Transcasting: Cost-Efficient Video Multicast for Heterogeneous Mobile Terminals

Morihiko Tamai *, Keiichi Yasumoto *, Naoki Shibata †, Minoru Ito *, Klara Nahrstedt ‡

* Nara Institute of Science and Technology
Ikoma, Nara 630-0192, JAPAN
{morihi-t,yasumoto,ito}@is.naist.jp

† Shiga University
Hikone, Shiga 522-8522, JAPAN
shibata@biwako.shiga-u.ac.jp

‡ University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
klara@cs.uiuc.edu

***Abstract*— This paper presents a cost-efficient video multicast method for live video streaming to heterogeneous mobile terminals over a content delivery network (CDN), where CDN consists of a video server, several proxies with wireless access points, and overlay links among the server and proxies. In this method, the original video sent from the server is converted into multiple versions with various qualities by letting proxies execute transcoding services based on the users' requirements, and delivered to mobile terminals along video delivery paths. To suppress the required computation and transfer costs in CDN, we propose an algorithm to calculate cost-efficient video delivery paths which minimizes the sum of the computation cost for proxies and the transfer cost on overlay links. Our basic idea for deriving cost-efficient delivery paths is to place transcoding service on different proxies in load-balancing manner, and to construct a minimal Steiner tree from all transcoding points of requested qualities. The overall goal of the placement is the balance between computation and transfer cost. Through simulations, we show that our algorithm can calculate more cost-efficient video delivery paths and achieve lower request rejections than other algorithms.**

I. Introduction

There is always a big demand for mobile users to watch live videos of big events such as Olympic games, World Cup soccer games, and major league baseball through their mobile terminals. Recent progress and spread of mobile terminals and fast bandwidth wireless communication technologies have enabled to stream videos to the mobile terminals. Besides it, as an infrastructure to stream videos to a lot of user terminals simultaneously, content delivery network (CDN) such as Akamai [1] is available. However, since mobile terminals are diverse in quality and have different screen sizes, computation powers, battery amounts, and available network bandwidths, it is difficult to stream live video to those diverse mobile terminals efficiently in terms of provider's and user's benefits. For efficient live video streaming to heterogeneous mobile terminals, it is required that (1) each mobile terminal receives live video with the best quality within its capability (in terms of screen size, network bandwidth, and computation power) and in the video format to easily play back the video with as small energy as possible, and (2) resource consumption in CDN is the smallest and no more than the CDN's available capacity.

To realize the above efficient live video streaming to heterogeneous mobile terminals, a lot of research efforts have been made so far, such as simulcasting [2], layered video coding (e.g., MPEG-4 FGS[3]), or MDC (Multiple Description Coding) approaches [4], [5], [6], [7], [8], [9], [10]. In the simulcasting method, several quality versions of a video are generated on-demand on the video server by on-line transcoding so that each quality version of the video is delivered to user terminals through a multicast tree independently of other versions of the video. In this method, the video server may be overloaded due to transcoding of video and transmission of many versions. In the layered video coding or MDC-based method, each user terminal is required to have multiple buffers for simultaneously processing the multiple layers or streams of video data. Thus, it is more costly than playing back single layer/stream video. Moreover, this method can only reduce one parameter of video (e.g., only bit rate) and cannot adjust each of quality parameters (i.e., picture size, frame rate, and bit rate). Therefore, it may not be suitable for applications in diverse mobile terminals.

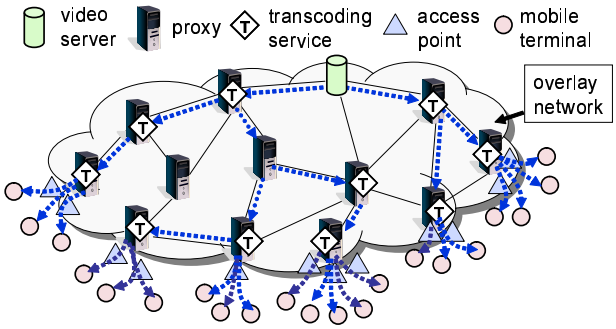


Fig. 1. Content Delivery Network.

In this paper, in order to satisfy the above requirements (1) and (2), we propose a method called *transcasting*, where each CDN proxy executes transcoding and forwarding services to realize an efficient live video multicast to heterogeneous mobile terminals. In the proposed method, we assume that a CDN consists of a video server, proxies with wireless access points, and overlay links between proxies and the server as shown in Fig. 1. We also assume that each mobile terminal joins the CDN by connecting to the closest proxy. In the proposed method, in order to satisfy the above requirements (1) and (2), a video is delivered to mobile terminals by letting proxies transcode a video into various quality versions in real-time according to requests of the mobile terminals. The advantage of this method is that it satisfies the requirement (1), that is, each mobile terminal can receive the video with more suitable quality and play it back with less energy than the layered coding or MDC-based method. For the requirement (2), we have to solve the problem to find the best video delivery paths including transcoding points for each requested quality so that the cost required for video delivery on CDN

is minimized. In this paper, we propose an algorithm where for each requested quality, more than one proxy transcodes the original video to the requested quality version and each proxy delivers the transcoded version to the corresponding mobile terminals through the minimum cost multicast tree. Through computer simulation, we show that our algorithm outperforms other algorithms including simulcasting method in terms of the computation and transfer costs required for video delivery on CDN, and that the algorithm can reduce the number of rejected requests from mobile terminals even when the computation capacities for proxies and bandwidth capacities of overlay links are limited.

This paper is structured as follows. In Sect. II, we briefly explain some studies related to this work. Sect. III describes the target environment and formulates the problem of video multicasting to heterogeneous mobile terminals. We give our proposed algorithm to solve the problem as well as other algorithms in Sect. IV, evaluate performance of the algorithms in Sect. V, and concludes the paper in Sect. VI.

II. Related Work

Some techniques to multicast video to diverse terminals through peer-to-peer network are proposed in [8], [9], [10], [11]. In P2P network, each node of the overlay network corresponds to user terminal. Due to user node's joining and/or leaving the network and degree of network congestion, overlay network topology as well as available bandwidths on overlay links dynamically change. Thus, in order to provide stable video streaming through P2P network, a mechanism for adapting to the fluctuation of overlay network is essential. On the other hand, overlay links in CDN are more stable than that in P2P network, thus the interest in CDN will be how to construct the video delivery paths which minimize the total cost required for video delivery on CDN. The above existing studies focus mainly on the mechanism to absorb fluctuation of the network, but do not consider minimization of required cost.

There are some studies on video multicasting to diverse terminals which take into account minimization of resource consumption at proxies and network [12], [13], [14]. These existing studies suppose that proxies provide transcoding service as well as various services such as text composition and translation of human voice. Services are executed at different proxies, and each user terminal receives the resulting video after applying the original video to a series of these services. Therefore, these studies suppose that each user terminal is given beforehand a *service specification* which describes in what order services should be applied to the video before reaching the terminal. In [12], Wang et, al. have proposed an algorithm to derive video delivery paths which satisfy service specifications and minimize the bandwidth consumption and end-to-end delay in CDN. In [13], [14], algorithms to derive delivery paths minimizing the computation power consumption at proxies as well as network resource consumption in CDN have been proposed. The above existing studies require service specifications in advance, and it would be difficult to find the set of service specifications which minimize resource consumption. Unlike the above existing studies, our proposed method does not need service specifications in advance and can derive the video delivery paths which minimize the total computation and transfer cost required for video delivery on CDN.

In [15], we have presented our preliminary work for the heterogeneous video multicasting with resource minimization in CDN. Our contribution in this paper and the difference from the previous work are as follows: First, we have modified the problem formulation from the previous work so that we can get more gains in saving total required cost by making user's request for video quality discrete. Second, for the modified problem, we have developed a much more sophisticated algorithm to derive delivery paths with less cost. Third, we have conducted thorough evaluation of the proposed algorithm through simulations by comparing it with several general methods including our previous work.

