Low Cost Cloud-Assisted Peer to Peer Live Streaming

Bahaa Aldeen Alghazawy, and Satoshi Fujita
Graduate School of Engineering, Hiroshima University, Hiroshima, Japan.
[e-mail: bahaa@se.hiroshima-u.ac.jp; fujita@se.hiroshima-u.ac.jp]
*Corresponding author: Bahaa Aldeen Alghazawy

Received October 15, 2015; revised December 28, 2015; revised January 26, 2016; accepted February 14, 2016; published April 30, 2016

Abstract

Recently, Peer-to-Peer (P2P) live streaming assisted by the cloud computing has attracted considerable attention to improve the reliability of the P2P such as the resilience to peer churn and the shortage of upload capacity. The cost of cloud-assistance is comprised of the number of requests issued to the cloud and the amount of data fetched from the cloud. In this paper, we propose three techniques to reduce the cost of such a cloud-assistance. More concretely, in the proposed method, 1) each peer which lost its parent in the overlay can find a new parent by referring to the information registered in the cloud, 2) several peers which proactively fetch chunks from the cloud are dynamically invested, and 3) the number of requests issued to the cloud is reduced by allowing peers to fetch a collection of chunks using a single request. The performance of the proposed method is evaluated by simulation. The simulation results indicate that it reduces the cost of conventional scheme by 46% while guaranteeing the quality of live streaming service.

Keywords: Live Streaming, Peer-to-Peer, Cloud-Assistance, and Quality of Service.

http://dx.doi.org/10.3837/tiis.2016.04.015 **ISSN : 1976-7277**

1. Introduction

In recent years, several commercial streaming services such as PPLive and UUSee have adopted Peer-to-Peer (P2P) technology to deliver live contents to thousands of users. Although it could realize a scalable video streaming with low cost compared with classical client/server systems [1-3], P2P live streaming is not reliable enough as it relies on the voluntary contribution of the participant peers. Therefore, it is hard to guarantee the quality of streaming service such as playback continuity and short playback delay [4]. To overcome such issues, *hybrid of P2P with other infrastructures* such as the cloud computing and content delivery networks has attracted considerable attentions in recent years [5-8].

In this paper, we focus on the assistance of P2P live streaming by the cloud with a storage service and a cloud content delivery network (CCDN)[1]. The assistance of P2P live streaming by the cloud is generally realized by storing latest chunks in the live stream to the storage service in the cloud. In addition, each peer is allowed to fetch missing chunks from the storage service through edge locations of the CCDN. These edge locations fetch chunks from the storage service upon receiving requests from peers. Another way for the assistance of P2P by the cloud is to rent computing instances and to use them as *virtual peers* to disseminate live stream to the existing peers [6-7]. However, the rent of computing instances is less flexible than the rent of upload bandwidth (e.g., the minimum rent period of computing instances is generally one hour). Thus to refine the cost of cloud assistance while maintaining the high performance, we take the former approach to propose a *cost effective* cloud-assisted P2P live streaming system.

The cost of cloud assistance is comprised of the amount of data fetched from the CCDN and the number of requests handled by the cloud. The latter cost is further divided into requests handled by the storage service and requests handled by the CCDN. We propose three techniques to reduce the cost of such a cloud assistance and evaluate them through extensive simulations. As the baseline model of our study, we use a multiple-tree overlay adopted in SplitStream [9]. In this model, a given live stream is equally divided into sub-streams by the sequence number of chunks contained in the stream. Each sub-stream is delivered to all peers (subscribers) through a different delivery tree. We assume that the sets of internal nodes of the trees must be mutually disjoint as in SplitStream. In such a multi-tree-structured P2P network, the leave of an internal node suspends the forwarding of sub-stream to the descendants. That significantly increases the amount of data fetched from the CCDN. The first technique reduces such a cost by exploiting the cloud storage service to explicitly register “orphaned” peers which lost their parent in the delivery trees. Match-making between orphaned peers and internal peers with enough capacity can be done by referring to the registered information. That significantly reduces the time before an orphaned peer becomes a child of a new parent in the delivery tree. However, it incurs additional cost due to requests handled by the storage service.

