンロードし，インストールするためにかかる時間は30分以下であった．また，製作したjavaのクラスファイル等を20のノードにインストールするのにかかる時間は1分程度であった．

# 4 評価

MTcastの有効性を検証するために，文献[8]で，(1) トランスコードに伴う負荷，(2) トランスコード木を再構築するときのオーバーヘッド，(3) ユーザの満足度について調べた．以下では，これらの概要に加え，新たに(4) トランスコードよるビデオ品質の劣化，(5) バックボーン帯域節約の効果 (6) WAN環境でのスタートアップ遅延とノード離脱時のデータ配信再開までの時間，の評価結果について述べる．

## 4.1 各種オーバヘッド

MTcastではユーザノード上でトランスコードを行うため，トランスコードによる負荷がビデオの再生に影響しないことが求められる．また，レイヤ内の最大離脱可能ノー

ド数 $k$ を求めるためには，様々な端末に対し，各端末がどの程度のトランスコード次数を確保できるかを調べる必要がある．デスクトップPC，ノートPC，PDAを用いて，ビデオを再生しながら同時にトランスコードを行う際の負荷を測定した結果，デスクトップPC，ノートPCそれぞれにおいて，トランスコード次数が1であれば，実時間でのトランスコード処理が可能であることを確かめている [8]．

また，木の再構築の際，(i) 各ノードからの品質要求の収集，(ii) トランスコード木の計算，(iii) 各レイヤの代表ノードへのトランスコード木に関する情報の配布にオーバヘッドが生じる．(i) に関して，計算ノードを大きな下り（受信）帯域を持っているノードから選択することで，オーバヘッドを十分に小さくできる見積もりを得ている．(ii) に関しては，Pentium 4 2.4GHz での計算時間が 1.5 秒程度であり，ボトルネックにはならない．(iii) に関しても，トランスコード木の一部の情報を木のそれぞれの部分に配布することで，配布する情報の量を実用上十分な値に減らすことが可能である [8]．

## 4.2 ユーザの満足度

文献 [8] で，MTcast が階層化マルチキャストより高い満足度を達成することをシミュレーション実験により確かめた．以下，その概要を述べる．

ユーザ $u$ の満足度 $0 \leq S_u \leq 1$ を文献 [3] と同様に，式 (5) のように定義した．ここで，$u.q$ はユーザ $u$ の要求品質（bps）を，$u.q'$ はユーザ $u$ が実際に受信できたビデオの品質（bps）を表す．$S_u$ の値が 1 に近いほど，ユーザの要求品質に近い再生品質が達成される．

$$S_u = 1 - \frac{|u.q - u.q'|}{u.q} \tag{5}$$

Inet 3.0 [10] を用いてノード数 6000 のトポロジを作成し，その中からリンク次数が 1 のノードを 1000 個選びユーザノードとした．ユーザノードが直接接続しているリンクを LAN 上のリンク，LAN 上のリンクに連結しているリンクを MAN 上のリンク，それ以外のリンクを WAN 上のリンクとする．LAN 上のリンクの下り（受信）帯域は，100〜500kbps(携帯端末)，2〜5Mbps(無線ブロードバンド)，10〜20Mbps(有線ブロードバンド) からシナリオに応じてある比率でランダムに選択した．各ユーザの要求品質は，300k から 3M まで一様分布とした．WAN 上のリンクの帯域は全て 6Gbps とした．

また，階層化マルチキャストにおける，ユーザの平均満足度をシミュレーションにより求めた．IP マルチキャストによる利用を想定し，各ノードはストリームの受信のみを行うものとする．ノード数を 1 から 1000 まで，階層数を 1 から 10 まで変化させ，平均満足度を求めた．

以上の実験より，ノード数が 200 以上の場合，階層数が 10 の階層化マルチキャスト手法に対し，提案手法は約 6%高い平均満足度を達成できることが分かった．実験の詳細は文献 [8] を参照のこと．

## 4.3 トランスコードによるビデオ品質劣化

ビデオを繰り返しトランスコードすると画質の劣化が生じる．MTcast での劣化の度合いを評価するため，あるビデオに対し，ビデオの画像サイズ，フレームレート，ビットレートを徐々に低くしてトランスコードを繰り返した場合と，同じ画像サイズ，フレームレート，ビットレートのビデオに一度のトランスコードで変換した場合について，品質劣化の程度を平均 PSNR 値を測定した．結果を図 4 に示す．結果より，4 回のトランスコードを繰り返した場合と，1 回のトランスコードで変換した場合の PSNR 値はほぼ同等であることが分かる．

