異なる品質要求を持つ複数ユーザへのピアツーピアビデオ配信手法

柴田直樹[†1] 孫弢[†2] 玉井森彦[†2]
安本慶一[†2] 伊藤実[†2] 森將豪[†1]

本論文では，多数のユーザが異なる品質で同じビデオの配信を要求する場合に，各ユーザノードにトランスコードおよび中継を行わせることで，ピアツーピアネットワークにおいて効率良くビデオを同時配信する MTcast と呼ぶ新しいビデオ同時配信手法を提案する．MTcast では，各ユーザ（ビデオ受信者）が各自の環境の制約に基づいて決定した要求品質を指定したビデオ配信要求を送ると，すべてのユーザが 1 つの配送木を経由して要求と同じかそれに近い品質のビデオを受信することができる．MTcast は，ユーザ数に対する高いスケーラビリティ，受信品質に対する高いユーザ満足度，配送開始までの待ち時間の短さや，ノードの故障に対する耐頑健性等の特徴を持つ．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することを確認した．

A Video Delivery Method for Users with Different Quality Requirements in P2P Networks

Naoki Shibata,[†1] Tao Sun,[†2] Morihiko Tamai,[†2]
Keiichi Yasumoto,[†2] Minoru Ito[†2] and Masaaki Mori[†1]

In this paper, we propose a new video delivery method called *MTcast* (*Multiple Transcode based video multicast*) which achieves efficient simultaneous video delivery to multiple users with different quality requirements by relying on user nodes to transcode and forward video to other user nodes in P2P networks. In MTcast, each user specifies a quality requirement for a video consisting of bitrate, picture size and frame rate based on the user's environmental resource limitations. All users can receive video with the specified quality (or near this quality) along a single delivery tree. The main characteristics of MTcast are in its scalability, high user satisfaction degree in received video quality, short startup latency and high robustness against node failure. Through simulations and experiments with our prototype implementation on PlanetLab, we have confirmed that MTcast can achieve practically high scalability and robustness, short startup latency and much higher user satisfaction than the layered multicast method.

1. はじめに

近年のインターネットの急速な普及とブロードバンド化，接続端末の多様化にともない，計算処理能力，表示サイズ，ネットワークの利用可能帯域幅等環境の異なる多数のユーザ端末に対し，ビデオ配信を効率良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方法として，マルチバージョン法[1)]，オンライントランスコード法[2)]，階層化マルチキャスト法[3)] 等の様々な方法が提案されてきた．マルチバージョン法では，異なるビットレートを持つ複数個のビデオバージョンをあらかじめ用意し，ユーザからの要求に対し，制約を満たす中で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では，元のビデオをサーバまたは中間ノード上で，ユーザが希望する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)] では，ビデオを階層符号化[5)] を用いて複数の層に符号化し，各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．各層のデータは独立したマルチキャストストリームとして配信されるので，各ユーザは自己の制約範囲内で可能な限りの複数個の層を受信できる．この方式では，ユーザ満足度を改善するためにはより多くの層

†1 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University
†2 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

数が必要となり，しかも多くの層からのビデオ復号にはデコードのために大きな処理能力とバッファが必要になる．マルチバージョン法は制御方式は簡単であるが，サーバのディスクスペースやネットワーク帯域の利用効率が低い．マルチバージョン法，階層化マルチキャスト法では，もし十分な数のバージョンや層が用意されないならば，要求品質と配信品質の間にズレが生じるという問題がある．一方，オンライントランスコード法では，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も果たしており，マルチキャスト木はビデオ配信サービスが木を通してすべてのユーザノードに提供されるように構成される．OMNI は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 では，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が議論されている．そこでは，ユーザノードはストリームデータの一部分をキャッシュし，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が，サーバの負荷は高い．OMNI と CoopNet は，ネットワーク条件の動的な変化やサーバ負荷等に依存してビデオ配信サービスを適応させることを目的としているが，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．

本論文では，同じビデオの配信に対し異なる品質要求を持つ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，効率良くビデオを同時配信する方式 MTcast（Multiple Transcoded based video multicast）[8] を提案する．MTcast では，各ユーザ（ビデオ受信者）は各自の環境の制約に基づいて決定したビットレート，画面サイズ，フレーム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．この際，ビデオの時間帯ごとあるいはシーンごとに異なる要求品質を指定することもできる．

