IEICE TRANS. INF. & SYST., VOL.E102–D, NO.11 NOVEMBER 2019

PAPER

# Peer-to-Peer Video Streaming of Non-Uniform Bitrate with Guaranteed Delivery Hops*

**Satoshi FUJITA**[†a)], ***Member***

**SUMMARY** In conventional video streaming systems, various kind of video streams are delivered from a dedicated server (e.g., edge server) to the subscribers so that a video stream of higher quality level is encoded with a higher bitrate. In this paper, we consider the problem of delivering those video streams with the assistance of Peer-to-Peer (P2P) technology with as small server cost as possible while keeping the performance of video streaming in terms of the throughput and the latency. The basic idea of the proposed method is to divide a given video stream into several sub-streams called stripes as evenly as possible and to deliver those stripes to the subscribers through different tree-structured overlays. Such a stripe-based approach could average the load of peers, and could effectively resolve the overloading of the overlay for high quality video streams. The performance of the proposed method is evaluated numerically. The result of evaluations indicates that the proposed method significantly reduces the server cost necessary to guarantee a designated delivery hops, compared with a naive tree-based scheme.

***key words:*** *P2P video streaming, tree-structured overlay, guaranteed delivery hops, quality level*

## 1. Introduction

Video streaming over the Internet has attracted considerable attention in recent years. In fact, IP video traffic occupies 73% of the worldwide consumer Internet traffic in 2016, and is expected to exceed 80% by 2021 [8]. Such *video streams* are delivered from the publisher to the subscribers in various manners. Many companies and organizations including YouTube, ESPN and BBC, adopt Content Delivery Network (CDN) such as Akamai[**], Azure CDN[***], and Verizon Digital Media Services[****]for broadcasting video streams, and according to the success of edge and fog computing [15], video streaming *assisted by* the Peer-to-Peer (P2P) technology has also attracted considerable attention as a method to realize a scalable, stable video streaming over the Internet [21], [26], [29], [32], [33].

This paper considers P2P-assisted video streaming with *non-uniform quality levels*, designated by the spatial resolution and the frame rate. For example, the resolution of high definition (HD) video is $1280 \times 720$ and that of 8K video is $7680 \times 4320$, whereas concerned with the frame rate, HD supports 30 fps and 8K supports up to 120 fps [23]. The difference of quality levels is generally reflected to the *bitrate* of the resulting video streams, while it is highly dependent on the underlying codec.

In this paper, we assume that each video stream has its own quality level encoded with a constant bitrate[*****], and consider the problem of delivering those video streams to their subscribers with as small server cost as possible while keeping the performance of video streaming in terms of the throughput and the latency. If a simple tree-structured overlay is used for delivering each video stream, which will be referred to as a *naive scheme* hereafter, subscribers of high quality video stream are easily overloaded, and it limits the number of participants which allows the delivery with a designated latency and a server cost [1], [12], [13]. In general, to attain a low latency in P2P video streaming, we should bound the number of intermediate peers encountered during the delivery of video streams, but it is difficult for high quality video streams since it occupies a large portion of the upload capacity of the participant which severely limits the number of peers which could directly receive a stream from intermediate peers (e.g., each peer can forward a received video stream to at most one peer if the bitrate of the stream exceeds a half of the upload capacity of the peer).

We will overcome this problem by dividing a given video stream into several sub-streams called stripes and by delivering those stripes to the subscribers through different tree-structured overlays. The division of a video stream into stripes is conducted in such a way that the bitrate of each stripe is as even as possible, indicating that a high quality video stream is delivered through many contributing peers. Such a stripe-based approach could average the load of the participants, and could effectively resolve the overloading of overlays for high quality video streams. More specifically, we could reduce the bitrate of a stream by dividing it into several stripes. The reduction of the bitrate could significantly reduce the server cost since the number of subscribers covered by a delivery tree with a fixed height increases *exponentially* while it *linearly* increases the overhead as the number of stripes increases. The performance of the proposed scheme is evaluated numerically. The result of evaluations indicates that the proposed method reduces the server

Manuscript received March 28, 2019.
Manuscript revised May 31, 2019.
Manuscript publicized August 8, 2019.

[†]The author is with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 739–8527 Japan.

*This work was partially supported by KAKENHI, the Grant-in-Aid for Scientific Research (B),F Grant Number 16H02807.

a) E-mail: fujita@se.hiroshima-u.ac.jp

DOI: 10.1587/transinf.2019EDP7088

**https://www.akamai.com/

***https://azure.microsoft.com/en-us/services/cdn/

****https://www.verizondigitalmedia.com/

*****Although practical codec such as MPEG-4 and H.264 encodes video with a variable bitrate, to clarify the exposition, we will assume that each stripe is encoded with a constant bitrate.

cost to only 3% of the naive scheme if the number of delivery hops is bounded by four, the upload capacity of each peer is fixed to eight (i.e., if each peer can simultaneously forward eight stripes to other peers), and the video stream is divided into four stripes.

The remainder of this paper is organized as follows. Section 2 overviews related work. Section 3 gives preliminaries. Section 4 describes the proposed method. Section 5 summarizes the result of evaluations. Finally, Sect. 6 concludes the paper with future work.