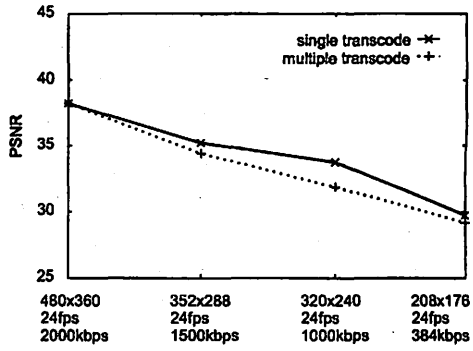


図 4: トランスコードに伴う PSNR 値の変化

## 4.4 バックボーン帯域節約の効果

本稿では，2.6 節で述べたように，実際の物理的ネットワークトポロジを考慮して，バックボーン帯域を節約する方法を提案した．以下では，この方法を使用した場合と，ランダムに子ノードの選択する場合をシミュレーションにより比較し，トランスコード木の全論理リンクにおける物理ホップ数の総和がどの程度異なるかを示す．

実験では，LAN 上のリンクの下り（受信）帯域は，4 つの場合を想定し，次の 3 種類からそれぞれ 1/3 の比率でランダムに選択する：(1) 携帯端末：100〜500kbps，(2) 無線ブロードバンド：2〜5Mbps，(3) 有線ブロードバンド：10〜20Mbps．また，リンクの上り（送信），下り（受信）帯域は対称とする．ノード数は 1000 とした．また，品質要求は一様分布に設定した．その結果，ランダムに選択する場合の物理ホップ数の総和は，4088，2.6 節の手法を使用する場合は，3121 となった．したがって，子ノードをランダムに選択する場合に比べ，総物理ホップ数を約 25%削減できることが分かった．

## 4.5 PlanetLab 上での評価

実環境上での提案手法の性能を評価するため，プロトタイプシステムを PlanetLab 上で稼働させた．以下，ユーザノードの新規参加時の遅延時間と，離脱時の遅延時間の計測結果について述べる．

### 実験の設定

解像度が 180 × 120 ピクセル，15fps のビデオを用い，音声は使用していない．ビデオコーデックとして Motion JPEG を用いた (MPEG から Motion JPEG に変換すると，サイズは約 4 倍になる)．

ユーザノード間のビデオデータの転送には，TCP を用いた．スケーラビリティを主に評価するため，トランスコード木を，$n$ 分木状に構成せず，1 レイヤを 2 ノードとし，レイヤを直列に接続した．評価実験では，18 の PlanetLab ノードと，それ以外のノードを 2 つ用いた．すなわち，10 段のトランスコード木と同等の実験を行った．これらのノードを表 2 に示す．表中の katsuo.naist.jp でビデオ配信サーバとユーザノードのプロセスを動作させた．その他のノードでは，ユーザノードのプロセスをそれぞれ一つ動作させた．PlanetLab の各ノードは恒常的に CPU 負荷が高く，リアルタイムトランスコードに必要な CPU 資源を確保するのが難しいため，実験では，各ノードでトランスコードを行わず，受信したデータを 1 フレーム分バッファしたあと，そのまま送信することとした．実際に一般ユーザの PC 上で提案手法を利用する場合，必要な CPU 資源およびメモリは十分に確保可能であると考えられる．

### スタートアップ遅延

表 3 にスタートアップにかかった時間を示す．コネクション確立とデータ受信開始は，それぞれスタートアップ

表 1: ノード離脱時の振る舞い

| 離脱ノード | 再接続先ノード | 再コネクション時間 | データ再受信開始 |
|---|---|---|---|
| planetlab1.netmedia.gist.ac.kr | thu1.6planetlab.edu.cn | 1,471ms | 1,805ms |
| planetlab4.ie.cuhk.edu.hk | planetlab1.netmedia.gist.ac.kr | 288ms | 411ms |
| planetlab-01.ece.uprm.edu | planetlab5.ie.cuhk.edu.hk | 1,114ms | 1,657ms |
| planetlab1.tmit.bme.hu | planetlab-02.ece.uprm.edu | 1,383ms | 1,626ms |
| planetlab01.cnds.unibe.ch | planetlab1.tmit.bme.hu | 1,541ms | 1,841ms |
| planetlab2.iii.u-tokyo.ac.jp | planet0.jaist.ac.jp | 61ms | 1,004ms |

表 2: 実験に用いたノード

| | |
|---|---|
| katsuo.naist.jp | pl1-higashi.ics.es.osaka-u.ac.jp |
| wakame.naist.jp | planetlab3.netmedia.gist.ac.kr |
| planet0.jaist.ac.jp | planetlab1.netmedia.gist.ac.kr |
| planetlab-01.naist.jp | planetlab2.iii.u-tokyo.ac.jp |
| planetlab-02.naist.jp | planetlab1.iii.u-tokyo.ac.jp |
| planetlab-03.naist.jp | planetlab5.ie.cuhk.edu.hk |
| planetlab-04.naist.jp | planetlab4.ie.cuhk.edu.hk |
| planetlab1.tmit.bme.hu | planetlab-02.ece.uprm.edu |
| planetlab2.tmit.bme.hu | planetlab-01.ece.uprm.edu |
| | planetlab02.cnds.unibe.ch |
| | planetlab01.cnds.unibe.ch |