TABLE I
Notations.

Notation	Meaning
s	video server
P	set of proxies
M	set of mobile terminals
L	set of overlay links among the server and proxies, and among proxies themselves
W	set of overlay links between proxies and mobile terminals
$G = (V, E)$	overlay network, where $V = \{s\} \cup P \cup M$ and $E = L \cup W$
$A_{comp}(p)$	computation capacity of $p \in P$
$A_{bw}(u, v)$	bandwidth capacity of $(u, v) \in L$
$C_{link}(u, v)$	link cost per unit of bandwidth of $(u, v) \in L$
q_{orig}	quality of original video
Q	set of all qualities which the users can request
$q.s$, $q.f$, and $q.b$	picture size, frame rate, and bit rate of the quality q
$q(m)$	video quality which $m \in M$ requested
$\mathcal{Q}$	set of video qualities which are required by at least one mobile terminal
$\mathcal{P}(q)$	set of proxies to which mobile terminals requesting quality q are connected
$R(v)$	set of all qualities which $v \in V$ receives along video transcasting tree
$\langle u, q, v, q' \rangle$	a transform relation, where $u, v \in V$ and $q, q' \in Q$
$\mathcal{S}$	set of transform relations
$C_{comp}(q, \hat{q}, \bar{q})$	function to calculate required computation cost for transcoding and forwarding videos from an input quality q, a set of transcoded qualities $\hat{q}$, and a list of forwarded qualities $\bar{q}$
$C_{dec}(q)$	function to calculate required computation cost to decode a video with a quality of q
$C_{enc}(q)$	function to calculate required computation cost to encode a video with a quality of q
$CompCost(\mathcal{S}, p)$	amount of incurred computation cost on $p \in P$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$
$B_{bw}(\mathcal{S}, u, v)$	amount of consumed bandwidth on $(u, v) \in L$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$
$TrferCost(\mathcal{S}, u, v)$	amount of incurred transfer cost on $(u, v) \in L$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$

III. Heterogeneous Video Multicasting Problem

In this section, we present the system model and formulate the problem of transcasting, i.e., cost-efficient video multicasting for heterogeneous mobile terminals. The notations used in this section and subsequent sections are summarized in Table I.

A. Models, Assumptions, and Definitions

1) Network Model: An overlay network which consists of a video server, proxies and mobile terminals is modeled as an undirected graph $G = (V, E)$, where V is the set of nodes and

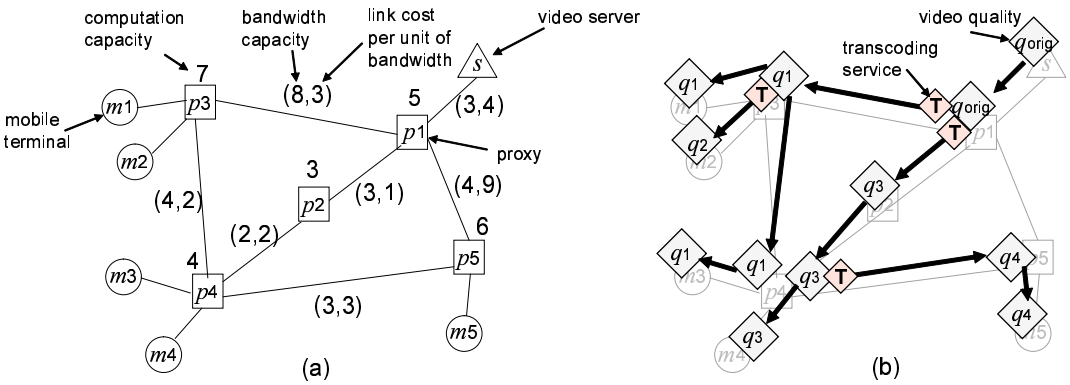


Fig. 2. Example of an overlay network and a video transcasting tree.

E is the set of overlay links among the nodes in V. We denote the video server, the set of proxies, and the set of mobile terminals as s, P and M, respectively. The set of nodes V is $V = \{s\} \cup P \cup M$. We denote the set of overlay links among the video server and proxies, and among the proxies themselves as L. For each mobile terminal $m \in M$, we assume that the nearest proxy $p \in P$ can be determined using a DNS-based server selection scheme [16], and regard that there is an overlay link between m and p. We denote the set of overlay links between mobile terminals and proxies as W. The set of overlay links E is $E = L \cup W$. An example of overlay network is shown in Fig. 2(a). For example, $(s, p_1) \in L$, $(p_1, p_3) \in L$, and $(p_3, m_1) \in W$ hold, in Fig. 2(a).

For each proxy $p \in P$, a fixed amount of computation resources is available. We denote the computation capacity of proxy p as $A_{comp}(p)$. Let (u, v) be an overlay link in L. Then, a fixed amount of bandwidth is allocated on (u, v) by using a network level QoS technique such as DiffServ [17]. We denote the bandwidth capacity of (u, v) as $A_{bw}(u, v)$. We assume that there is sufficiently available bandwidth on each overlay link in W. For each $(u, v) \in L$, a link cost per unit of bandwidth is associated. We denote the link cost per unit of bandwidth of (u, v) as $C_{link}(u, v)$. We assume that a link cost per unit of bandwidth has a positive value.

2) Service Model: The video server provides the video delivery service with transcoding services on proxies for a live video stream such as Olympic game. Before the video server starts a video delivery, it announces the starting time of the video delivery and all video qualities which the users can request. We denote the quality of the original video as q_{orig}. The set of all qualities which the users can request is denoted by Q. Here, q_{orig} is contained in Q (i.e., $q_{orig} \in Q$). We assume that each video quality is decided by three parameters: picture size, frame rate, and bit rate. For each quality $q \in Q$, the picture size, frame rate, and bit rate are denoted by $q.s$, $q.f$ and $q.b$, respectively.

Mobile terminals with diverse capabilities in terms of screen sizes, computation powers, wireless transmission speeds, and (remaining) battery amounts request the video delivery service. Each mobile terminal sends a video delivery request with one of the video qualities in Q to the video server before the pre-scheduled starting time of the video delivery, and receives the video stream through the nearest proxy after the video delivery starts. The quality request made at a mobile terminal $m \in M$ is denoted by $q(m)$, where $q(m) \in Q$.

During the video delivery, each proxy receives video streams from the video server or an upstream proxy, changes the video quality in the streams by using the transcoding services if necessary, and forwards the resulting streams to mobile terminals or a downstream proxy along the video multicast tree. Each proxy can execute multiple transcoding services and each service can handle one stream. For quality $q, q' \in Q$, we denote $q' \leq q$ if and only if $(q'.s \leq q.s) \wedge (q'.f \leq q.f) \wedge (q'.b \leq q.b)$ holds. Since it is impossible to transcode a video from lower quality to higher quality, $q' \leq q$ has to be satisfied when transcoding video from quality q to quality q'.

3) Transform Relation: Video transcasting delivery from the video server to each mobile terminal is realized by finding a video multicast tree connecting the video server and each mobile terminal with input/output video qualities for each proxy. We call this tree a *video transcasting tree*. Fig. 2(b) shows an example of video transcasting tree on the overlay network of Fig. 2(a) when quality requests on mobile terminals are $q(m_1) = q_1$, $q(m_2) = q_2$, $q(m_3) = q_1$, $q(m_4) = q_3$, and $q(m_5) = q_4$.

Let $\mathcal{Q}(\subseteq Q)$ denote the set of video qualities which are required by at least one mobile terminal. For each $q \in \mathcal{Q}$, $\mathcal{P}(q)$ denotes the set of proxies to which mobile terminals requesting quality q are connected. Here, note that $\mathcal{P}(q) \neq \emptyset$. For example, in Fig. 2(b), $\mathcal{P}(q_1) = \{p_3, p_4\}$ holds.