## 2. Related Work

Video streaming assisted by the P2P technology has been widely used in recent years [28], [30], [31]. For example, the demonstration experiments conducted by NHK science & technology research laboratories during London Olympics in 2012 indicate that a mesh-based P2P realizes the delivery of live contents to more than 1600 subscribers in 1.5 Mbps in a stable manner [22]. P2P video streaming systems can be classified into several types by the way of delivering video contents to the subscribers, including tree-type such as SCRIBE [6] and Bayeux [34], mesh-type such as Bullet [17], PRIME [20], and CoolStreaming/DONet [29] and a hybrid of mesh and tree such as mTreebone [25].

The idea of using *multiple trees* for video streaming was firstly adopted in SplitStream [5]. More concretely, SplitStream divides a given video stream into $b$ stripes so that the $j^{th}$ stripe, for $1 \le j \le b$, consists of the $(bi + j)$th chunks in the given stream for $i \ge 0$ [1]. Then it delivers those stripes through different spanning trees to have disjoint sets of internal nodes. In other words, in SplitStream, each peer joins at most one spanning tree as an internal node and joins all of the other spanning trees as a leaf node. Such a construction of the set of spanning trees enables peers to contribute their upload capacity with low cost, and balances the load of all peers participating in the streaming system [2], [9].

Theoretical aspects concerned with multiple-tree-based P2P video streaming have also been studied in recent years. Liu [18] considered the problem of minimizing the broadcast time of each chunk contained in a given stream under the constraint such that each peer can upload at most one chunk at a time[†], and proposed an algorithm which broadcasts every chunk to $n$ subscribers in $\lceil \log_2 n \rceil$ steps. In this algorithm, any two consecutive chunks are delivered through different binomial trees since in order to enable the delivery of chunks in $\lceil \log_2 n \rceil$ steps, every peer receiving a chunk must continuously upload the chunk to different receivers in the succeeding steps until the chunk is received by all subscribers (to complete the broadcast of a chunk to $n$ subscribers in $\lceil \log_2 n \rceil$ steps, the number of subscribers receiving the chunk must *double* in each step). Liu's result was extended to the cases in which each peer has a constant number of neighbors in the overlay [10] and the upload capacity of peers is not uniform [11], respectively.

In addition to the above results, the upper bound on the network capacity of multiple-tree-based P2P was discussed in different contexts. For example, [19] discussed the network capacity of peer-assisted live streaming, and [16] considered the problem of constructing multiple trees which maximize the network capacity by considering the topology of the underlying physical network.

[†]This assumption is slightly relaxed by Bianchi *et al.* [3] so that each peer can upload at most $k$ chunks at a time.

## 3. Preliminaries

### 3.1 Model of P2P System

Let us consider a P2P system consisting of a media server and a set of subscribers $\mathcal{P}$. Each video stream published by the media server has a fixed *quality level* drawn from $\{1, 2, \ldots, k\}$, where $k$ means the highest quality level. Let $\mathcal{S} \stackrel{\text{def}}{=} \{\sigma(u) : u \in \mathcal{P}\}$ be the set of video streams subscribed by the peers in $\mathcal{P}$, where $\sigma(u)$ denotes the video stream subscribed by peer $u$.

In the proposed method, we assume that a video stream of quality level $i$ is divided into $i$ **stripes** by the media server beforehand, and those stripes are delivered to the corresponding subscribers through different delivery trees[††]. More concretely, the media server pushes each stripe to several subscribers each of which serves as the root of a delivery tree corresponding to the stripe, and each stripe received by the root is disseminated to the remaining nodes in the delivery tree through tree edges (note that each stripe can have several roots). In other words, peer $u$ can receive a stripe generated from stream $\sigma(u) \in \mathcal{S}$ by connecting to a peer in a delivery tree corresponding to the stripe as a receiver, and can restore $\sigma(u)$ from received stripes, where we assume that the overhead for the restoration is negligible. To simplify the exposition, in the following, we assume that every stripe has the same bitrate and each peer has an ability of simultaneously forwarding *any* $b$ stripes to other peers, where $b$ is called the **upload capacity** of a peer. In addition, we will bound the height of any delivery tree by $h$ under the constraint on the upload capacity, to guarantee that every stripe is received by any subscriber with a designated latency (i.e., a deepest leaf of the delivery tree receives the stripe $h$ hops away from the media server, and a peer connecting to the deepest leaf receives the stripe within $h + 1$ hops away from the media server).

### 3.2 Basic Observations

For each $s \in \mathcal{S}$, let $q(s)$ denote the quality level of stream $s$,

[††]Such a division into stripes can be realized by using MDC (Multiple Description Coding) [4], [24], for example. MDC was originally proposed to improve the resiliency of video streaming, in such a way that each stripe can be decoded independently and the quality of restored video stream increases as the number of decoded stripes increases. MDC-based P2P video streaming was proposed in [7].

and $n(s)$ denote the number of subscribers of $s$. Let $X$ be the upload capacity of the media server consumed for delivering $\mathcal{S}$ to their subscribers, which equals to the total number of roots for $\mathcal{S}$. The following inequality holds regardless of the intended latency (i.e., delivery hops) of the video streaming:

$$b \sum_{s \in \mathcal{S}} n(s) + X \geq \sum_{s \in \mathcal{S}} q(s) \times n(s), \tag{1}$$

where the left hand side is the amount of upload capacities and the right hand side is the amount of required downloads. In client/server systems, $X = \sum_{s \in \mathcal{S}} q(s) \times n(s)$, since each subscriber directly receives $q(s)$ stripes from the media server. If $n(s^*) = |\mathcal{P}|$ and $X = q(s^*)$ for some $s^* \in \mathcal{S}$, namely, if all peers subscribes to a single stream $s^*$ and every stripe generated from $s^*$ is pushed to exactly one peer by the media server, then Eq. (1) is restated as

$$b \geq q(s^*)\left(1 - \frac{q(s^*)}{n(s^*)}\right),$$

which intuitively implies that each peer must have an upload capacity of amount $q(s^*)$, which could slightly reduce if $n(s^*)(= |\mathcal{P}|) < \sqrt{q(s^*)}$.