The second and third techniques are based on the first technique. The key idea of the second technique is to proactively fetch chunks by several internal nodes selected from each tree. The selected node (peer) plays the role of a root for the corresponding tree, which reduces the height of the delivery tree and the load of other internal nodes. The number of peers to be selected is controlled by referring to the number of orphaned peers registered to the storage

[1] Example of CCDN includes CloudFront: http://aws.amazon.com/cloudfront

service. That is by the reason if the number of selected peers is too large, the amount of proactively fetched chunks becomes excessive and if it is too small, the cost due to orphaned peers increases. The third technique reduces the number of requests handled by the CCDN in the second technique by allowing peers to request a collection of chunks instead of individual chunks. More concretely, when proactively fetching chunks from the CCDN as a selected peer for tree t, it acquires consecutive chunks associated with the whole stream (instead of a particular sub-stream) in the form of frames of chunks, and serves as a root for tree t.

The performance of the proposed method is evaluated by simulation. We implement our simulator on top of an event-driven P2P simulator [10]. The simulation results are summarized as follows.

- In comparison with the baseline model, the first technique is able to save 33% of the system monetary cost.
- The second technique saves extra 5% (38% in total) by reducing the number of orphaned peers per failure due to the short height of the delivery tree.
- The third technique saves up to 46% by reducing the number of requests.
- Due to the assistance of the cloud, all techniques are able to guarantee the quality of live streaming service.

The remainder of this paper is organized as follows. Section 2 overviews related works. Section 3 formally describes a model of hybrid P2P consisting of multi-tree overlay assisted by the cloud. Section 4 describes the baseline model of live streaming. Section 5 describes the details of the proposed method. Section 6 summarizes the results of simulations. Finally, Section 7 concludes the paper with future work.

2. Related Work

Recently, cloud computing has become a promising platform offering an elastic resource allocation and cost-effective (pay-as-you-go) model. Researchers leveraged the cloud computing for providing different services. A cost-effective content distribution is discussed in [11-12]. Mobile users are supported by the cloud to improve the quality of streaming service in [13-14]. Authors in [15] designed a migration strategy for a video-on-demand (VoD) provider to partially migrate its videos to the cloud trying to minimize its cost. In [16], a mechanism is proposed to retrieve video segments' metadata over the cloud for VoD. CloudStraem [17] transcoded videos to SVC (scalable video coding) over the cloud and delivered them in an adaptive manner to network dynamics through a cloud-based SVC-proxy. That leads the researchers to focus on the efficient scheduling of user media requests to cloud servers. To reduce the response time and cost, the allocation of virtual machines to physical servers by considering the requirements of the cloud media service is discussed in [18]. Authors in [19-20] developed a blind scheduling algorithm for mobile media cloud where no prior knowledge is available on the request rate and service time. Jointly, the algorithm ensures simplicity, fairness among servers and minimized user waiting time. To minimize the cost and assure the streaming quality from a video provider perspective, researchers studied different mechanisms to allocate cloud resources based on the dynamic demand. In [21-22] provisioned resources in the cloud are dynamically scaled up and down to meet the user demands in VoD. Authors in [23] studied how to optimally procure virtual machines for VoD where the cost and quality of experience trade-off is investigated under Amazon pricing models.

Specifically, live-streaming has a high dynamism in peer population and the most stringent latency requirements. There are many proposals concerned with the hybrid of P2P live

streaming with other infrastructures, such as content delivery networks (CDNs) and the cloud computing. Examples of P2P-assisted CDNs include LiveSky [24], PSP [25] and PACDN [26]. In addition, PROSE [27] efficiently utilizes CDN resources by proactively injecting them into the P2P. More concretely, it divides CDN resources into two parts, and uses the first part to serve streaming contents on demand. The second part is used to proactively push streaming contents to several selected peers (super-peers) to improve the overall performance. Unfortunately, provisioning resources in the CDN is semi-static, and such resources could be either insufficient (affects the QoS) or underutilized (extra cost) due to the highly dynamic demand.