提案方式では，6 項目，(1) 高い拡張性：受信ノード数の増加に対応できること，(2) 高いユーザ満足度：受信する品質が要求した品質に近いこと，(3) 妥当な必要資源量：配信システムの制御機構に使用される計算機資源が妥当であること，(4) 短い配送開始遅延：リクエストを送信後配信が開始されるまでの時間が短いこと，(5) 短い配送遅延と妥当なトランスコード回数：ビデオソースがデータを送信してから末端のユーザノードがデータを受信するまでの時間が短いこと，(6) 高い耐故障性：ノードやリンクの障害が起こっても配信が継続されること，を達成することを目標とする．

提案方式では，上記 (1)–(3) を実現するために，その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．ここでは，より高い品質要求を持つユーザノードは木の根近くに，そして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良く実現する．上記の (4)–(6) を実現するために，全ノードを互いに近い品質要求を持つ k 個のノードからなるレイヤと呼ばれるグループ群に分割し，要求する品質によりレイヤ間の親子関係を設定し，各レイヤのノード群が，その子レイヤのノード群に対し，共同でストリーム配信を行う機構を考案・導入した．本機構により，新規参入ノードへすばやくストリームの配送を開始でき，あるノードが配信元ノードの故障等によりストリームの配信が受けられなくなった場合に，配信元ノードを同じレイヤの別ノードに高速に切り替えることが可能である．

MTcast を実装し，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. 対象とする環境

本論文では，以下のような環境を想定している．

- ネットワーク環境：複数ドメインにまたがるオーバレイネットワーク
- ユーザ端末：デスクトップ PC，ラップトップ PC，PDA，携帯電話，etc.
- 通信インフラ：固定ブロードバンド（専用線，ADSL，CATV，etc.），無線回線（無線 LAN，W-CDMA，Bluetooth，GSM/PDC，etc.）のいずれかを介してインターネットに接続
- ユーザ数：500〜100,000 程度
- 対象コンテンツ：ビデオ（録画済み，生放送の

両方）

本論文では，TV 放送のように，あるビデオコンテンツを決められた時間にいっせいに配信開始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．ただし，ユーザは放送開始後いつでも，現在放送中のシーンからのビデオの配信を受けることができる．

ユーザノードはオーバレイリンクにより互いに結ばれているものとし，各ノードは他のノードと送受信するために TCP または UDP を用いるものとする．

3. MTcast の詳細

まず MTcast アルゴリズムで用いられる記法について定義し，MTcast の詳細について説明する．

3.1 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 $U = \{u_1, \ldots, u_N\}$ 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）はあらかじめ既知であると仮定し，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．一般に $u_i.q$ で，ビットレート，画像サイズ，フレームレートの組が指定可能であるが，本論文では簡単のためビットレートのみを扱うものとする[⋆1]．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 $u_i.n_{trans}(q)$ で，また，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．

提案方式では，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．

3.2 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．提案方式では，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 と仮定する．3.3 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まる．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．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．

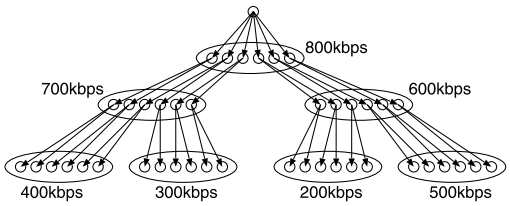

図 1 トランスコード木（$n = 2$，$k = 6$）による配信の様子
Fig. 1 Example of transcode tree, in case of $n = 2$, $k = 6$.

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 1 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 1 つ選ばれる．トランスコード木上のすべ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．

3.3 トランスコード木の構築

提案方式では，トランスコード木の計算をただ 1 つのノード u_C で集中的に行う．ただし，u_C はトランスコード木を構築するごとに変更でき，その決め方は 3.4 節で説明する．ここで，u_C 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 の情報を有するものと仮定する．トランスコード木構築アルゴリズムは以下の 3 つのステップからなる．

ステップ 1：ユーザノードの集合 U を，内部ノードの集合 U_I と葉ノードの集合 U_L に分割する．根ノード s はつねに U_I に入れる．各ノード $u \in U$ に対して，アルゴリズムは下記の不等式が成り立つか否かを調べる．

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

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

上記不等式が成り立てば，u を U_I に入れ，さもな

⋆1 パラメータベクトルを品質として扱う方法は文献 9) で議論している．

くば U_L に入れる．不等式 (1) と (2) は，ノード u が 1 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，そして u が $(n+1)$ 本のビデオストリームを同時転送できることを，それぞれ表している．

分割後に $|U_I| < |U|/n$ であれば，すべ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とが分かる．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ことが考えられる．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 $|U_I| > |U|/n$ がつねに成り立つように調整できる．