In the above model of P2P video streaming, the delivery of a stripe is realized with the assistance of several peers. A peer $u$ which assists the delivery of stream $s^*$ is called a **helper** if $\sigma(u) \neq s^*$. It is known that even if $b = k$, the existence of helper is mandatory to enable the delivery of $k$ stripes to $\sum_{i=0}^{h} b^i - 2$ subscribers as long as $X = b$ [14].

## 4. Proposed Method

Given a set of subscribers of a video stream, we can easily construct a set of spanning trees which delivers all stripes generated from the stream to the subscribers (in fact, SplitStream uses such a set of spanning trees). However, it is not trivial *how to maintain such trees against dynamic change of the set of subscribers, while keeping the delivery of stripes to the existing peers with a designated delivery hops (i.e., latency)*. The main contribution of this paper is to propose a concrete procedure for such maintenance operations.

### 4.1 Core Tree

The proposed scheme is designed with the notion of core trees. **Core tree** is a subtree of a complete $b$-ary tree of height $h$ consisting of $1 + b + b^2 + \cdots + b^{h-1}$ peers (recall that in the graph theory, the height of a trivial tree consisting of a single peer is defined to be one). Figure 1 illustrates a core tree for $b = 3$ and $h = 3$. Since a complete core tree has $b^{h-1}$ leaves with upload capacity $b$ each, it has an upload capacity of amount $b^h$. In other words, if a stripe is pushed to the root of a complete core tree, it can be delivered to $b^h$ peers through the tree in exactly $h + 1$ hops away from the media server excluding peers contained in the core tree.

To deliver a video stream divided into several stripes, we should prepare at least one core tree for each stripe, and the number of core trees should increase as the number of subscribers increases since a single core tree can contain at most $\frac{b^{h+1}-1}{b-1}$ subscribers.

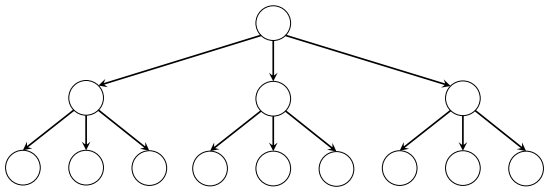


**Fig. 1** Core tree for $b = 3$ and $h = 3$.

### 4.2 Growth of Core Trees

Next, we give a procedure to *grow* core trees while keeping the delivery of stripes to the subscribers within $h + 1$ hops. A core tree is said to be **active** (or incomplete) if it contains less than $\sum_{i=0}^{h-1} b^i$ peers. Consider the delivery of a video stream of quality level $b$ to the growing set of subscribers (the case of other quality levels will be discussed later). Let $R = \{r_1.r_2, \ldots, r_b\}$ be the set of $b$ stripes generated from the video stream.

In the proposed procedure, the set of core trees is maintained in such a way that it contains exactly one active core tree for each stripe to satisfy an invariant such that *each active tree has the residual capacity of amount b*. The procedure starts with $b$ trees, each of which consists of a single helper peer and is dedicated to the delivery of a stripe in $R$. Such an initial configuration certainly satisfies the above invariant, since each helper is not required to forward a received stripe to other helpers; namely each tree has a residual capacity of amount $b$.

Those $b$ active trees grow by repeating *rounds* in which all of $b$ trees increase their size by exactly one. Assume that during a round, $b$ peers $u_1, u_2, \ldots$, and $u_b$ join the set of subscribers in this order. Let $T_i$ denote the active tree for stripe $r_i$. For each $1 \leq i \leq b$, $u_i$ becomes a member of tree $T_i$. More concretely,

1. Let $v$ be a peer in $T_i$ at the smallest depth (i.e., closest to the root) among peers with less than $b$ children (i.e., with a positive residual capacity). Peer $u_i$ becomes a child of $v$ in $T_i$, and if $v$ is forwarding stripe $r_i$ to a peer $w$ belonging to other tree, then $u_i$ substitutes for the role of forwarding $r_i$ to $w$ so that $v$ can have up to $b$ children in $T_i$.
2. $u_i$ establishes a *tentative connection* to a peer $w'$ in $T_j$ (with a residual capacity) for each $i + 1 \leq j \leq b$, to receive stripe $r_j$ from $w'$. Note that such $w'$ always exists since we are assuming that each active tree has a residual capacity of amount $b$ at the beginning of a round, and such $w'$ can be quickly identified by keeping the indices to (at most $b$) such peers on the media server.
3. $u_i$ establishes a connection to $u_j$ for each $1 \leq j \leq i - 1$, to receive stripe $r_j$ from $u_j$, and to substitute for the role