In regard to the hybrid cloud-P2P live streaming, CALMS [5] presented a general framework to migrate the live streaming service to the cloud. The cloud servers are rented dynamically from different regions based on a demand prediction mechanism. AngleCast [6] is a typical P2P-assisted cloud, Computing instances (EC2) called angels are deployed on-demand to maintain a predefined high streaming rate. These angels are augmented with an optimized multiple-tree P2P overlay to reduce the cost of cloud resources. A central entity "registrar" is used to orchestrate the multiple-tree overlay. Clive [7] is a cloud-assisted P2P based on a mesh-structured P2P overlay. To guarantee the streaming quality, it rents helpers from the cloud, which are either active or passive (i.e., computing instances or storage service). Clive determines which and how many helpers to be rented so as to maximize the system performance while bounding the cloud cost. Most recently, VMCAST [8] is proposed as a stability enhancing solution for the tree-based multicast overlays based on the cloud virtual machine assistance.

In this paper, we propose a cloud-assisted P2P based on a multi-tree-structured P2P overlay. Unlike mesh-based P2P overlay, multi-tree-structured P2P overlay has the following flaws while it has advantages such as the short playback delay and low overhead: 1) the chunk size is smaller than in mesh-based overlays which increases the number of requests submitted to the cloud to acquire missing chunks; and 2) the leave of a peer causes suspension of the video stream at descendants of the leaving peer which also causes the increase in the number of requests submitted to the cloud. The proposed method scales the bandwidth resources flexibly by asking more/fewer peers to proactively fetch the streaming data from the cloud CDN rather than renting computing instances.

3. System Model

3.1 Overview

This paper focuses on the hybrid of P2P and cloud computing platform. The role of P2P is to deliver live streams issued by a media server to the peers (users) through a P2P overlay. Along with that, the role of cloud platform is to assist the delivery by using a storage service and a cloud content delivery network (CCDN). For the P2P overlay, we adopt a multiple tree structure. Such a structure is weak against peer churn compared with other structures. However, it is efficient in terms of the latency and the message overhead, and it exhibits a good performance with respect to the broadcasting time [28-29] and the maximum streaming rate [30]. In other words, the multiple-tree structure is an appealing structure if we can overcome its shortcomings (e.g., stability and robustness) by combining it with other services such as the cloud. In our model, we consider a P2P overlay consisting of N trees which span all peers. Live stream is a sequence of chunks which are given unique sequence numbers starting from 0. Chunks are equally divided into N sub-streams so that the ith sub-stream

consists of chunks with sequence number $j \equiv i \ (mod\ N)$. The reader should note that by letting r be the bit-rate of the given stream, the bit-rate of each sub-stream is given as r/N. As will be described later, chunks are delivered to the peers in such a way that the ith sub-stream is delivered through the ith tree for each $0 \leq i \leq N-1$.

Let $c(u)$ denote the upload bandwidth of a peer u. Since the bit-rate of each sub-stream is r/N, u can accommodate at most $s(u) \doteq \lfloor c(u)N/r \rfloor$ peers as the "children" in the P2P overlay. $s(u)$ is called the capacity of u and we say that u has a free capacity if it has less than $s(u)$ children. The resource index R of a P2P concerned with the delivery of the given stream is the ratio of the available capacity of the P2P to the capacity which is necessary to deliver N sub-streams to all peers in the P2P, i.e.,

$$R \doteq \frac{\sum_{u \in V} s(u)}{N \times |V|} \tag{1}$$

where V is the set of peers in the P2P. In the following, we are particularly interested in cases such that $R \simeq 1$. Note that in hybrid live streaming systems, the resource index of the P2P indicates the amount of contribution of the cloud platform; i.e., the amount of contribution of the cloud should be large as the value of resource index decreases.

As for the mechanism to detect the leave of adjacent peers in the P2P, we assume that each peer exchanges hello messages with its neighbors for every τ seconds. Similarly, each peer detects the leave of parent in a delivery tree by monitoring the delay of chunks contained in the corresponding sub-stream.