ステップ 2：全ノード集合 U をレイヤに割り当てる．U_I の要素はそれらの要求品質の降順にソートされ，k 個ずつ要素が束ねられて，1 つの内部レイヤに割り当てられる．各レイヤの最初，2 番目のノードをレイヤの**代表ノード**，**副代表ノード**として選ぶ．要求品質の最小値がレイヤ品質として割り当てられる．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：トランスコード木が構成される．アルゴリズムは内部レイヤをレイヤ品質の降順にソートし，各レイヤのレイヤ品質がその親レイヤの品質を超えないように，それら内部レイヤの完全 n 分木を構成する．次いで，アルゴリズムは各葉レイヤ 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合わせられる．内部レイヤを n 分木に割り当てる順番は深さ優先（depth-first），幅優先（breadth-first），等がある．**図 2** に深さ優先を用いた例を示す．

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．

u_c がトランスコード木を計算するうえで，最も計算量オーダの大きな処理は，ステップ 3 におけるソートである．したがって，全体の計算量オーダは $O(Nlog(N))$ となる．

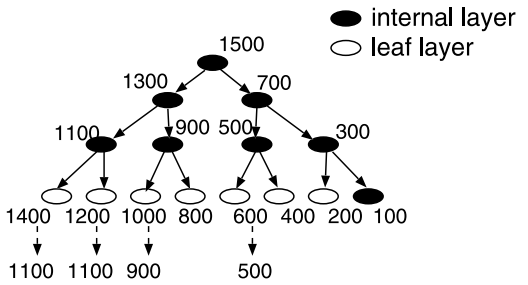

図 2 深さ優先探索順のレイヤ木の構成
Fig. 2 Layer tree construction in order of depth-first search.

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変更せずに済ませたい場合には，不等式 (2) を満たすノードの数が $|U|/n$ 以上になる範囲で n の最大値を決める．

一方，トランスコード木が構成される以前に，f 個のノードが 1 つのレイヤから同時に離れるならば，そのレイヤの残りの $k-f$ 個のノードは，ビデオを $n \cdot k$ 個の子ノードへ伝送しなければならない．したがって，各レイヤにおいて最大 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．

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

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

したがって，f と n の値を先に決めたうえで，上記の不等式を満たす k の値を求めればよい．

3.4 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 で表す．ビデオストリームの受信を希望するユーザは，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 3.3 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．ここで，δ は定数であり，トランスコード木を計算し，必要な情報をすべてのノードに配布するためにかかる時間である．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．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される．次いで，サーバ s は，トランスコード木 T に含まれる全ノードへ，ビデオ配送に必要な以下の情報 I または I' を配布する．

- 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 の情報（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情報（レイヤ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，親ノードが所属するレイヤの代表ノードと副代表ノード，u_C と t_r

情報配布のため，サーバ s はまずデータ I を根レイヤの代表ノードに送る．代表ノードは子レイヤの代表ノードに I を転送する．これを葉レイヤに到達するまで繰り返す．また，各レイヤの代表ノードは，I から I' を抽出し，同じレイヤの他のノードに配布する．これらの情報がすべてのノードに行きわたると，ビデオ配送の準備が整う．

(2) 新規配信要求とノード故障に対する対処

3.3 節で説明したように，内部レイヤ内の各ノードは，ビデオストリームを 1 本分余分に転送するための特別な上り（送信）帯域幅を持っている．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，ビデオストリームを受信するために，この余剰帯域幅を使うことができる．ここで，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 $n+1$ となることを許す．内部ノード u_f は n 個の子ノードを持ち，すでにこれらの子ノードに伝送しているビデオストリームを u_{new} へ伝送すればよいため，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．

トランスコード木を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（送信）帯域幅を再確保する．もし，レイヤの代表ノードが故障・離脱したとき，代表ノードの子ノードは新たな親ノードを見つけることができない．代表ノードの故障・離脱時には，子ノードの 1 つ u がそれを検出し，副代表ノードに代表ノード切替え要求メッセージを送信し，これを受信した副代表ノードは，以降代表ノードとして振る舞う（この際，新たな副代表ノードを割り当て，子レイヤのノード群に通知を行う）．なお，あるレイヤ L において代表ノードと副代表ノードが同時に故障・離脱したときには，それらのノードの故障・離脱を検知したノード u が，u の属するレイヤの代表ノード u_r に通知し，u_r は所有するレイヤ木の情報から L の新たな代表ノードを決定し，そのノードに切替え要求メッセージを送信する．