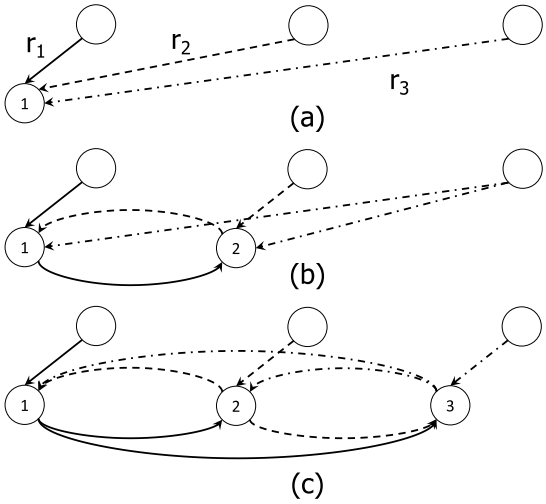



**Fig. 2** Joining of the first three peers to the initial core trees. Stripe $r_1$ is delivered to peers 2 and 3 through peer 1; Although stripe $r_2$, indicated by dashed lines, is delivered to peer 1 from the root in (a), it is substituted by peer 2 in (b) and (c).

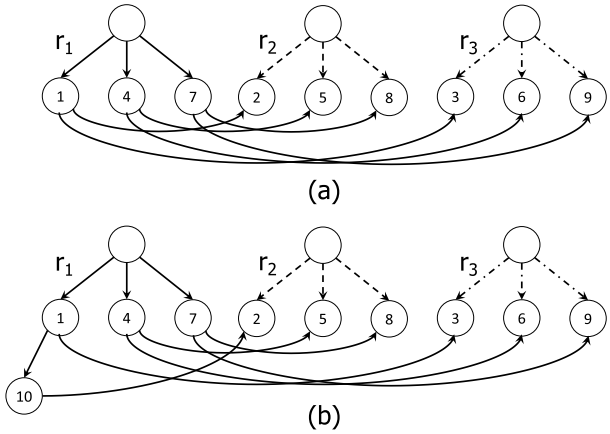



**Fig. 3** Joining of the tenth peer to the initial core trees. Figure (a) shows the configuration after the joining of the ninth peer, where the second hop of stripes $r_2$ and $r_3$ is not displayed to clarify the exposition. In Fig. (b), the tenth peer 10 joins $T_1$ as a child of peer 1, and then substitutes for the forwarding of $r_1$ to peer 2 (when peer 1 accepts the second child, it will substitute for the forwarding of $r_1$ to peer 3 so that peer 1 can have the third child).

of forwarding stripe $r_i$ to $u_j$.

Figure 2 illustrates the first round of the procedure for $b = 3$, where peers 1, 2 and 3 join the set of subscribers in this order. At any point in time, each peer receives three stripes $r_1$, $r_2$, and $r_3$ indicated by solid, dashed, and dot-dashed lines, respectively. Figure 3 illustrates the first step of the fourth round. After becoming a child of peer 1, peer 10 substitutes for the role of forwarding stripe $r_1$ to peer 2 (to clarify the exposition, this figure omits the second hop of the delivery of stripes $r_2$ and $r_3$).

The reader should observe that if each active tree has a residual capacity of amount $b$ at the beginning of a round, it still has the same capacity after the round, since although the capacity of the tree of amount $b$ is consumed by $b$ new peers, a newly added peer increases the capacity of the tree by $b$. In addition, since each peer in a core tree can have $b$ children, after the $(\sum_{i=0}^{h} b^i - 1)$st round, we have $b$ complete core trees of capacity $b$ each. Thus we can add $b$ more peers, say $u'_1, u'_2, \ldots, u'_b$, to the resulting (complete) core trees so that each peer $u'_i$ receives $b$ stripes in exactly $h + 1$ hops away from the media server through those core trees. Those $b$ peers can become roots of new active trees by requesting the media server to directly push $r_i$ to $u'_i$ (in contrast to the initial core trees, the roots of next trees need not be helper since they receive $b - 1$ stripes from former active trees).

The overhead required for adding a new peer to the set of active trees is evaluated as follows. The new peer contacts $b$ peers to receive stripes through those peers, and contacts $b$ peers to forward a received stripe to those peers, while the "substitution" of the role of forwarding needs an additional contact. Peers to be connected can be quickly identified by keeping the indices to $b$ lastly joined peers on the media server for each active tree (i.e., the memory size is $O(b)$ for each stripe), although it incurs a bottleneck at the server when many peers arrive at the system in a short time period. Finally, an additional contact to the media server should occur if the joined peer becomes the root of a new active tree. In summary, the overhead for adding a new peer to the set of $b$ active trees is $O(b)$.

### 4.3 Leave from Core Trees

This subsection describes a procedure to *remove* a designated peer from the set of subscribers. Recall that core trees grow by repeating round. Suppose that in the current round, peers $v_1, v_2, \ldots$, and $v_{b'}$ join the set of subscribers in this order, where $1 \le b' \le b$. For each subscriber $v$ contained in a core tree, let $Q_{in}(v)$ denote the set of peers forwarding a stripe to $v$ (which contains the parent of $v$ in a core tree unless it is the root), and $Q_{out}(v)$ be the set of peers receiving a stripe from $v$.