3.2 Cloud Computing Platform

Fig. 1 illustrates the behavior of the cloud platform. At first, it continuously receives a stream of chunks from the media server, and keeps chunks issued in the recent m seconds in the storage service. Each peer in the P2P can request these chunks at any time, and requested chunks are delivered to the requester through edge locations in the CCDN. Such a request is issued by a peer when it detects the missing of a chunk in a sub-stream. A chunk is considered to be missed when the remaining time to its playback point meets a certain threshold. Then, a peer and its descendants may consider same chunks as missed in a close time and acquire them through the CCDN. Moreover, when a peer acquires the missed chunk from the CCDN, it will be busy in forwarding the latest chunks to its children. Hence, in this case, chunks acquired through the CCDN are not forwarded to the children to bound the overhead and redundant data. A different case, as in section 5.2, is that a subset of peers (proactively) issue requests to fetch some (not-yet missed) chunks from CCDN, i.e., to increase the system throughput. Forwarding these chunks to children is necessary to get the benefit of the proactive design.

In addition to the recently issued chunks, in the proposed method, the storage service keeps information concerned with "orphaned" peers. We say that a peer is orphaned if the connection to the parent is lost. See Section 5 for the details.

In the following, we measure the cost of such a cloud assistance by using a pricing model used in actual cloud computing platform. More concretely, we use Amazon's pricing model [31-32] in which the customer should pay for: 1) the amount of data bandwidth fetched from the CCDN, 2) the number of requests issued to the CCDN and 3) the number of requests issued to the storage service. We omit the other costs such as the storage cost, as they are common for all cloud assisted schemes.

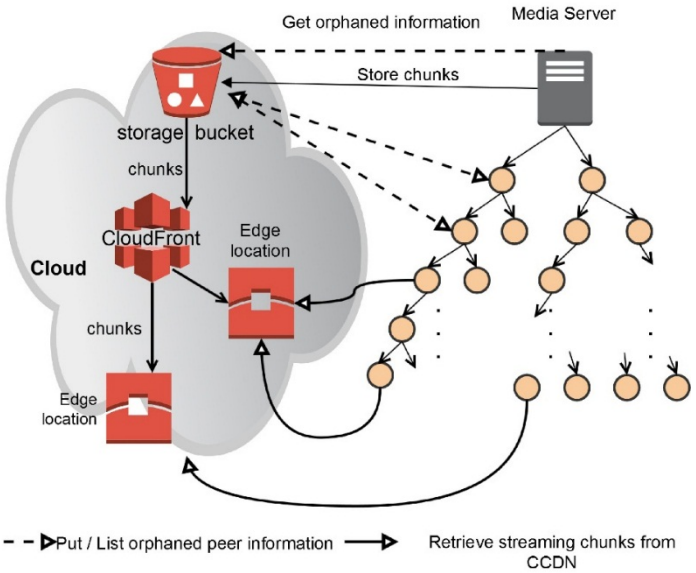

Fig. 1. Cloud-assisted P2P live streaming model.

4. Baseline Model

This section introduces a model of hybrid P2P systems which will be used as a baseline in succeeding sections. This model is a variant of the SplitStream live streaming system [9] in which N delivery trees are organized so that internal nodes of the trees are mutually disjoint. In other words, each peer can join at most one tree as an internal node and the other trees as a leaf node. Let T be the set of N trees. At the time of participation, each peer determines the role of the peer in each tree; i.e., internal or not. Let $\sigma: V \times T \rightarrow \{0,1\}$ be a function indicating the selection made by the peers, where $\sigma(p,t) = 1$ if p joins tree t as a candidate of internal node. Note that in the SplitStream, such a selection is conducted in a random manner.

Assume that peer q submits peer p a request to subscribe to a sub-stream through tree t, where p is a peer with $\sigma(p,t) = 1$. Peer p accepts the request if it has a free capacity. Moreover, p accepts the request if $\sigma(q,t) = 1$ and it has a child k with $\sigma(k,t) = 0$. Then p admits q as a new child after pushing out k from the tree (thus k is orphaned at this time). If it does not accept the request, p forwards the request to a child k' with $\sigma(k',t) = 1$, and such a forwarding is repeated until it reaches a peer which accepts the request or fails to be forwarded to a child. Note that this process always succeeds for q with $\sigma(q,t) = 1$ since any internal node with a deepest level of the tree either has a free capacity or has a leaf child k'' with $\sigma(k'',t) = 0$, while it might fail for q with $\sigma(q,t) = 0$.

In the baseline model, to reduce the length of such forwarding steps, we assume that the first peer p receiving the request from q with $\sigma(q,t) = 1$ is selected in the following manner:

1) if q is an orphaned peer and the depth of q in the tree was two or more before being orphaned, the former grandparent of q is selected as p, and
2) otherwise, a random peer is selected as p.