(3) 新規配信要求に対する手続き

新しくビデオ配信を要求するユーザノード u_{new} は，ビデオストリームをすでに受信しているノードを少なくとも 1 つ知っていると仮定し，そのノードを u^* とする．u_{new} は，トランスコード木の中でビデオストリームを転送してもらうのに最良の親ノードを以下の手順で見つけ，ビデオの配送を要求する．(1) u_{new} は自分が要求する品質 $u_{new}.q$ を指定したビデオ配信要求を u^* に送信する．(2) もし u^* がレイヤの代表ノードでなければ，u^* は自身の属するレイヤの代表ノード（u_r と表記する）に受信した要求を転送する．(3) u_r は要求を受信すると，自身が保持するレイヤ木の情報を u_{new} に送る．(4) u_{new} はレイヤ木の情報を受信すると，$u_{new}.q$ に最も近いレイヤ品質を有するレイヤを見つけ出して，そのレイヤの代表ノード u'_r にビデオ配信要求を送信する．(5) u'_r はそのレイヤから親ノードとして，余剰上り（送信）帯域を持つノード u' を選択し，u_{new} のビデオ配信要求を転送する．(6) 最後に，u' が u_{new} にビデオストリームの配送を開始する．

(4) ノード故障・離脱からの回復

各ノード u は，ある指定された時間内にいかなるデータも受信しなかった（または平均データ受信率が期待されている値に比べ極端に小さかった）ときに，親ノードが故障・離脱したと見なす．u が自分の親ノード u_p の故障・離脱を検出すると，u は u_p のレイヤの代表ノードに対してビデオ転送要求を送信する．新たなビデオ配送要求の場合と同様，ビデオストリームを転送可能な余剰上り（送信）帯域を持っているノードがあれば，そのノードを代替ノードとして選択し，ストリームを転送させる．親ノードが切り替わる間，u は自身でバッファしておいたビデオデータを再生することで，シームレスな再生が可能となる．ノード故障時の親ノード切替えの例を**図 3** に示す．

(5) トランスコード木の再構築

ノード u_C は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 はすべてのノードにあらかじめ配布されている（スタートアップ時の手順参照）．時刻 $t_r - \delta'$ 以前に，あるユーザノード u が受信ビデオ品質の変更を

行いたい場合，u は新しい品質要求を u' の現在のレイヤの代表ノードへ送信することができる．ここで，δ' はすべてのノードの品質要求を収集し，トランスコード木を計算し，必要な情報を全ノードへ配布するのに要する時間である．各レイヤ L の代表ノード u_L は，レイヤ L のすべてのメンバと，（L が子レイヤを持っていれば）L の子レイヤの代表ノードから品質要求を受信すると，受信した品質要求を含むリストを作成し，レイヤ L の親レイヤの代表ノードに対して，送信する．上の手順で，u_C は，時刻 t_r 後に有効となる予定のすべてのノードの要求品質を収集する．

次に，ノード u_c は，3.3 節のアルゴリズムによりトランスコード木を計算し，すべての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 の情報を分配する．

時刻 t_r に，すべてのノードは現在の親ノードから配信されているストリームの受信を中止し，新しいトランスコード木の根レイヤに属するノードは，新しい木に沿ったビデオストリームの配信を開始する．内部レイヤに属するノードもまた，ストリームを受信すると新しい子レイヤのノードに転送する．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（**切替え遅延時間**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，各ノードでは切替え遅延時間以上のある時間分のストリームデータをつねにバッファしておき，そこからビデオを再生させる．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え遅延時間分のビデオデータで満たさなければならない．そのため，ビデオストリームをそのビットレートより若干速く伝送する．

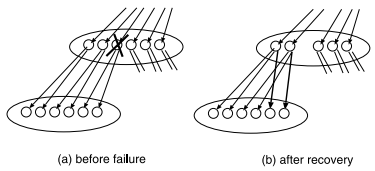

図 3 故障ノード発生からのリカバリ
Fig. 3 Recovery from node failure.

4. 評　　価

MTcast の有効性を検証するために，4 つの項目，(i) トランスコードにともなう負荷，(ii) トランスコード木を再構築するときのオーバヘッド，(iii) ユーザの満足度，(iv) 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，の評価実験を行った．

4.1 トランスコードにともなう負荷

提案手法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．そこで，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．実験では，各種端末上でビデオを再生，トランスコードする際の最大処理速度（fps）を求め，ビデオの実際の再生速度（fps）と比較した．トランスコード次数（同時にトランスコードするビデオの本数）を 1 から 3 まで変化させ，最大処理速度の変化を計測した．デコーダとして $mpeg2dec-0.4.0b$，エンコーダとして $ffmpeg0.4.9-pre1$ を用いた．実験に使用したパラメータと実験結果を**表 1** に示す．表中の各端末の仕様は，デスクトップ PC（CPU: Pentium4 2.4 GHz），ノート PC（CPU: Celeron 1 GHz），PDA（SHARP Zaurus SL-C700，CPU: XScale PXA250 400 MHz）である．また，ビデオのフレームレートはすべて 24 fps である．表 1 より，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，十分な処理速度を達成できることが分かる．また，PDA は内部ノードになれないことが分かる．