Assume that a leaving peer $u$ is a member of a core tree concerned with stripe $r_i$. If $u = v_{b'}$, then the removal of $u$ proceeds as follows (see Fig. 4 for illustration).

1. It requests $|Q_{out}(u)|(\le b)$ peers in $T_i$ with a residual capacity to forward $r_i$ to a peer in $Q_{out}(u)$.
2. It requests all peers in $Q_{in}(u)$ to stop the forwarding of a stripe to $u$.
3. Then $v_{b'}$ leaves.

On the other hand, if $u \ne v_{b'}$, $v_{b'}$ is removed from $T_{b'}$ and then substitutes for the leaving peer $u$ (see Fig. 5 for illustration). More concretely,

1. It requests all peers in $Q_{in}(u)$ to change the receiver of a stripe to $v_{b'}$ and requests all peers in $Q_{in}(v_{b'})$ to stop the forwarding of a stripe to $v_{b'}$.
2. It requests $v_{b'}$ to forward stripe $r_i$ to all peers in $Q_{out}(u)$ and requests $|Q_{out}(v_{b'})|(\le b)$ peers in $T_{b'}$ with a residual capacity to forward stripe $r_{b'}$ to a peer in $Q_{out}(v_{b'})$.
3. Then $u$ leaves.

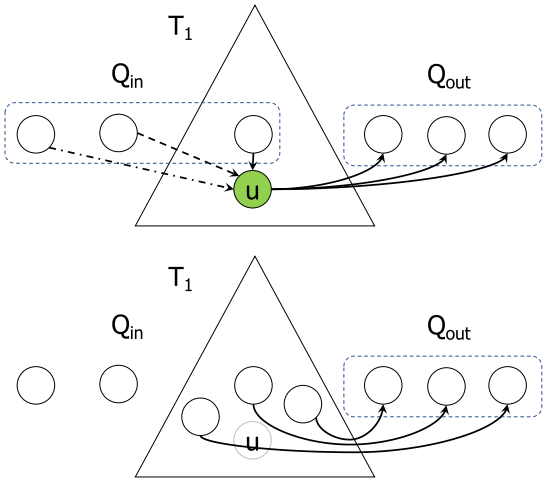



**Fig. 4** When leaving peer $u$ is the lastly joined peer. The forwarding of a stripe to $u$ is canceled, and the forwarding of stripe $r_1$ by $u$ is substituted by peers in $T_1$ with residual capacity.

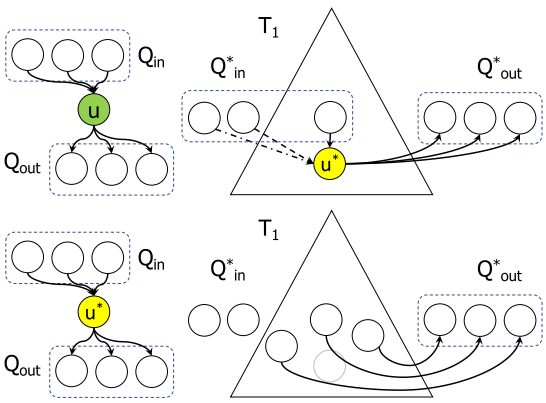



**Fig. 5** When leaving peer $u$ joined $T_i$ in a former round.

If several peers leave the system, the removal of the second peer can start after completing the removal of the first peer, and such a conflict can be resolved at the media server (in other words, we are assuming that the media server plays the role of tracker).

Similar to the addition of a peer, the overhead required for removing a peer from a core tree can also be bounded by $O(b)$ by keeping the indices to $b$ lastly joined peers on the media server for each active tree.

### 4.4 Video Streaming of Lower Quality Level

The proposed scheme uses $b^*$ core trees for delivering a video stream of quality level $b^*$. Thus if $b^* < b$, we can allow several subscribers *not to join any core tree*. More concretely, among $b$ peers newly joined the set of subscribers during a round, merely $b^*$ peers should become a member of a core tree and the remaining $b - b^* \geq 1$ peers are not required to forward any stripe to other peers. See Fig. 6 for illustration.

Such a modification does not violate the invariant such that the residual capacity of each active core tree is exactly

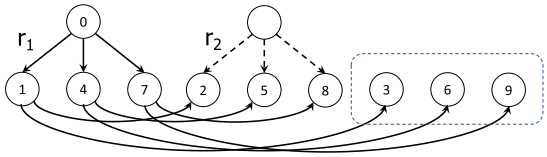



**Fig. 6** When $b = 3$ and $i = 2$, peers 3, 6 and 9 are not required to forward a stripe to other peers.

$b$, and does not reduce the number of subscribers covered by $b^*$ core trees from $\sum_{j=0}^{h} b^j$, while it allows $\sum_{j=0}^{h-1} b^j(b - b^*)$ peers not to join any core tree, which could be used as a helper for other video streams; i.e., the proposed stripe-based scheme could effectively realize a load balancing among video streams of different quality levels.