In addition, if $\sigma(q,t) = 0$, q does not request any p but registers itself to a pool of orphaned peers concerned with tree t, and waits for a response from peers with a free capacity. In the baseline model, the registration message issued by q is collected to the root of tree t' with $\sigma(q,t') = 1$ (i.e., it is sent to the parent of q in tree t' and is forwarded up to the root of

the tree). Such collected messages are forwarded down from the root of tree t, so that they are received by an internal node of t with a free capacity, if any. If q receives several responses, it accepts only one and discards others.

Thus the cost of the baseline model concerned with the cloud assistance is comprised of the number of requests issued to the CCDN and the amount of chunks fetched from the CCDN. Results of detailed evaluation are given in Section 6.

5. Proposed Method

The proposed method consists of three techniques; 1) quick recovery from the status being orphaned with the aid of cloud storage services, 2) proactive fetching of chunks from the CCDN, and 3) the reduction of the number of requests to the CCDN with the notion of frames. In the succeeding subsections, we explain each technique in detail.

5.1 First Technique: Quick Recovery with Storage Service

In hybrid P2Ps, once a peer becomes orphaned, the delivery of chunks to the peer is suspended until it becomes a child of a new parent. During such a suspension, the orphaned peer should fetch missing chunks from the CCDN to keep the playback of the live stream. Thus, a short suspension time will significantly reduce the amount of fetched data.

The basic idea of the first technique is to reduce such a suspension time with the aid of cloud storage services (note that it is experimentally evaluated that general cloud storage services can serve as many requests as it receives [7]). More concretely, when a peer becomes orphaned, it sends a PUT request to a bucket in the cloud storage to add a file concerned with the event. The directory of buckets and the way of authentication should be known to all peers in advance. The file name encodes peer's IP, port, and the name of sub-stream, i.e., it needs to add different files for each tree. On the other hand, any peer with a free capacity can find orphaned peers by sending a LIST request to the bucket which returns a list of file names with necessary metadata. After obtaining it, the free capacity peer tries to accommodate orphaned peers as new children until its capacity is exhausted. Naturally, each orphaned peer which becomes a child of a new parent deletes the corresponding file in the bucket by sending a DELETE request.

Such a match-making mechanism is also used to balance the load of trees. More concretely, we modify the selection of a tree conducted by each newly arrived peer in the baseline model in such a way that it joins the tree as an internal node with the largest number of orphaned peers (recall that such a number of orphaned peers can be easily obtained by sending a LIST request to the bucket).

Additional cost due to the first technique is the number of requests handled by the storage service. Usually, requests to the storage service are more expensive than requests to the CCDN (see [31] and [32] for the example of cloud price in Amazon). Still, as will be evaluated later, the additional cost incurred by the first technique is smaller than the benefit of quick recovery.

5.2 Second Technique: Proactive Bandwidth Investment

The key idea of the second technique is to allow several peers selected from each tree to conduct a proactive fetch of chunks from the CCDN so that the delayed chunks due to the shortage of P2P capacity does not occur. Such selected peers, called cloud peers hereafter, plays the role of a root concerned with the delivery of the corresponding sub-stream. See

Fig. 2 for illustration. As will be evaluated later, with the notion of cloud peers, we could reduce the depth of the overlay and enlarge the available capacity of the delivery tree.