For a node $v \in V$, let $R(v)$ denote the set of all video qualities received by v. For example, $R(p_4) = \{q_1, q_3\}$ holds for proxy p_4 in Fig. 2(b). $R(m) = \{q(m)\}$ holds for each mobile terminal $m \in M$. For convenience, we assume $R(s) = \{q_{orig}\}$ holds. In the video transcasting tree, we say that *transform relation* $\langle u, q, v, q' \rangle$ exists, if and only if for a node $v \in P \cup M$ and a video quality $q' \in R(v)$, a video quality q and a node $u \in \{s\} \cup P$ exist, which satisfy $q' \leq q$, $q \in R(u)$, and $(u, v) \in E$, where u forwards a video of quality q' through the overlay link (u, v). For transform relation $\langle u, q, v, q' \rangle$, if $q \neq q'$ holds, a transcoding service from quality q to q' is executed on u. On the other hand, if $q = q'$, u just forwards the video to v. For example, on transform relation $\langle p_4, q_3, p_5, q_4 \rangle$ in Fig. 2(b), proxy p_4 executes a transcoding service from quality q_3 to q_4. On the other hand, on transform relation $\langle p_3, q_1, p_4, q_1 \rangle$ in Fig. 2(b), proxy p_3 just forwards the video of quality q_1 to proxy p_4. A video transcasting tree can be represented as a set of transform relations. For example, the video transcasting tree shown in Fig. 2(b) consists of 11 transform relations. We denote a set of transform relations by $\mathcal{S}$.

4) Computation Cost: Each proxy incurs some amount of computation cost due to transcoding and forwarding videos. We assume that a function to calculate the required compu-

tation cost for transcoding and forwarding videos from an input video quality, a set of transcoded video qualities, and a list of forwarded video qualities, is given. Let $F(p,q)$ denote the list of forwarded video qualities which is derived from a received video quality $q \in R(p)$ at a proxy $p \in P$. Given a set of transform relations $\mathcal{S}$, a proxy $p \in P$, and a quality $q \in R(p)$, $F(p,q)$ is defined as follows: $F(p,q) = \langle q' \in Q \mid \langle p,q,v,q' \rangle \in \mathcal{S},\ v \in P \cup M \rangle$. For example, $F(p_3,q_1) = \langle q_1, q_2, q_1 \rangle$ in Fig. 2(b). Here, q_1 appears twice in $F(p_3,q_1)$, because it is forwarded to both m_1 and p_4. Similarly, let $T(p,q)$ denote the set of video qualities transcoded from an input quality $q \in R(p)$ at a proxy $p \in P$. $T(p,q)$ is defined as follows: $T(p,q) = \{q' \in F(p,q) \mid q' \neq q\}$. For example, $T(p_3,q_1) = \{q_2\}$ in Fig. 2(b).

The amount of computation cost incurred on a proxy $p \in P$ by transcoding a video of quality q to $|T(p,q)|$ versions with different qualities and by forwarding videos to $|F(p,q)|$ downstream nodes is calculated by $C_{comp}(q, T(p,q), F(p,q))$. Here, C_{comp} is a function to calculate the required computation cost from an input video quality, a set of transcoded video qualities, and a list of forwarded video qualities. $C_{dec}(q)$ denotes a function to calculate the computation cost to decode a video with a quality of q. $C_{enc}(q)$ denotes a function to calculate the computation cost to encode a video with a quality of q. As a result, $C_{comp}(q, T(p,q), F(p,q))$ is defined as the sum of $C_{dec}(q)$, $\sum_{q' \in T(p,q)} C_{enc}(q')$ and the cost for receiving a video with a quality of q and sending videos with qualities of $F(p,q)$. We will look into deeper insight of C_{comp} in Sect. V for experiments.

The amount of incurred computation cost on a proxy $p \in P$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$ is denoted by $CompCost(\mathcal{S}, p)$ and is defined as:

$$CompCost(\mathcal{S}, p) \stackrel{def}{=} \sum_{q \in R(p)} C_{comp}(q, T(p,q), F(p,q)).$$

5) Transfer Cost: Each overlay link incurs some amount of transfer cost due to transferring video streams. Let $D(u,v)$ denote the list of transferred video qualities on an overlay link $(u,v) \in L$. Given a set of transform relations $\mathcal{S}$ and an overlay link $(u,v) \in L$, $D(u,v)$ is defined as follows: $D(u,v) = \langle q' \in Q \mid \langle u,q,v,q' \rangle \in \mathcal{S} \vee \langle v,q,u,q' \rangle \in \mathcal{S},\ q \in Q \rangle$. For example, $D(p_4,p_5) = \langle q_4 \rangle$ in Fig. 2(b).

The amount of consumed bandwidth on an overlay link $(u,v) \in L$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$ is denoted by $B_{bw}(\mathcal{S},u,v)$ and is defined as:

$$B_{bw}(\mathcal{S},u,v) \stackrel{def}{=} \sum_{q \in D(u,v)} q.b.$$

The amount of incurred transfer cost on an overlay link $(u,v) \in L$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$ is denoted by $TrferCost(\mathcal{S},u,v)$ and is defined as:

$$TrferCost(\mathcal{S},u,v) \stackrel{def}{=} C_{link}(u,v) \times B_{bw}(\mathcal{S},u,v).$$

B. Problem Definition

Given an overlay network G and requested quality $q(m) \in Q$ for each mobile terminal $m \in M$, our target problem is to find a set of transform relations $\mathcal{S}$ which satisfies all of the following conditions (1)–(4):

$$\text{for each } \langle u,q,v,q' \rangle \in \mathcal{S},\ (u,v) \in E \wedge q' \leq q. \tag{1}$$

$$\begin{aligned} &\text{for each } m \in M, \\ &\exists p_0, p_1, \ldots, p_k \in P,\ \exists q_0, q_1, \ldots, q_k \in Q \ \ s.t. \\ &\langle s, q_{orig}, p_0, q_0 \rangle, \langle p_0, q_0, p_1, q_1 \rangle, \ldots \\ &\ldots, \langle p_k, q_k, m, q(m) \rangle \in \mathcal{S}. \end{aligned} \tag{2}$$

$$\text{for each } p \in P,\ CompCost(\mathcal{S}, p) \leq A_{comp}(p). \tag{3}$$

$$\text{for each } (u,v) \in L,\ B_{bw}(\mathcal{S},u,v) \leq A_{bw}(u,v). \tag{4}$$

The condition (1) means that all transform relations in $\mathcal{S}$ have to go through an overlay link in E, and they have to be involved in either just forwarding of a stream or transcoding to a lower quality and forwarding the resulting stream. The condition (2) means that for each $m \in M$, $\mathcal{S}$ has to contain a feasible route from the video server s to m. The condition (3) and (4) are the constraints on computation capacity of each proxy and bandwidth capacity of each overlay link.

Generally speaking, there are more than one solutions for $\mathcal{S}$ which satisfy all of the above conditions. In this paper, we consider the following problem to find a video transcasting tree which reduces the total amount of incurred cost on the overlay network:

$$\begin{aligned} \textbf{minimize} \quad & \sum_{p \in P} CompCost(\mathcal{S}, p) \\ & + \sum_{(u,v) \in L} TrferCost(\mathcal{S},u,v) \\ \textbf{subject to} \quad & \text{constraints (1)–(4).} \end{aligned} \tag{5}$$

The objective function in Equation (5) is the sum of all computation cost incurred on each proxy $p \in P$ and transfer cost incurred on each overlay link $(u,v) \in L$ when the video delivery is realized based on a set of transform relations $\mathcal{S}$.

IV. Transcasting Tree Construction Algorithms

In this section, first we briefly address the Steiner tree problem, and then give our algorithms based on the Steiner tree to solve the problem formulated in Sect. III-B.