### 4.5 Video Streaming of Higher Quality Level

With the notion of helpers, we could realize the delivery of $b^* > b$ stripes to the subscribers; namely we could realize the delivery of a video stream with a *super high quality* exceeding the upload capacity $b$. The idea is to use $b^* - b + 1$ **helper trees** consisting of $\sum_{j=0}^{h-1} b^j$ helpers in addition to $b-1$ ordinary core trees. The role of ordinary core trees is to mutually deliver $b - 1$ stripes to

$$(b-1)\sum_{j=0}^{h-1} b^j = b^h - 1$$

peers in the ordinary core trees, and the role of helper trees is to forward the remaining $b^* - b + 1$ stripes to $b^h$ peers contained in ordinary core trees.

## 5. Evaluation

This section numerically evaluates the performance of the proposed method. At first, we analyze the server cost required for guaranteeing a given latency (i.e., delivery hops). We then evaluate the residual capacity of the overall network to certify that the reduction of the server cost does not rely on the overloading of the subscribers. Since there is no previous scheme for maintaining delivery trees while keeping the delivery of stripes with a designated latency, we merely compare it with a naive tree-based scheme which does not divide a given video stream into stripes.

In the following evaluation, the upload capacity $b$ is fixed to either 8 or 12, since those value have many divisors. Quality level $i$ of video stream is varied from 2 to $b/2$ so that each peer can have at least two children in the naive scheme. Finally, we fix $h$ to three, which is the maximum number of hops away from the root peer, since the effect of peer-assistance is too restrictive for small $h$'s, and a too long delivery path could cause failures and performance degradation.

### 5.1 Server Cost

By assumption, the upload of a video stream of quality level $i$ consumes the upload capacity of amount $i$. Thus in the naive scheme, each peer can have at most $d = \lfloor b/i \rfloor$ children in the delivery tree, which implies that if $d \geq 2$, the server cost of amount $i$ can cover at most

$$\sum_{i=0}^{h+1} d^i = \frac{d^{h+2} - 1}{d - 1}$$

peers provided that the latency from the media server is bounded by $h + 1$. Hence the server cost which is necessary to cover $N$ subscribers within $h + 1$ hops is at least

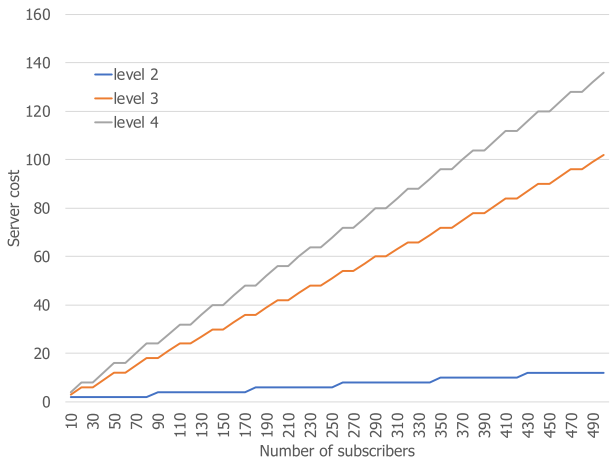
$$\left\lceil \frac{N(d-1)}{d^{h+2} - 1} \right\rceil \times i.$$

Figure 7 shows how this value increases as the number of subscribers $N$ increases for each $i$. Although it could be bounded by a small value for small $i$'s, it rapidly increases as the quality level $i$ increases, e.g., when $b = 12$, $h = 3$, and $N = 500$, the server cost is only four for $i = 2$ (i.e., 500 peers are covered by two delivery trees), but increases to 18 for $i = 3$, and to 52 for $i = 4$.

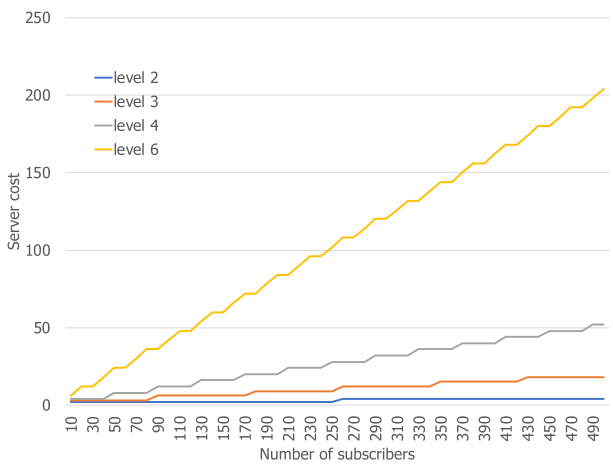
On the other hand, in the proposed scheme, since a video stream of quality level $i$ uses $i$ core trees consisting of $\sum_{j=0}^{h} b^j$ peers for the delivery of $i$ stripes, the server cost is bounded by $i$ as long as $N \leq ib \sum_{j=0}^{h} b^j$. Hence the server cost to cover the delivery of $i$ stripes to $N$ subscribers within $h + 1$ hops is

$$\left\lceil \frac{N(b-1)}{b(b^{h+1} - 1)} \right\rceil \times i$$

(note that when $i = 1$, the proposed scheme covers fewer subscribers than the naive scheme). The server cost significantly reduces by the proposed scheme. In particular, for any $i \leq b$, it coincides with the number of stripes $i$ (i.e., it takes the minimum value) as long as $N \leq b \sum_{j=0}^{h} b^j$; e.g., if $N \leq 4680$ for $b = 8$ and $h = 3$, and if $N \leq 22620$ for $b = 12$ and $h = 3$.




(a) When $b = 8$ and $h = 3$.




(b) When $b = 12$ and $h = 3$.