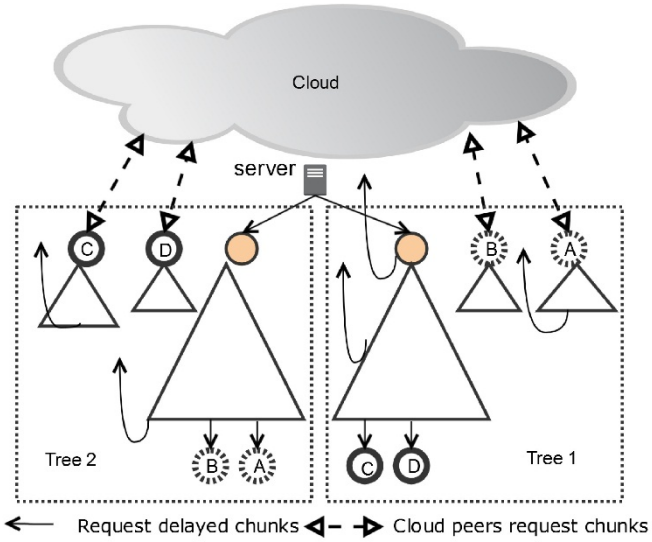

Fig. 2. Proactive bandwidth investment. Assume the estimated P2P shortage is 4. Then, two peers are selected in each tree to act as cloud peers, where peers A and B fetch sub-stream 1 from the CCDN, and peers C and D fetch sub-stream 2.

The number of cloud peers is related to P2P shortage and determined as follows (see Step 1 of **Algorithm 1**). At first, the server periodically issues a LIST request to the storage service to acquire the latest list O of orphaned peers. A request and computational overhead is incurred when the length of such a period is short (few seconds). On the other hand, a long period might make it difficult to follow the dynamic change in the P2P shortage. In our implementation we found a period of 30 seconds as a good balance. Since the list O might not accurately reflect the shortage of the P2P capacity (i.e., $N \times |V| \leq \sum_{u \in V} s(u)$), the server identifies a subset $L(\subseteq O)$ which have been orphaned for a time longer than τ_n, and regards it as a subset of *actual orphaned peers due to P2P shortage*. A typical value of τ_n is four seconds, while as for the selection of the value of τ_n, there is a tradeoff between the cost of proactive fetching and the cost of reactive fetching. Automatic adjustment of parameter τ_n is left as a challenging future work.

If $|O| > \theta$ holds for a predetermined threshold θ, and all trees in T contain at least one peer belonging to subset L, then the server selects $|L|/N$ random peers from each tree and asks them to act as cloud peers (see Step 2 of **Algorithm 1**). The reader should note that the latter condition is necessary to exclude extreme cases in which the distribution of peers in L across trees is highly imbalanced since such an imbalance could be naturally resolved by accepting more peers to the system without investing cloud peers. If $|O| \leq \theta'$ for some $\theta' \leq \theta$, on the other hand, the server eliminates a set of cloud peers to reduce the cost of proactive fetching. Eliminated cloud peers rejoin their tree, while keeping to fetch chunks from the CCDN till they become a child of new parents. A typical value of threshold θ is $10N$, which implies that it allows each tree to have ten orphaned peers on average. In the implementation, $5N$ cloud peers are eliminated when $|O| \leq \theta'$. Thus, each tree will have new five orphaned peers to be adopted by the free capacity peers. Selecting these values as a multiple of N allows us to have a fair comparison between the second and third technique (section 5.3) by having the same ratio of selected to eliminated cloud peers.

Algorithm 1. Selection of cloud peers.

$\boldsymbol{O} \leftarrow$ The set of all orphaned peers
$\boldsymbol{L} \leftarrow \emptyset$
$\boldsymbol{N} \leftarrow$ The number of trees (sub-streams)
$\boldsymbol{C(N)} \leftarrow$ Array represents the set of cloud peers in each tree.
$I(N) \leftarrow$ Array represents the set of internal peers in each tree.
$(C[i] \subset I[i]$ for $0 \leq i \leq N-1)$

//Step 1: Determine the number of cloud peers.
for each peer *j* **in** O **do**
  $\boldsymbol{t_j} \leftarrow$ the time when peer *j* was registered as an orphaned peer
  if *current_time* $- \boldsymbol{t_j} > \tau_n$ **then**
    $\boldsymbol{L} \leftarrow \boldsymbol{L} \cup \{\boldsymbol{j}\}$
  end if
end for
// $|L|$ represents the estimated shortage of P2P capacity