A. Steiner Tree Problem

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ and a subset of vertices $\mathcal{D} \subseteq \mathcal{V}$, where $\mathcal{V}$, $\mathcal{E}$, and c are a set of vertices, a set of edges, and a cost function mapping from $\mathcal{E}$ to $\mathbf{N}^+$, respectively, Steiner tree problem is the problem to find a tree with the minimum total cost among all trees in $\mathcal{G}$ each of which includes all vertices of $\mathcal{D}$. The tree is called the *minimal Steiner tree*. This problem is known as NP-hard problem [18], and several heuristic algorithms are proposed to compute approximated solutions (e.g., [19]). Our transcasting problem described in Sect. III-B implies the Steiner tree problem, and we have the following theorem.

Theorem 1: The transcasting problem is NP-hard.

Proof: We show that the transcasting problem is NP-hard by reducing the Steiner tree problem to it. For both the Steiner tree problem and the transcasting problem, we consider the corresponding decision problems which ask if there is a (Steiner or transcasting) tree which has a cost equal to or less than a given integer k. Given an instance of the Steiner tree problem (i.e., an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$, a subset of vertices $\mathcal{D} \subseteq \mathcal{V}$, and an integer k), we construct an instance of the transcasting problem as follows. We select an arbitrary node from $\mathcal{D}$, and regard it as the video server s. The set of proxies P and the set of overlay links L are set to $P = \mathcal{V} - \{s\}$

and $L = \mathcal{E}$, respectively. We set the computation capacity of each proxy $p \in P$ to infinity. For each overlay link $(u, v) \in L$, we set the bandwidth capacity to infinity, and set the link cost per unit of bandwidth to be $C_{link}(u, v) = c((u, v))$. The set of all qualities Q is $Q = \{q_{orig}\}$ where $q_{orig}.b = 1$. For each proxy $p \in P$, if $p \in \mathcal{D}$, then we assume that only one mobile terminal which requests q_{orig} is connected to p. Otherwise, we assume that no mobile terminals are connected to p. The above construction can be done in polynomial time.

In the Steiner tree problem, if there is a Steiner tree of cost k or less, we can construct a corresponding transcasting tree of cost k or less by connecting s and all proxies in $\mathcal{P}(q_{orig})$ along the Steiner tree. Conversely, in the transcasting problem, if there is a transcasting tree of cost k or less, we can construct a corresponding Steiner tree of cost k or less based on the set of transform relations which represents the transcasting tree. This shows that the transcasting problem is NP-hard. ■

In the following subsections, we propose two heuristic algorithms to solve the transcasting problem: *two-layered tree algorithm* and its extension *divided tree algorithm*.

B. Two-layered Tree Algorithm

In simulcasting [2], for each quality $q \in \mathcal{Q}$, a minimal Steiner tree containing a video server s as its root and all proxies in $\mathcal{P}(q)$ is constructed so that the video is delivered through the tree. In this method, however, the video server has to transcode the video to $|\mathcal{Q}|$ versions with different qualities. So, the server may be overloaded when $|\mathcal{Q}|$ is large. Similarly to simulcasting, the two-layered tree algorithm proposed below constructs a minimal Steiner tree for each quality $q \in \mathcal{Q}$, but it assigns the roots of the Steiner trees for $\mathcal{Q}$ to different proxies, aiming at load distribution of transcoding points. Those assigned proxies are called *transcoding proxies*, hereafter. Each transcoding proxy transcodes the original video to the version with quality q by executing a transcoding service. Two-layered tree algorithm constructs a video transcasting tree in the following two hierarchies: *first-level Steiner tree* which is the minimal Steiner tree including the video server s and all the transcoding proxies, and *second-level Steiner tree* which is the minimal Steiner tree constructed for each quality $q \in \mathcal{Q}$ including a transcoding proxy as its root and all proxies in $\mathcal{P}(q)$. The details on construction of these trees are shown below.

Step 1 (Construction of second-level Steiner tree): Let $q_1, q_2, ...,$ and $q_{|\mathcal{Q}|}$ denote items of $\mathcal{Q}$ in increasing order of their bit rates.
For $i = 1$ to $|\mathcal{Q}|$, carry out the following sub-steps.

Step 1-1: Compute the minimal Steiner tree including all proxies of $\mathcal{P}(q_i)$ over the CDN (the root of the Steiner tree, i.e., the transcoding proxy, is decided later in Step 2). Here, in order to consider bandwidth constraint, we remove overlay links whose bandwidth capacities are less than $q_i.b$ before computation. If some proxies in $\mathcal{P}(q_i)$ are excluded from the tree, the requests of mobile terminals connecting to the proxies are rejected.

Step 1-2: $q_i.b$ units of bandwidth are subtracted from the bandwidth capacity of each overlay link contained in the calculated Steiner tree.

Step 2 (Decision of transcoding proxy): Let $q_1, q_2, ...,$ and $q_{|\mathcal{Q}|}$ denote items of $\mathcal{Q}$ in increasing order of their bit rates.
For $i = 1$ to $|\mathcal{Q}|$, carry out the following sub-steps.

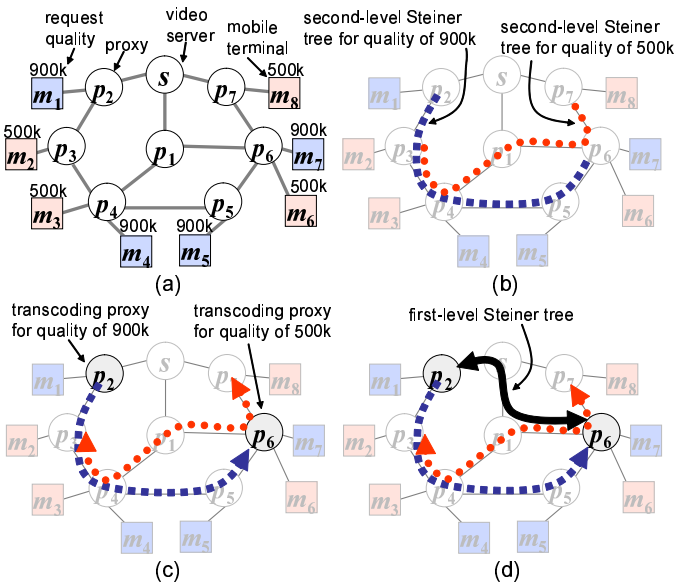


Fig. 3. Construction of video transcasting tree using two-layered tree algorithm.

Step 2-1: For each proxy p in $\mathcal{P}(q_i)$, if it satisfies the following three conditions, then p is regarded as a candidate of a transcoding proxy for quality q_i: (1) p is not excluded from the Steiner tree derived in Step 1-1; (2) there is a path between p and s where each overlay link in the path has a bandwidth capacity no less than $q_{orig}.b$; and (3) the computation cost required for transcoding a video of quality q_{orig} to the version with quality q_i does not exceed the computation capacity of p.
If there are multiple candidates, then one is selected at random for the transcoding proxy of quality q_i. If there is no candidate, the transcoding proxy for q_i is not decided.

Step 2-2: if a transcoding proxy is decided, then the computation cost required for transcoding from quality q_{orig} to q_i is subtracted from the computation capacity of the proxy. If a transcoding proxy is not decided in Step 2-1, mobile terminals requesting q_i are rejected, and $q_i.b$ units of bandwidth are added to each overlay link of the Steiner tree derived in Step 1-1.

Step 3 (Construction of first-level Steiner tree): Compute the minimal Steiner tree including a video server s and all transcoding proxies.

In Fig. 3, we show an example of transcasting tree construction by the two-layered tree algorithm. Fig. 3 (a) depicts the topology of an overlay network (CDN) with requested qualities of mobile terminals[1]. Fig. 3(b) depicts the situation just after the second-level Steiner trees for requested qualities are constructed (i.e., after Step 1). Fig. 3(c) shows the situation after a

[1]In the figure, each requested quality is shown only by bit rate for space limitation, but actually given by a tuple of picture size, frame rate, and bit rate.

transcoding proxy is assigned to the second-level Steiner tree for each quality (i.e., after Step 2). Finally, Fig. 3(d) shows the situation after the first-level Steiner tree is constructed.