**Fig. 7** Server cost in the naive scheme.

### 5.2 Residual Capacity

Finally, we evaluate the residual capacity of peers in the proposed scheme, and compare it with the naive scheme to certify that the reduction of the server cost does not rely on the overloading of the subscribers. In the proposed scheme, a video stream of quality level $i < b$ yields peers with a residual capacity. More precisely, for sufficiently large $N$, the ratio of peers to have residual capacity of amount $b$ is given as $\frac{b-i}{b}$ for each level $1 \leq i \leq b$. On the other hand, the residual capacity in the naive scheme is evaluated as follows. Assume $i \leq b/2$, without loss of generality, since otherwise, given video stream should be delivered through a path consisting of at most $h+1$ peers. The ratio of leaf peers in $d$-ary trees of height $h+1$ is maximized when all leaves are of depth $h+1$; namely when it is a complete $d$-ary tree. Since a complete $d$-ary tree of height $h+1$ has $d^h$ leaves and $\sum_{i=0}^{h-1} d^i$ internal peers, for sufficiently large $N$, the ratio

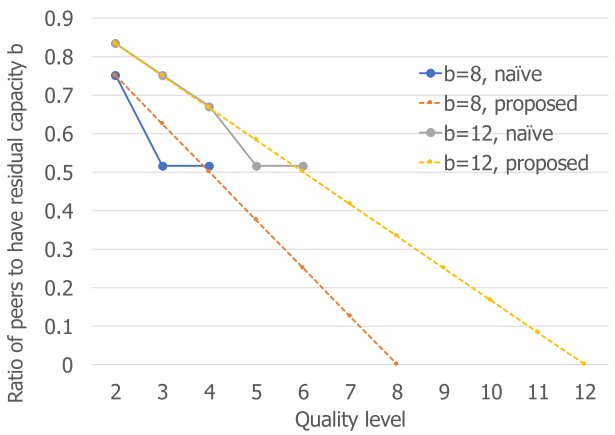



**Fig. 8** The ratio of peers to have residual capacity of amount $b$ in each scheme for $h = 3$. Dashed lines represent the proposed scheme and solid lines with marks indicate the naive scheme.

of leaf peers to have residual capacity of amount $b$ is given as $\frac{d^h}{\sum_{i=0}^{h} b^i}$. In addition, given $N \leq \sum_{i=0}^{h} b^i$ subscribers, we can construct a $d$-ary tree of height at most $h+1$ so that there is at most one peer to have a residual capacity of amount $b'$ for some $1 \leq b' \leq b-1$ (i.e., so that the residual capacity of the other peers is either 0 or $b$).

Figure 8 shows how those values decrease as the quality level $i$ increases. We could observe from the figure that if $i \leq b/2$, the proposed scheme allow more peers to have residual capacity of amount $b$ than the naive scheme.

## 6. Concluding Remarks

This paper proposes a scheme to deliver video streams to the subscribers with a designated latency in different quality levels. The proposed scheme is designed with the notion of core trees, and could significantly reduce the server cost while guaranteeing a designated latency. A future work is to evaluate the dynamic behavior of the proposed scheme by conducting event-driven simulations.