4.2 トランスコードにともなう画質劣化

提案方式では，トランスコード木の根ノードから葉ノードまで複数回のトランスコードを行うが，ビデオを繰り返しトランスコードすることで画質の劣化が生

表 1 ビデオ再生中のトランスコード処理速度
Table 1 Maximum processing speed while playing back a video.

device	original video		transcoded video		transcoding degree		
	picture size	bit rate (kbps)	picture size	bit rate (kbps)	1	2	3
Desktop PC	640 × 480	3,000	480 × 360	2,000	35.66	20.03	14.84
Desktop PC	480 × 360	2,000	352 × 288	1,500	61.60	36.40	25.89
Laptop PC	352 × 288	1,500	320 × 240	1,000	49.90	30.65	21.84
PDA	320 × 240	1,000	208 × 176	384	10.12	6.04	4.33

じる可能性がある．そこで，あるビデオに対し，ビデオの画像サイズ，フレームレート，ビットレートを変化させてトランスコードした場合について品質劣化を平均 PSNR 値を測定することで調べた．

640 × 480，24 fps，3,000 kbps のビデオを，208 × 176，24 fps，384 kbps のビデオへ，表 1 の各デバイスにおけるパラメータ値に従い，4 回のトランスコードを経て変換した場合の PSNR 値は，それぞれ 38，34，32，30 となった．一方，640×480，24 fps，3,000 kbps のビデオを各パラメータ値へ 1 回のトランスコードで変換した場合の PSNR 値はそれぞれ 38，35，34，31 となった．したがって，4 回のトランスコードを経て段階的に品質を下げる場合，1 回のトランスコードで変換する場合に比べて，最大 6%程度 PSNR 値が劣化することが分かった．本ビデオ配信方式で数万ノードに配信する場合に相当する，8 回程度トランスコードが繰り返される場合でも画質の劣化は 10%程度に収まり，十分実用に耐えることが分かった．

4.3 木の再構築にともなうオーバヘッド

提案方式では，木の再構築のとき，(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．

上記 (i) に関して，100,000 個のノードが 50 Byte の品質要求メッセージを計算ノード u_C に送る場合には，5 MByte のメッセージが u_C に送られる[⋆1]．送信時間を 10 秒とする場合（送信時間はトランスコード木の再構築周期より短い必要がある），平均送信スピードは 4 Mbps になる．大きな下り（受信）帯域を持っているノードのみ u_c として選択可能とすることで，(i) は帯域のボトルネックにならないと考えられる．なお，各ノードが現在のトランスコード木に沿って品質要求メッセージを転送すれば，中間ノードが自分の要求と子ノードのメッセージをマージして送信できるので，u_C が必要な下り帯域は十分に少なくできる．また，各ノードの要求品質を収集する際，あるレイヤの代表ノードに集まるデータ量は，u_c に集まるデータ量の約 $\frac{1}{n^m}$ になる（n，m はそれぞれ木の分岐数とレイヤの深さ）．幅優先の順で情報収集木を構築するので，情報収集のための制御トラフィックによる利用可能帯域の圧迫は無視できるほど小さいと考えられる．

上記 (ii) と (iii) に関して，ユーザ数を 1,000 から 100,000 に変化させ，トランスコード木のサイズと計算時間を調べた．ここで，トランスコード木のパラメータを $n = 2$，$k = 6$ とする．結果を**表 2** に示す．

表 2 木の再構築にともなうオーバヘッド
Table 2 Size and computation time of a transcode tree.

number of nodes	computation time (sec)	size of tree (byte)
1,000	0.016	3 K
10,000	0.140	30 K
100,000	1.497	300 K

100,000 ノードまでの木の計算時間は約 1.5 秒であり（Pentium 4，2.4 GHz），計算時間はボトルネックにならない．

ユーザ数が 10,000 のとき，トランスコード木の構造に関する情報サイズは 30 Kbyte であり，この情報をレイヤ木に沿って各レイヤの代表ノードに 10 秒で送信するとき，代表ノードが必要とする帯域は 24 Kbps となり，実運用上での許容範囲に収まる．100,000 ユーザの場合には，この帯域は 240 Kbps になるが，`gzip` のような圧縮アルゴリズムを用いて木のデータのサイズを削減したり，あるいはトランスコード木の一部の情報のみ代表ノードに送ることでこの値を実用範囲内に収めることが可能である．