//Step 2: adding / removing cloud peers.
$\boldsymbol{\theta} \leftarrow \boldsymbol{10N}$
if $|\boldsymbol{O}| > \boldsymbol{\theta}$ **and** every tree contains at least one peer in $\boldsymbol{L}$ **then**
  for $\boldsymbol{i = 0}$ **to** $\boldsymbol{N-1}$ **do**
    // select new $|L|/N$ cloud peers in each tree.
    $U \leftarrow$ a set of $|L|/N$ peers randomly chosen form $I[i] \setminus C[i]$
    $\boldsymbol{C[i]} \leftarrow \boldsymbol{C[i]} \cup \boldsymbol{U}$
  end for
else
  if $|\boldsymbol{O}| \leq \boldsymbol{\theta'}$ **then**
    //eliminate $5N$ cloud peers from each tree
    for $\boldsymbol{i = 0}$ **to** $\boldsymbol{N-1}$ **do**
      $U' \leftarrow$ a set of 5 peers randomly chosen form $C[i]$
      $\boldsymbol{C[i]} \leftarrow \boldsymbol{C[i]} \setminus \boldsymbol{U'}$
    end for
  end if
end if

The cost due to the second technique is similar to the first technique, while it has an apparent advantage so that the depth of the trees becomes much smaller and a disadvantage so that the server should manage the set of cloud peers.

5.3 Third Technique: Less Requests to the CCDN

The objective of the third technique is to reduce the number of requests issued to the CCDN with the notion of frames. More concretely, when a cloud peer proactively fetches chunks

from the CCDN, it requests a frame consisting of F consecutive chunks instead of individual chunks (see **Fig. 3** for illustration). In addition, after being selected as a cloud peer for tree t, it fetches chunks corresponding to the full stream from the CCDN, while it forwards chunks corresponding to a sub-stream associated with tree t to the children in the tree. Then to keep the amount of data fetched from the CCDN, we decrease the number of cloud peers from $|L|/N$ to $|L|/N^2$. See **Fig. 4** for illustration.

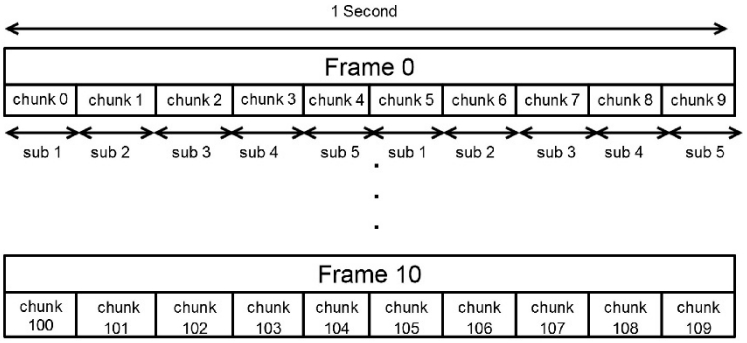

Fig. 3. The structure of frames consisting of 10 consecutive chunks. In this example, each frame corresponds to a part of the stream of 1 second.

Similar effect to the notion of frames could be obtained by increasing the size of each chunk. However, in tree-based streaming systems, it is preferable to have small chunks since each peer needs to receive the whole chunk before forwarding it to the children. In fact, IP packet of 1 Kbyte length is used as the basic chunk in the SplitStream [9]. This is in contrast to mesh-based systems which use large chunks (60 Kbytes in [1], 14 Kbytes in some PPLive channels [33]) to avoid excessive overhead of signaling per chunk.

The increase of frame size F reduces the number of requests issued to the CCDN inverse proportionally. For example, assume that the stream carries 20 chunks per second and the server invested 20 cloud peers, i.e., the CCDN should receive 400 requests per second from cloud peers under the second technique. If the frame size is set to 10, which corresponds to 0.5 second, the number of requests reduces to 40 per second (90 % save), and if the frame size is set to 40, it reduces to 10 per second (97.5 % save). A disadvantage of the large frame size is the waste of chunks which occurs when the set of cloud peers changes. For example, by scaling the number of cloud peers down, some of them need to rejoin all trees by finding new parents. Such new parents may not have received the latest chunks yet so they will forward duplicated chunks. Also, when asking a peer to act as a cloud peer, that peer will request the latest frame which may include some of already received chunks. Another way of wasting the chunks is when a cloud peer leaves the system after asking a large frame from the CCDN. In the evaluation given in the next section, we set the value of F to two seconds.

6. Evaluation

To evaluate the performance of the proposed method, we conducted extensive simulations using an event-driven packet level simulator encoded in C++. The simulator is implemented on top of a peer to peer simulator [10]. It uses a real world node-to-node latency matrix measured on the Internet [34] with the average end-to-end delay 79 ms, while it does not consider the queuing management in routers.