### References

[1] H. Ando and S. Fujita, "Tight Bounds for Two-Hop Delivery in Homogeneous P2P Video Streaming Systems," Proc. 4th International Symposium on Computing and Networking (CANDAR), pp.1–8, 2016.
[2] B.A. Alghazawy and S. Fujita, "A Scheme for Maximal Resource Utilization in Peer-To-Peer Live Streaming," Int. J. Comput. Netw. & Communications (IJCNC), vol.7, no.5, pp.13–28, Sept. 2015.
[3] G. Bianchi, N.B. Melazzi, L. Bracciale, F.L. Piccolo, and S. Salsano, "Streamline: An Optimal Distribution Algorithm for Peer-to-Peer Real-Time Streaming," IEEE Trans. Parallel Distrib. Syst., vol.21, no.6, pp.857–871, June 2010.
[4] M. Biswas, M.R. Frater, and J.F. Arnold, "Multiple description wavelet video coding employing a new tree structure," IEEE Trans. Circuits Syst. Video Technol. vol.18, no.10, pp.1361–1368,, 2008.
[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth Multicast in Cooperative Environments," Proc. the 19th ACM Symp. Operating Systems Principles (SOSP), pp.298–313, 2003.
[6] M. Castro, P. Druschel, A.-M Kermarrec, and A. Rowstron, "SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," IEEE J. Sel. A. Commun., vol.20, no.8, pp.1489–1499, Oct. 2002.
[7] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptions for video streaming over multiple paths," Multimedia Systems, vol.10, no.4, pp.275–285, 2005.
[8] Cisco, "Visual Networking Index-Forecast and Methodology 2016-2021," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf,
[9] G. Dan, V. Fodor, and I. Chatzidrossos, "On the Performance of Multiple-Tree-Based Peer-to-Peer Live Streaming," Proc. the 26th IEEE Int'l Conf. Computer Communications (INFOCOM), pp.2556–2560, 2007.
[10] S. Fujita, "Optimal Serial Broadcast of Successive Chunks," Theoretical Computer Science, vol.575, pp.3–9, April 2015.
[11] S. Fujita, "Broadcasting a Stream of Chunks in Heterogeneous Networks with a Short Maximum Broadcast Time," J. Interconnection Networks (JOIN), vol.18, no.2n03, 2018.
[12] S. Fujita, "Cloud-Assisted Peer-to-Peer Video Streaming with Minimum Latency," IEICE Trans. Inf. & Syst., vol.E102-D, no.2, pp.239–246, Feb. 2019.
[13] S. Fujita, "Flash Crowd Absorber for P2P Video Streaming," IEICE Trans. Inf. & Syst., vol.E102-D, no.2, pp.261–268, Feb. 2019.
[14] S. Fujita, "Multi-Tree-Based Peer-to-Peer Video Streaming with a Guaranteed Latency," vol.E102-D, no.9, pp.1707–1714, Sept. 2019.
[15] H.-J. Hong, "From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices," Proc. IEEE Int. Conf. Cloud Computing Technology and Science (CloudCom), pp.331–334, 2017.
[16] X. Jin, W.-P.K. Yiu, S.-H.G. Chan, and Y. Wang, "On Maximizing Tree capacity for Topology-Aware Peer-to-Peer Streaming," IEEE Trans. Multimedia, vol.9, no.8, pp.1580–1592, Dec. 2007.
[17] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth Data Dissemination using an Overlay Mesh," Proc. the 19th ACM Symp. Operating Systems Principles (SOSP), pp.282–297, 2003.
[18] Y. Liu, "On the Minimum Delay Peer-to-Peer Video Streaming: How Realtime Can It Be?" Proc. 15th ACM Int'l Conf. on Multimedia., pp.127–136, 2007.
[19] S. Liu, Z.-S. Rui, W. Jiang, J. Rexford, and M. Chiang, "Performance Bounds for Peer-Assisted Live Streaming," Proc. Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS), pp.313–324, 2008.
[20] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-Driven Mesh-Based Streaming," IEEE/ACM Trans. Networking, vol.17, no.4, pp.1052–1065, Aug. 2009.
[21] L. Natali and M.L. Merani, "Successfully Mapping DASH Over a P2P Live Streaming Architecture," IEEE Trans. Circuits Syst. Video Technol., vol.27, no.6, pp.1326–1339, June 2017.
[22] https://www.nhk.or.jp/strl/publica/rd/rd137/PDF/P57-62.pdf (in Japanese)
[23] S. Saito, T. Shitomi, S. Asakura, A. Satou, M. Okano, K. Murayama, and K. Tsuchida, "8K Terrestrial Transmission Field Tests Using Dual-Polarized MIMO and Higher-Order Modulation OFDM," IEEE Trans. Broadcasting, vol.62, no.1, pp.306–311, March 2016.
[24] V. Stankovic, R. Hamzaoui, and Z. Xiong, "Robust layered multiple description coding of scalable media data for multicast," IEEE Signal Process. Lett. vol.12, no.2, pp.154–157, Feb. 2005.
[25] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming," IEEE Trans. Parallel Distrib. Syst., vol.21, no.3, pp.379–392, March 2010.
[26] W. Wu, H. Mao, Y. Wang, J. Wang, W. Wang, and C. Tian, "CoolConferencing: Enabling Robust Peer-to-Peer Multi-Party Video Conferencing," IEEE Access, vol.5, pp.25474–25486, 2017.
[27] X. Xiao, Q. Zhang, Y. Shi, and Y. Gao, "How Much to Share: A Repeated Game Model for Peer-to-Peer Streaming under Service Differentiation Incentives," IEEE Trans. Parallel Distrib. Syst., vol.23, no.2, pp.288–295, Feb. 2012.
[28] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Livesky: Enhancing CDN with P2P," ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) vol.6, no.3, Aug. 2010.
[29] X. Zhang, J. Liu, B. Li, and Y. -S.P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," Proc. the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies, vol.3, pp.2102–2111, 2005.
[30] G. Zhang, W. Liu, X. Hei, and W. Cheng, "Unreeling Xunlei kankan: understanding hybrid CDN-P2P video-on-demand streaming," IEEE Trans. Multimedia, vol.17, no.2, pp.229–242 Feb. 2015.
[31] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec, "Peer-assisted content distribution in Akamai NetSession," Proc. 2013 conference on Internet measurement conference, pp.31–42, ACM, 2013.
[32] C. Zhao, J. Zhao, X. Lin, and C. Wu, "Capacity of P2P On-Demand Streaming With Simple, Robust, and Decentralized Control," IEEE/ACM Trans. Networking, vol.24, no.5, pp.2607–2620, Oct. 2016.
[33] Y. Zhou, T.Z.J. Fu, and D.M. Chiu, "A Unifying Model and Analysis of P2P VoD Replication and Scheduling," IEEE/ACM Trans.

Networking, vol.23, no.4, pp.1163–1175, Aug. 2015.
[34] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," Proc. NOSSDAV'1, pp.11–20, 2001.

**Satoshi Fujita** received the B.E. degree in electrical engineering, M.E. degree in systems engineering, and Dr.E. degree in information engineering from Hiroshima University in 1985, 1987, and 1990, respectively. He is a Professor at Graduate School of Engineering, Hiroshima University. His research interests include communication algorithms, parallel algorithms, graph algorithms, and parallel computer systems. He is a member of the Information Processing Society of Japan, SIAM Japan, and IEEE Computer Society.