4.4 ユーザの満足度

本節では，ユーザ間での異なる品質要求に対応するためのビデオ配信手法として一般的に参照される階層化マルチキャストとの比較により，提案方式の有効性を検証する．ユーザ u の満足度 $0 \leq S_u \leq 1$ を文献 3) と同様に，次のように定義する．

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

ここで，$u.q$ はユーザ u の要求品質（bps），$u.q'$ 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．

実験は次のような設定で行った：Inet 3.0 [10] を用いてノード数 6,000 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 1,000 個選びユーザノードとする．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，4 つの場合を想定し，次の 3 種類から**表 3** に指定した比率でランダムに選択する：(1) 携帯端末：100〜500 kbps，(2) 無線ブロードバンド：2 Mbps〜5 Mbps，(3) 有線ブロードバンド：10 Mbps〜20 Mbps．また，リンク

⋆1 木を再構築するとき，すべてのノードではなく，新規加入ノードと要求品質が変わるノードのみが計算ノードに品質要求を送るので，必要メッセージ量は実際にはこれより少ない．

表 3 実験の帯域設定
Table 3 Available bandwidth setting in the experiment.

	100 k to 500 k	2 M to 5 M	10 M to 20 M
case1	33%	33%	33%
case2	5%	33%	62%
case3	45%	10%	45%
case4	62%	33%	5%

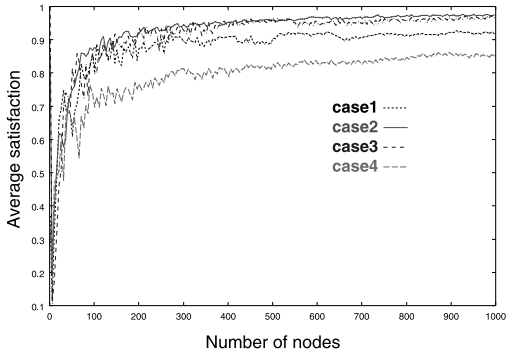

図 4 利用可能帯域の分布とユーザ満足度の関係
Fig. 4 Average user satisfaction depending on available bandwidth distribution.

の上り（送信），下り（受信）帯域は対称とする．

各ユーザの要求品質は，300 k から 3 M まで一様分布とした．MAN 上のリンクの帯域は，連結している LAN 上のリンクの帯域の合計とする．また，WAN 上のリンクの帯域はすべて 6 Gbps とする．

提案方式における，ユーザの平均満足度（$\sum S_u/N$，ただし N はユーザ数）をシミュレーションにより計測した．ユーザ数は 1 から 1,000 まで変化させ，3 種類の品質要求と表 3 に示す 4 種類のユーザ端末における利用可能帯域の分布について，トランスコード木の次数 $n = 2$，レイヤの大きさ $k = 6$ に設定した際の平均満足度を測定した．実験結果を**図 4** に示す．ここで，図の x 軸と y 軸はそれぞれユーザノード数と満足度を示す．

ユーザの利用可能帯域の分散の変化（たとえば，case2〜case3）が満足度に与える影響は小さいことが分かる．また，ユーザの利用可能帯域の平均が大きいほど満足度は高くなることが分かる．また，上記以外にも要求品質の分布を変えて実験を行い，満足度がそれほど変わらないことを確認している．

比較のため，階層化マルチキャストを使用した場合のユーザの平均満足度をシミュレーションにより求めた．シミュレーションでは，表 3 の case1 の分布のみ使用し，階層数は 4，6，8，10 とした．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．階層化マルチキャストにおける各層の符号化レートは焼き鈍し法により，ユーザ満足度が高くなるよう最適化した（文献 3) 参照）．ノード数を 1 から 1,000 まで，階層数を 1 から 10 まで変化させたときの平均満足度を求めた．測定結果を**図 5** に示す．

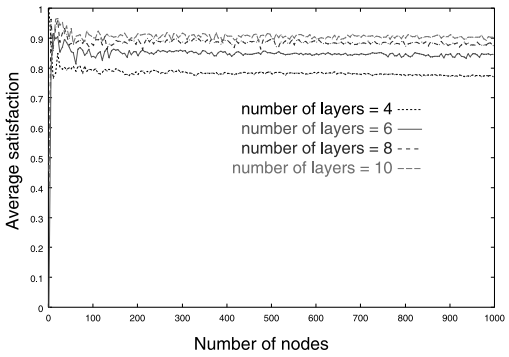

図 5 階層化マルチキャストのユーザの満足度の平均値
Fig. 5 Average user satisfaction on layered multicast method.