C. Divided Tree Algorithm

In two-layered tree algorithm, for each quality $q \in \mathcal{Q}$, only one transcoding proxy is assigned in q's Steiner tree and a version of quality q is generated at the proxy and transferred to other proxies (and to mobile terminals) through the tree. In this method, the number of transcoding services is small, thus the totally required computation cost is also small for video delivery. On the other hand, since the second-level Steiner tree for each quality q generated by this algorithm contains all proxies of $\mathcal{P}(q)$, the tree size tends to be large and may incur large amount of transfer cost in CDN. For this problem, we propose the divided tree algorithm below by extending the two-layered tree algorithm.

In the divided tree algorithm, we divide second-level Steiner tree for each $q \in \mathcal{Q}$ to multiple sub-trees by removing some overlay links included in the tree. Then, we assign a transcoding proxy to each of the divided sub-trees. This algorithm increases the computation cost, but decreases bandwidth consumption on the removed links and thus can decrease the total transfer cost. The problem is how many sub-trees each second-level Steiner tree should be divided to, to minimize the overall computation and transfer cost.

In the divided tree algorithm, for each second-level Steiner tree of quality $q \in \mathcal{Q}$, the highest cost's overlay link is picked up and the transfer cost through the link is compared with the computation cost for executing an additional encoder for the quality q. If the transfer cost incurred on the link is larger than the computation cost incurred by the additional encoder, the link is removed so that the second-level Steiner tree is divided to two sub-trees. For the divided second-level Steiner tree, the same procedure is repeated. If there is no overlay link whose transfer cost is larger than the computation cost of adding an encoder, then tree division ends. The detailed procedure is shown below. Note that the following procedure is executed just after Step 1-1 of the two-layered tree algorithm.

Step 1-1' (Division of second-level Steiner tree): Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph representing the second-level Steiner tree for quality q, where $\mathcal{V} \subseteq \{s\} \cup P$ and $\mathcal{E} \subseteq L$ are the set of nodes and the set of edges, respectively, which are included in the second-level Steiner tree for quality q. Carry out the following sub-steps.

Step A: If there are more than one edge in $\mathcal{G}$, then select the edge (u, v) with highest cost (details of selection method are explained later). Otherwise, finish division of the current tree.

Step B: Evaluate if the following inequality holds for the selected edge (u, v):

$$C_{enc}(q) < C_{trfer}(q, u, v), \tag{6}$$

where $C_{trfer}(q, u, v) \stackrel{def}{=} q.b \times C_{link}(u, v)$.

Step C: If the inequality (6) holds, edge (u, v) is removed from $\mathcal{G}$ and go to Step A. Otherwise, finish division of the current tree.

In the divided tree algorithm, a transcoding proxy is assigned to each sub-tree. So, Step 2-1 and Step 2-2 of the two-layered tree algorithm are applied for each of the divided sub-trees.

We will explain how to select the highest cost's edge in Step A through an example shown in Fig. 4. Fig. 4 (a) depicts

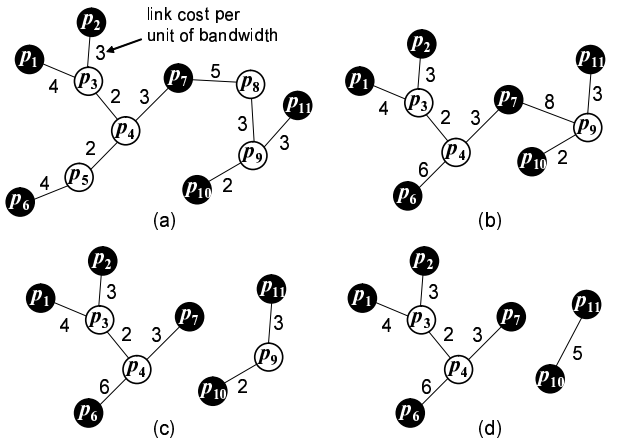


Fig. 4. An example of the process of merging edges.

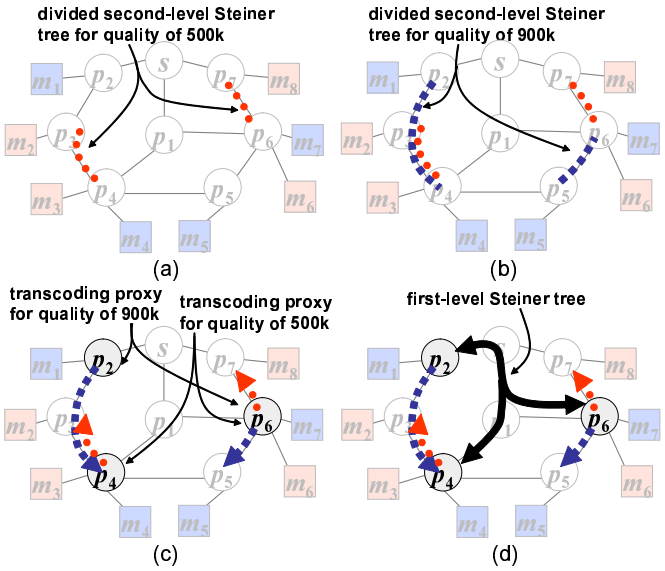


Fig. 5. Construction of video transcasting tree using divided tree algorithm.

an instance of the second-level Steiner tree. Here, black nodes are proxies to which mobile terminals requesting quality $q \in \mathcal{Q}$ are connected (i.e, the proxies belong to the set $\mathcal{P}(q)$). White nodes are proxies to which no mobile terminal requesting q is connected. Given this situation, first the highest cost's edge (p_7, p_8) may be removed in Step A. However, when doing so, proxy p_8 will receive the video version with q, although no mobile terminal connected to p_8 requests quality q. This means the wastage of bandwidth. To avoid such a situation, we apply the following pre-processing to the second-level Steiner tree for q before Step A: if a white node has exactly two edges, we merge those two edges to one edge and remove the white node so that the merged edge has the sum of the link costs of the two edges as its cost. For example, if we apply the above pre-processing to the graph in Fig. 4(a), the new graph shown in Fig. 4(b) is derived. In this example, edges (p_4, p_5) and (p_5, p_6), edges (p_7, p_8) and (p_8, p_9) are merged, respectively. Fig. 4(c) shows the situation that the link (p_7, p_9) which has the largest cost is removed from Fig. 4(b). When a link is removed, some edges which can be merged may appear (e.g., edge (p_9, p_{10}) and (p_9, p_{11})) as shown in Fig. 4(c). In that

case, we further merge those edges as shown in Fig. 4(d).

An example of transcasting tree construction by the divided tree algorithm is shown in Fig. 5. Here, note that the topology of the overlay network and qualities requested by mobile terminals are the same as in Fig. 3(a). Also note that the second-level Steiner tree for each quality before division is the same as in Fig. 3(b). Fig. 5(a) depicts the sub-trees generated by dividing the second-level Steiner tree for quality 500 Kbps. In this example, overlay links between proxies p_4 and p_6 are removed. Fig. 5(b) depicts the sub-trees generated by dividing the second-level Steiner tree for quality 900 Kbps. In this example, the overlay link between proxies p_4 and p_5 is removed. Fig. 5(c) shows the situation after a transcoding proxy is assigned to each sub-tree. Finally, Fig. 5(d) shows the situation after the first-level Steiner tree is constructed. Note that in Fig. 5(d), we can consider further optimization by letting the proxy p_1 or the server node s to execute a transcoding service from the original quality to 900 Kbps instead of p_6, and forward 900 Kbps of video to p_6. If the bit rate of the original video is much higher than 900 Kbps, we can reduce large amount of transfer cost incurred on the overlay link between p_1 and p_6. However, in our target environment, each proxy has many mobile terminals, and some of the terminals always request video with highest quality. In such a situation, further optimization of first-level Steiner tree is not required.