開始から，最後のノードが親ノードとコネクションを確立するまで，および最初のデータを受信するのにかかる時間である．コネクションの確立は，それぞれのノードが並列に実行するので，単純に最もコネクションの確立に時間のかかったノードの時間となる．データ受信開始は全てのノードを経由する必要があるので，トランスコード木の高さに比例した時間がかかるはずであるが，実際にはコネクションにかかる時間が支配的であるため，目立った増加は見られない．いずれも最大でも 20 秒未満であり，十分実用的な数値となっている．

表 3: スタートアップ遅延

| ノード数 | コネクション確立 | データ受信開始 |
|---|---|---|
| 4 | 10,871ms | 19,040ms |
| 8 | 11,681ms | 16,993ms |
| 12 | 15,307ms | 15,307ms |
| 16 | 11,317ms | 18,781ms |
| 20 | 12,144ms | 17,562ms |

### ノード離脱時の振る舞い

表 1 に，ノード離脱時の再コネクションにかかる時間等について示す．再コネクション時間は，離脱ノードが離脱リクエストを送ってから，離脱ノードの下位ノードが新たな親ノード (再接続先ノード) とコネクションを確立するまでの時間であり，データ再受信開始時間は，新たな親から最初のデータを受け取るまでの時間である．上記の全 20 ノードを使用して，上記の実験と同様にレイヤが直列に繋がったトポロジでビデオ配信を行っている最中に，表中のそれぞれの離脱ノードが離脱したときの再コネクション完了までの時間を計測した．いずれの時間も 2 秒未満であり，十分実用的である．前述のように，この時間の間はあらかじめバッファされたビデオデータが再生すれば，ユーザはノードが離脱したことを意識することはない．

## 5 おわりに

本稿では，互いに異なる多ユーザに対する効率的な同時ビデオ配信のためのビデオ配送方法 MTcast を実装し，その有効性を PlanetLab 上での実験を通して示した．今後，ビットレートのみトランスコードするのではなく，画像サイズ，フレームレートを独立してトランスコードする方式に拡張したい．

## 参考文献


[1] G. Conklin, G. Greenbaum, K. Lillevold, and A. Lippman: "Video Coding for Streaming Media Delivery on the Internet," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 3, 2001.

[2] S. Jacobs and A. Eleftheriadis: "Streaming Video using Dynamic Rate Shaping and TCP Flow Control," Visual Communication and Image Representation Journal, 1998. (invited paper).

[3] J. Liu, B. Li, and Y.-Q. Zhang: "An End-to-End Adaptation Protocol for Layered Video Multicast Using Optimal Rate Allocation," IEEE Trans. on Multimedia, Vol. 6, No. 1, 2004.

[4] B. Vickers, C. Albuquerque and T. Suda : "Source-Adaptive Multilayered Multicast Algorithms for Real-Time Video Distribution," IEEE/ACM Trans. on Networking, Vol.8, No.6, pp.720-733, 2000.

[5] H. Radha, M. Shaar, Y. Chen: "The MPEG-4 Fine-Grained-Scalable video coding method for multimedia streaming over IP," IEEE Trans. on Multimedia, Vol. 3, No. 1, 2001.

[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee and S. Khuller : "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," Proc. of IEEE Inforcom 2003, pp.1521-1531, 2003.

[7] V. Padmanabhan, H. Wang, P. Chow and K. Sripanidkulchai : "Distributiong streaming media content using cooperative networking," Proc. of the 12th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002), pp.177-186, 2002.

[8] T. Sun, M. Tamai, K. Yasumoto, N. Shibata, M. Ito and M. Mori : "MTcast : Robust and Efficient P2P-Based Video Delivery for Heterogeneous Users," Proc. of the 9th Int'l. Conf. on Principles of Distributed Systems (OPODIS 2005), pp.176-190, 2005.

[9] D. Liu, E. Setton, B. Shen and S. Chen : "PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices," Proc. of 17th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2007), pp.51-53, 2007.

[10] J. Winick and S. Jamin: "Inet3.0: Internet Topology Generator," Tech. Report UM-CSE-TR-456-02 (http://irl.eecs.umich.edu/jamin/), 2002.

[11] PlanetLab: "An open platform for developing, deploying, and accessing planetary-scale services," https://www.planet-lab.org/.