図 4 の case1 と図 5 の比較より，ノード数が 200 以上の場合，提案手法は階層化マルチキャスト手法（ただし，階層数は 10 以下）より高い平均満足度を達成できることが分かる（文献 3) より，計算量の観点から，階層化マルチキャスト手法での階層数は 10 程度までが現実的な範囲である）．

4.5 PlanetLab 上での実装と評価

現実の WAN 環境において提案手法の性能およびスケーラビリティを評価するため，提案手法を実装し，インターネット上に構築された WAN テストベッドである PlanetLab[11] 上の世界各国のノードを用いて評価を行った．以下，評価実験に使用したプロトタイプシステムの実装の詳細について述べた後，評価実験の設定と結果について述べる．

実装の詳細

Java 言語を用いてプロトタイプシステムを作成した．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等のソフトウェアモジュールの集合とし，中央サーバからの指示に従って自由に構成できるようにした．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 1 つのクラスに対応するようにした．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 Java RMI（Remote Method Invocation）を使用して行うこととした．中

央サーバの指示により生成したクラスのインスタンスは，Hashtable に登録し，このときに同時に指定した ID により，参照およびメソッド呼び出しができるように構成した．また，提案システムを実装するうえで，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．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ごとに RMI によりこのキューに状態の変化を知らせるメッセージを入れ，中央サーバは，順にキューからメッセージを取り出し，必要な処理を行うようにした．

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，ノードごと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 の実行環境等を wget コマンドによりダウンロードし，インストールする．

実験の設定

ユーザノードの新規参加時の遅延時間と，離脱時の遅延時間を計測した．実験には，ビデオコーデックとして Motion JPEG を用いた．ユーザノード間のビデオデータの転送には，TCP を用いた．評価実験では，ビデオのみを扱い，音声は使用していない．実験では，解像度が 180×120 ピクセル，15 fps のビデオを用いた．なお，MPEG から同じ解像度・フレームレートの Motion JPEG に変換すると，ファイルサイズは約 4 倍になる．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 4** 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，

表 4 実験に用いたノード
Table 4 Nodes used in experiments.

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

表 5 スタートアップ遅延
Table 5 Time required for starting up.

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

ユーザノードのプロセスをそれぞれ 1 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[⋆1]，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 5 にスタートアップに必要な時間を示す．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．データ受信開始はすべ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[⋆2]．

ノード離脱時の振舞い

表 6 に，ノード離脱時の再コネクションにかかる時

⋆1 実験では Java AWT 内の JPEG コーデックを使用したため，通常のコーデックを利用するよりも負荷が高い．
⋆2 これらの時間には，java の JIT コンパイルにかかる時間が含まれている．

表 6 ノード離脱時の振舞い
Table 6 Time required for filling leaving node.

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	1,471 ms	1,805 ms
planetlab4.ie.cuhk.edu.hk	planetlab1.netmedia.gist.ac.kr	288 ms	411 ms
planetlab-01.ece.uprm.edu	planetlab5.ie.cuhk.edu.hk	1,114 ms	1,657 ms
planetlab1.tmit.bme.hu	planetlab-02.ece.uprm.edu	1,383 ms	1,626 ms
planetlab01.cnds.unibe.ch	planetlab1.tmit.bme.hu	1,541 ms	1,841 ms
planetlab2.iii.u-tokyo.ac.jp	planet0.jaist.ac.jp	61 ms	1,004 ms

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つな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されるので，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

本論文では，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を達成するための新しいビデオ配送方法を提案し，その有効性をシミュレーションおよび PlanetLab 上での実験を通して示した．

本論文では，トランスコード木の構成は単一ノードで集中アルゴリズムにより計算することを想定した．実装したアルゴリズムは 100,000 ノード程度の規模であれば十分実用的な時間で動作することを確認したが，今後，対規模拡張性をさらに改善するために木構築のための分散アルゴリズムを考案していきたい．対規模拡張性を改善する最もシンプルな方法は，数万ノードごとにトランスコード木を 1 つずつ用意することである．これによって提案方式を大幅に変更することなく目的を実現できる．その一方で，ユーザノード数が増大したときに問題となりうるのは，バックボーンネットワークの物理リンクのリンクストレス（その物理リンクを利用するオーバレイリンクの本数）が高くなり，帯域を無駄に消費することである．この問題に対処するため，トランスコード木の計算の際に物理的なネットワークトポロジを考慮する方法について現在検討中である．