V. Evaluation

In this section, we give evaluation results on the transcasting approach using our divided tree algorithm through computer simulations. We have used the following four algorithms to construct video transcasting tree and compared performance among them.

(1) *simulcasting*: it transfers video through $|\mathcal{Q}|$ Steiner trees whose roots are at the video server s, and all transcoding services for $\mathcal{Q}$ are executed at s. Each Steiner tree for a quality $q \in \mathcal{Q}$ is constructed in increasing order of their bit rates. (2) *two-layered tree*: it transfers video through the video transcasting tree constructed by the two-layered tree algorithm explained in Sect. IV-B. (3) *divided tree*: it transfers video through the video transcasting tree constructed by the divided tree algorithm explained in Sect. IV-C. (4) *full divided tree* [15]: it transfers video through completely divided sub-trees for each quality $q \in \mathcal{Q}$ which are generated by dividing the second-level Steiner tree for q to sub-trees so that each sub-tree contains only one proxy with mobile terminals requesting q.

In the experiment, we have evaluated the algorithms by the following metrics: (i) Total required computation and transfer costs when there is no limitation on computation or bandwidth capacities in CDN; (ii) Required computation and transfer costs per mobile terminal when there is limitation on either computation capacity or transfer capacity, or both capacities in CDN, and the number of rejected requests; and (iii) computation time to construct video transcasting tree. We have implemented a heuristic algorithm in [19] and used it to construct the minimal Steiner tree in each of the above algorithms.

A. Experimental Setup

In the experiment, we use the BRITE topology generator [20] to generate AS-level network topology based on Waxman model for CDN. In the generated topology, we select one node as a video server, and assign other nodes as proxies. Each mobile terminal connects to a proxy selected at random. In the following experiments, we assume that the number of proxies is 100. The quality of the original video is 640×480 pixels for picture size, 30fps for frame rate, and 1.5 Mbps for bit rate. We consider 5 quality levels for each quality parameter. That is, each mobile terminal can request the video quality for picture size among 640×480, 576×432, 505×379, 423×317, and 320×240, for frame rate among 30 fps, 25 fps, 20 fps, 15 fps, and 10 fps, and for bit rate among 1.5 Mbps, 1.2 Mbps, 0.94 Mbps, 0.66 Mbps, and 0.38 Mbps (i.e., $|Q| = 125$). Each mobile terminal selects one value among 5 quality levels for each quality parameter at random, and requests video with the decided quality. We specify the link cost for each overlay link per unit of bandwidth (Mbps) between 1 and 5 at random.

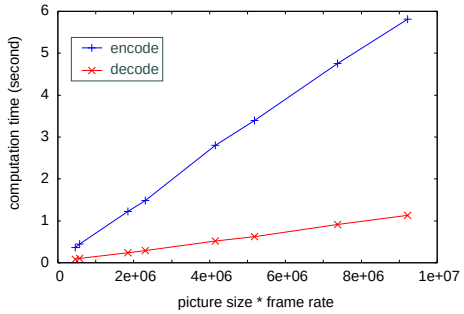


Fig. 6. Computation time to decode/encode MPEG-1 video with length of 60 seconds.

B. Computation Cost for Transcoding

In this subsection, we show an example of the function C_{comp} described in Sect. III-A. In general, C_{comp} consists of parts to calculate required computation costs for decoding an input quality, for encoding a set of output qualities, and for processing packets of video streams. In the experiment, we assume that transcoding of a video is conducted by decoding the received video stream, dropping some frames, resizing the surviving frames, and re-encoding the frames. We approximate computation cost required for transcoding to be the sum of decoding and encoding by considering that costs for dropping and resizing of frames are small enough to be ignored.

We have conducted an experiment to investigate the computation cost for transcoding service. For the purpose, we have investigated the relationship between the number of pixels per second in a video (i.e., product of number of pixels in a picture frame and frame rate) and time taken by decoding and encoding of the video. In the experiment, we used a desktop PC (Intel Core2 Duo E6600 (2.4GHz), 2GB RAM, Linux 2.6.18.1), and measured decoding and encoding time with FFmpeg [21] for some MPEG-1 videos whose lengths are 60 seconds. Experimental result is shown in Fig. 6. Fig. 6 suggests that time taken for decoding and encoding is almost linear to the number of pixels per second in the video. According to the result in this experiment, we assume that the computation cost required for transcoding a video is linear to the number of pixels per second in the video. Based on this assumption, the amounts of computation cost required for decoding and encoding video are defined as follows: $C_{dec}(q) = \tau_d\,(q.s \times q.f)$ for decoding a video of a quality q, $C_{enc}(q') = \tau_e\,(q'.s \times q'.f)$ for encoding a video of a quality q'. Here, τ_d and τ_e denote the computation costs required to process one pixel per second to decode and encode a video, respectively. In order to calculate computation cost, we specify

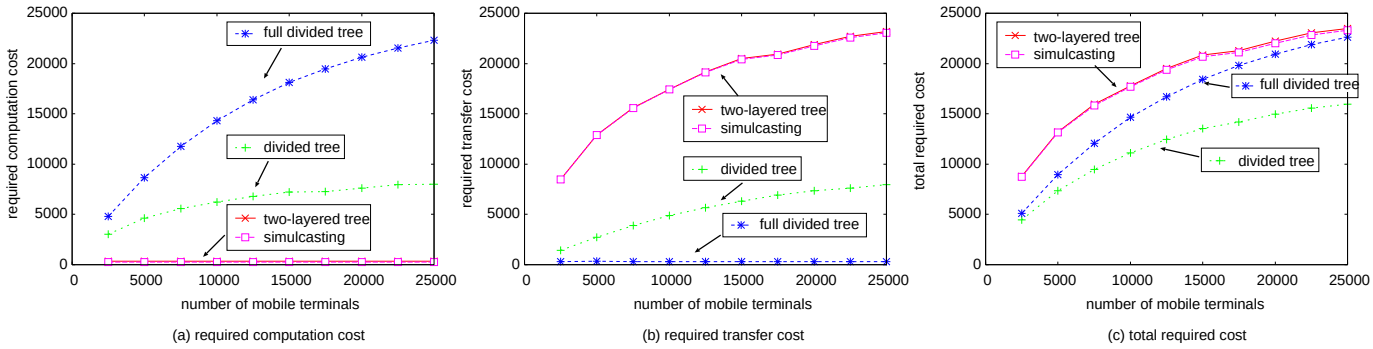


Fig. 7. Required cost vs. number of mobile terminals (no limitations on computation or bandwidth capacities).

the values for τ_d and τ_e as follows: $\tau_d = 1/(q_{orig}.s \times q_{orig}.f)$ and $\tau_e = 5\tau_d$. Here, we specify the value of τ_d so that the computation cost for decoding the original video will be 1. Also we specify the value of τ_e based on the ratio between the time for decoding and encoding in Fig. 6. In the following experiments, we assume that required computation cost on proxies for processing packets of video streams can be ignored.

C. Evaluation Results

1) Required Cost for Video Delivery under No Limitations on Computation or Bandwidth Capacities: We compare the above algorithms in terms of the required computation and transfer costs when there is no limitation on computation or bandwidth capacities in CDN. We show the results in Fig. 7(a)–(c). Horizontal axis of each figure represents the number of mobile terminals. Vertical axes of Fig. 7(a), Fig. 7(b), and Fig. 7(c) represent required computation cost, required transfer cost, and total required cost (sum of the required computation and transfer costs), respectively.