参考文献

1) Conklin, G., Greenbaum, G., Lillevold, K. and Lippman, A.: Video Coding for Streaming Media Delivery on the Internet, *IEEE Trans. Circuits and Systems for Video Technology*, Vol.11, No.3 (2001).
2) Jacobs, S. and Eleftheriadis, A.: Streaming Video using Dynamic Rate Shaping and TCP Flow Control, *Visual Communication and Image Representation Journal* (1998). (invited paper).
3) Liu, J., Li, B. and Zhang, Y.-Q.: An End-to-End Adaptation Protocol for Layered Video Multicast Using Optimal Rate Allocation, *IEEE Trans. Multimedia*, Vol.6, No.1 (2004).
4) Vickers, B., Albuquerque, C. and Suda, T.: Source-Adaptive Multilayered Multicast Algorithms for Real-Time Video Distribution, *IEEE/ACM Trans. Networking*, Vol.8, No.6, pp.720–733 (2000).
5) Radha, H., Shaar, M. and Chen, Y.: The MPEG-4 Fine-Grained-Scalable video coding method for multimedia streaming over IP, *IEEE Trans. Multimedia*, Vol.3, No.1 (2001).
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 Inforcom 2003*, pp.1521–1531 (2003).
7) Padmanabhan, V., Wang, H., Chow, P. and Sripanidkulchai, K.: Distributiong streaming media content using cooperative networking, *Proc. 12th Int'l. Workshop on Network and Operating Systems Support for Digital Audio and Video* (*NOSSDAV 2002*), pp.177–186 (2002).
8) Sun, T., Tamai, M., Yasumoto, K., Shibata, N., Ito, M. and Mori, M.: MTcast: Robust and Efficient P2P-Based Video Delivery for Heterogeneous Users, *Proc. 9th Int'l. Conf. on Principles of Distributed Systems* (*OPODIS 2005*), pp.176–190 (2005).
9) Yamaoka, S., Sun, T., Tamai, M., Yasumoto, K., Shibata, N. and Ito, M.: Resource Aware

Service Composition for Video Multicast to Heterogeneous Mobile Users, *Proc. 1st Int'l. Workshop on Multimedia Service Composition* (*MSC'05*), pp.37–46 (2005).
10) Winick, J. and Jamin, S.: Inet3.0: Internet Topology Generator, Tech. Report UM-CSE-TR-456-02 (2002). http://irl.eecs.umich.edu/jamin/
11) PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. https://www.planet-lab.org/

(平成 19 年 5 月 18 日受付)
(平成 19 年 11 月 6 日採録)

柴田 直樹（正会員）

1996 年，1998 年，2001 年にそれぞれ大阪大学基礎工学部中退，同大学院基礎工学研究科博士前期課程修了，同大学院基礎工学研究科博士後期課程修了．2001 年より奈良先端科学技術大学院大学情報科学研究科助手．2004 年 1 月より滋賀大学経済学部情報管理学科講師．2004 年 4 月より現在，滋賀大学経済学部情報管理学科助教授．分散システム，ITS，遺伝的アルゴリズム等の研究に従事．IEEE 会員．

孫 弢（学生会員）

1995 年中国青島大学自動制御学科卒業．2003 年 4 月滋賀大学大学院経営学修士課程修了．2006 年 3 月奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．分散システム，マルチメディア通信システムに関する研究に従事．

玉井 森彦（学生会員）

2002 年岡山県立大学情報工学部情報システム工学科卒業．2007 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．現在，日本学術振興会特別研究員．マルチメディア通信システム，分散処理方式に興味を持つ．

安本 慶一（正会員）

1991 年大阪大学基礎工学部情報工学科卒業．1995 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．1997 年モントリオール大学客員研究員．2002 年より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．ACM，IEEE/CS 各会員．

伊藤 実（正会員）

1977 年，1979 年，1983 年にそれぞれ大阪大学基礎工学部卒業，同大学院基礎工学研究科博士前期課程修了，同大学院基礎工学研究科博士後期課程修了．1979 年より大阪大学基礎工学部助手．1986 年より大阪大学基礎工学部講師．1989 年より大阪大学基礎工学部助教授．1993 年 4 月より現在，奈良先端科学技術大学院大学情報科学研究科教授．関係データベース理論，オブジェクト指向のデータベースのアプリケーション，DNA プローブ等の研究に従事．ACM，IEEE 各会員．

森 將豪（正会員）

1971 年名古屋工業大学工学部電子工学科卒業．1973 年大阪大学大学院修士課程修了．現在，滋賀大学経済学部情報管理学科教授．工学博士．1992 年ブリティッシュコロンビア大学客員研究員．分散システムや，通信プロトコルの検証・テスト等に関する研究に従事．IEEE 会員．