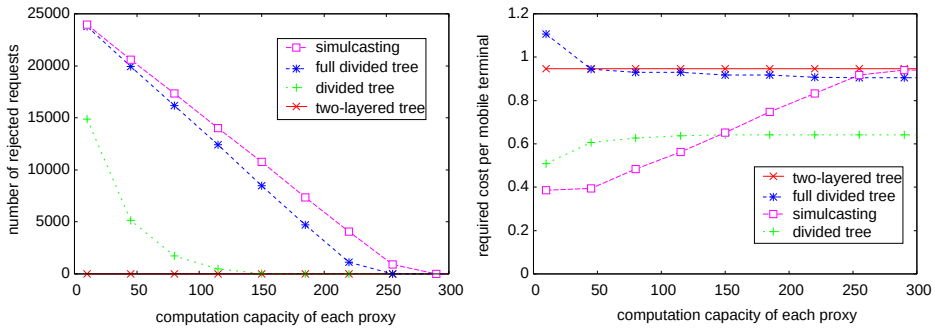
Fig. 7(a) shows that full divided tree takes more computation cost than other algorithms. On the other hand, Fig. 7(b) shows that simulcasting and two-layered tree take more transfer cost than other algorithms. In both figures, we can see that required cost in divided tree falls in the middle between full divided tree, and simulcasting and two-layered trees. As a result, in Fig. 7(c), we can see that divided tree takes less total cost than other algorithms, and its gain over the other algorithms increases as the number of mobile terminals increases.

2) Number of Rejected Requests from Mobile Terminals and Required Cost Per Mobile Terminal for Video Delivery under Limitations on Computation and Bandwidth Capacities: We compare the four algorithms with respect to the number of rejected requests from mobile terminals and required cost per mobile terminal when there is limitation on computation capacity of each proxy and bandwidth capacity of each overlay link. For simulcasting, the video server has the same computation capacity as one proxy. In the following experiments, we set the number of mobile terminals to 25,000.

First, we show the results when there is a limitation on computation capacity of each proxy (without limitation on bandwidth capacity of each overlay link). Fig. 8(a) and Fig. 8(b) show the number of rejected requests from mobile terminals and required cost per mobile terminal, respectively. Here, a required cost per mobile terminal is calculated by dividing the required total cost in overlay network by the number of accepted requests. Horizontal axis of these figures represents a computation capacity of each proxy. For example, when this value is 150, it means that each proxy has enough computation capacity to be able to decode 150 videos with the qualities of the original video, simultaneously.

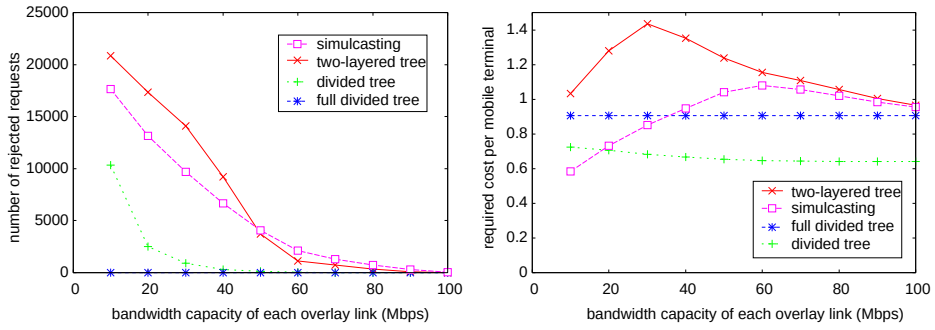
Fig. 8(a) shows that simulcasting and full divided tree have larger number of rejected requests than two-layered tree and divided tree. According to the figure, divided tree has larger number of rejected requests than two-layered tree when computation capacity of each proxy is less than 150. The reason is that divided tree algorithm assigns a transcoding proxy to each sub-tree in increasing order of bit rate of video quality and for some sub-trees for higher quality, transcoding proxies may not be assigned due to the computation capacity of each proxy, which results in rejected requests of mobile terminals. However, Fig. 8(b) shows that divided tree greatly reduces required cost per mobile terminal than two-layered tree. Thus, when the numbers of rejected requests in divided tree and two-layered tree are almost the same (i.e., when computation capacity of each proxy is more than 100 in Fig. 8(a)), divided tree can derive video transcasting tree with smaller cost than two-layered tree. We note that, in Fig. 8(b), required cost per mobile terminal of simulcasting increases as computation capacity of each proxy increases. The reason is that each Steiner tree for a requested quality is constructed in increasing order of the bit rate of the quality. Thus, in simulcasting, when computation capacity of each proxy increases (i.e., the computation capacity of the video server increases), the video server can execute transcoding services for the requested qualities with higher bit rates, and they will incur more transfer cost on overlay links.

Next, we show the results when there is a limitation on bandwidth capacity of each overlay link (without limitation on computation capacity of each proxy). Fig. 9(a) and Fig. 9(b) show the number of rejected requests and required cost per mobile terminal, respectively. Horizontal axis of these figures represent constraint on bandwidth capacity of each overlay link in Mbps. Fig. 9(a) suggests that full divided tree and divided tree have smaller number of rejected requests than simulcasting and two-layered tree even when bandwidth capacity is small (e.g., less than 30 Mbps). Two layered tree has larger number of rejected requests than simulcasting when bandwidth capacity is less than 50 Mbps. The reason is that in two-layered tree, large amount of bandwidth is consumed by second-level Steiner trees and many requests may be rejected due to bandwidth shortage for transferring original quality video from a server node to proxies of root nodes of second-level Steiner trees, since the first-level Steiner tree is constructed after all second-level trees are constructed. Fig. 9(b) shows that divided tree takes smaller required cost per mobile terminal



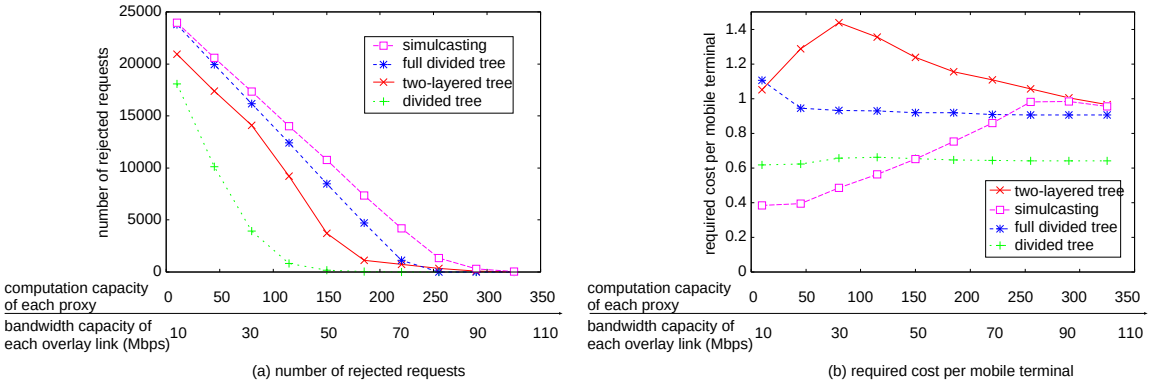
(a) number of rejected requests (b) required cost per mobile terminal

Fig. 8. Number of rejected requests and required cost per mobile terminal vs. computation capacity of each proxy (no limitation on bandwidth capacity).



(a) number of rejected requests (b) required cost per mobile terminal

Fig. 9. Number of rejected requests and required cost per mobile terminal vs. bandwidth capacity of each overlay link (no limitation on computation capacity).



(a) number of rejected requests (b) required cost per mobile terminal

Fig. 10. Number of rejected requests and required cost per mobile terminal vs. computation capacity of each proxy and bandwidth capacity of each overlay link.

than full divided tree. Thus, divided tree can derive more cost-efficient video transcasting tree than full divided tree when the numbers of rejected requests in these algorithms are the same (i.e., when bandwidth capacity is more than 30 Mbps in Fig. 9(a)). In two-layered tree and simulcasting, required cost per mobile terminal decreases when bandwidth capacity is more than 30 Mbps and 60 Mbps, respectively. The reason is that when bandwidth capacity is large, the Steiner tree with the smaller transfer cost can be found.

Finally, we show the results when there is a limitation on computation capacity of each proxy and bandwidth capacity of each overlay link. Fig. 10(a) and Fig. 10(b) show the number of rejected requests and required cost per mobile terminal, respectively. The horizontal axis of these figures represent both computation capacity and bandwidth capacity. Fig. 10(a) shows that divided tree has smaller number of rejected requests than other algorithms. Fig. 10(b) shows that as computation capacity and bandwidth capacity reach certain threshold, divided tree takes less required cost per mobile terminal than simulcasting.

As a result, it is shown that divided tree can derive more cost-efficient video transcasting tree and achieve smaller number of rejected requests than other algorithms in wide range of CDNs which have different constraints on computation and bandwidth capacities.

In the above experiments, we specified a link cost per unit of bandwidth for each overlay link as a random number in the range of [1:5] and configured one unit of computation cost as the computation cost required to decode the original video. In general, these configurations on link cost and computation cost depend on CDN, and different configurations may be used for each CDN. Thus, some configurations may exist where two-layered tree or full divided tree is more cost-efficient than divided tree. However, as described in Sect. IV-C, divided

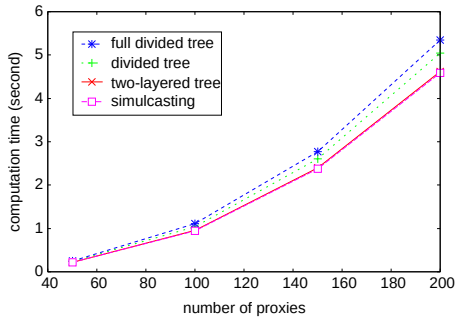


Fig. 11. Computation time for deriving video transcasting tree.

tree algorithm decides whether to divide each second-level Steiner tree further or not, taking into account the balance between transfer cost and computation cost. Thus, divided tree algorithm can derive a video transcasting tree similar to the two-layered tree when computation cost is much higher than link cost, and derive a video transcasting tree similar to the full divided tree when link cost is much higher than computation cost. Consequently, divided tree algorithm can derive cost-efficient video transcasting tree in wide range of configurations of CDN than other algorithms.

3) Computation Time for Transcasting Tree Construction: We have measured computation time of the four algorithms for deriving video transcasting tree. In this experiment, we used the following desktop PC: Intel Core2 Duo E6600 (2.4GHz), 2GB RAM, Linux 2.6.18.1. We supposed that there is no limitation on computation capacity or bandwidth capacity. The number of mobile terminals was 25,000.

The experimental result is shown in Fig. 11. The horizontal axis of the figure represents the number of proxies in the overlay network. Fig. 11 suggests that there is almost no difference of computation time among the four algorithms. The reason is that most time consuming part of each algorithm is the calculation of the minimal Steiner trees, and that time for dividing each tree is relatively small. According to the figure, when the number of proxies is 200, each mobile terminal has to send its request about 6 seconds before the video broadcast starts. We believe that this is practical enough.

In order to treat mobile terminals which request a video after its broadcast starts, the divided tree algorithm may be executed periodically to optimize the video transcasting tree accommodating new requests from those terminals. If the reconstruction period of the video transcasting tree is 30 minutes, the computation time is about 1% of the period, so the overhead by this computation time and path establishment time seems small enough for practical use.

VI. Conclusions

In this paper, we proposed transcasting, a cost-efficient video multicast method for heterogeneous mobile terminals with different quality requests. In the proposed transcasting method, a minimum cost multicast tree is constructed for each requested quality, and the original video is transcoded at proxies to a version with each of requested quality and the version is delivered through the corresponding multicast tree. We have developed an algorithm where a multicast tree of each quality is divided to appropriate number of sub-trees so that total required cost for video delivery is minimized taking into account balance between required computation cost and required transfer cost.

Through simulations, we confirmed that the proposed divided tree algorithm outperforms simulcasting method [2], in which a server transcodes the original video to multiple versions with different qualities, and other method [15].

References

[1] Akamai home page, http://www.akamai.com/.
[2] S. Y. Cheung, M. H. Ammar, and X. Li, On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution, *Proc. of the 15th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'96)*, pp. 553–560, 1996.
[3] W. Li, Overview of Fine Granularity Scalability in MPEG-4 Video Standard, *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 11, No. 3, pp. 301–317, 2001.
[4] S. McCanne, V. Jacobson, and M. Vetterli, Receiver-driven Layered Multicast, *Proc. of ACM SIGCOMM'96*, pp. 117–130, 1996.
[5] B. J. Vickers, C. Albuquerque, and T. Suda, Source Adaptive Multi-Layered Multicast Algorithms for Real-Time Video Distribution, *IEEE/ACM Trans. on Networking (TON)*, Vol. 8, No. 6, pp. 720–733, 2000.
[6] T. Kim and M. H. Ammar, A Comparison of Layering and Stream Replication Video Multicast Schemes, *Proc. of the 11th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, pp. 63–72, 2001.
[7] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, On Multiple Description Streaming with Content Delivery Networks, *Proc. of the 21st Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'02)*, pp. 1736–1745, 2002.
[8] Y. Cui and K. Nahrstedt, Layered Peer-to-Peer Streaming, *Proc. of the 13th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03)*, pp. 162–171, 2003.
[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, Distributing 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'02)*, pp. 177–186, 2002.
[10] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, Resilient Peer-to-Peer Streaming, *Proc. of the 11th IEEE Int'l Conf. on Network Protocols (ICNP'03)*, pp. 16–27, 2003.
[11] T. Sun, M. Tamai, K. Yasumoto, N. Shibata, M. Ito, and M. Mori, MT-cast: Robust and Efficient P2P-based Video Delivery for Heterogeneous Users, *Proc. of the 9th Int'l Conf. on Principles of Distributed Systems (OPODIS'05)*, pp. 176-190, 2005.
[12] M. Wang, B. Li, and Z. Li, sFlow: Towards Resource-Efficient and Agile Service Federation in Service Overlay Networks, *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pp. 628–635, 2004.
[13] X. Gu and K. Nahrstedt, Distributed Multimedia Service Composition with Statistical QoS Assurances, *IEEE Trans. on Multimedia*, Vol. 8, No. 1, pp. 141–151, 2006.
[14] J. Jin and K. Nahrstedt, Service Composition for Generic Service Graphs, *ACM Multimedia Systems Journal*, Vol. 11, No. 6, pp. 568–581, 2006.
[15] S. Yamaoka, T. Sun, M. Tamai, K. Yasumoto, N. Shibata, and M. Ito, Resource-Aware Service Composition for Video Multicast to Heterogeneous Mobile Users, *Proc. of the 1st ACM Int'l Workshop on Multimedia Service Composition (MSC'05) (ACM Multimedia 2005 Workshop)*, pp. 37–46, 2005.
[16] A. Shaikh, R. Tewari, and M. Agrawal, On the Effectiveness of DNS-based Server Selection, *Proc. of the 20th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'01)*, pp. 1801–1810, 2001.
[17] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, An Architecture for Differentiated Services, *IETF RFC 2475*, 1998.
[18] R. M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations (R. E. Miller and J. W. Thatcher eds.)*, Plenum Press, pp. 85–103, 1972.
[19] L. Kou, G. Markowsky, and L. Berman, A Fast Algorithm for Steiner Trees, *Acta Informatica*, Vol. 15, No. 2, pp. 141–145, 1981.
[20] A. Medina, A. Lakhina, I. Matta, and J. Byers, BRITE: An Approach to Universal Topology Generation, *Proc. of the 9th Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, pp. 346–353, 2001.
[21] FFmpeg, http://ffmpeg.mplayerhq.hu/.