In the following, we compare the following four schemes:

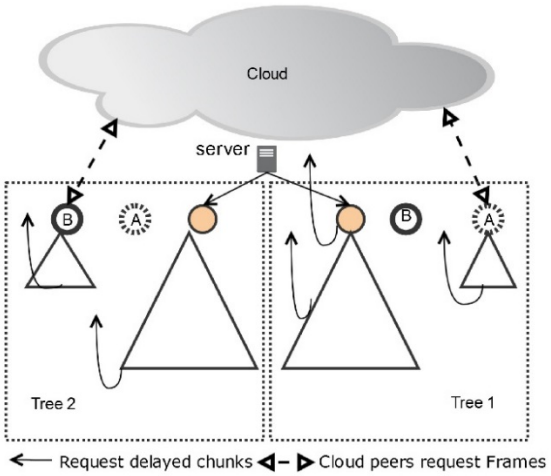

Fig. 4. Third Technique. Assume $N = 2$ and the P2P shortage is four sub-streams. One peer is selected as a cloud peer for each tree. Peer A (resp. peer B) fetches the full stream from the CCDN and acts as a root in the first tree (resp. the second tree).

- cloud-assisted P2P live streaming scheme Baseline which is based on the baseline model;
- scheme Orphan which is based on the first technique;
- scheme Proactive which is based on the first and second techniques; and
- scheme Frame which is based on the first, second and third techniques.

As for the metrics for evaluation, we consider: 1) the amount of data fetched from the CCDN (Section 6.2), 2) the number of requests handled by the cloud (Section 6.3), and 3) the quality of live stream including the delivery ratio and the average playback delay (Section 6.5). We also calculate the monetary cost of the schemes by assuming the cloud price of AWS (Section 6.4).

6.1 Setup

In the simulations, we consider a video stream of 1000 Kbps which is the bit-rate recommended by YouTube[2] for the 480p video quality. The chunk size is 6250 bytes and the number of sub-streams is $N = 5$ (i.e., each sub-stream carries 4 chunks per second). Each peer starts playback after buffering the stream for 15 seconds, and it sends a request for a chunk to the CCDN when the remaining time before playing back the chunk becomes 2 seconds. The frame size in Frame is set to $F = 40$, i.e., each frame corresponds to a part of the stream of 2 seconds.

The number of peers is 1000 and the upload bandwidth of the server is 2 Mbps. To reflect the heterogeneity of peers, we consider the distribution of upload bandwidth summarized in **Table 1** (based on [35]); e.g., the upload bandwidth of 147 peers is 512 Kbps which indicates that the capacity of these peers is $\lfloor 512/200 \rfloor = 2$. Each peer possesses a download bandwidth exceeding 1000 Kbps to enjoy the live streaming. The resource index calculated from the above parameters is $R = 0.944$, which is slightly less than 1.00. Thus, the shortage 0.056 (= 1.000 – 0.944) must be compensated by the CCDN.

[2] https://support.google.com/youtube/

Table 1. Distribution of the upload bandwidth of peers.

Bandwidth [Kbps]	384	512	768	1024	3000
Share [%]	32.9	14.7	8	28.1	16.3
Capacity	1	2	3	5	15

The simulation time is fixed to 900 seconds. To realize a dynamic behavior of peers, we use real P2P traces as in [2], in which the number of online peers gradually increases from 0 to around 1000 and then reaches a stable state in which 1% of peers join/leave in each second. See **Fig. 5** for illustration. Among leaving peers, 95% of them gracefully leave and 5% of them ungracefully leave [36], where the latter is detected by observing the lag of chunks received from the parent and the missing of heartbeat messages transmitted by the children every 5 seconds. We conduct such a simulation run 15 times and take an average. The total download time per simulation run is 13025 min on average (note that it is smaller than 15000 min (= 900 × 1000/60) due to churn). In addition, the total amount of chunks downloaded by the peers is 93.16 GB, which implies that 5.21 GB of chunks corresponding to 0.056 (= 1 – 0.944) of downloaded chunks must be fetched from the CCDN, where 0.944 is the resource index of the hybrid P2P.

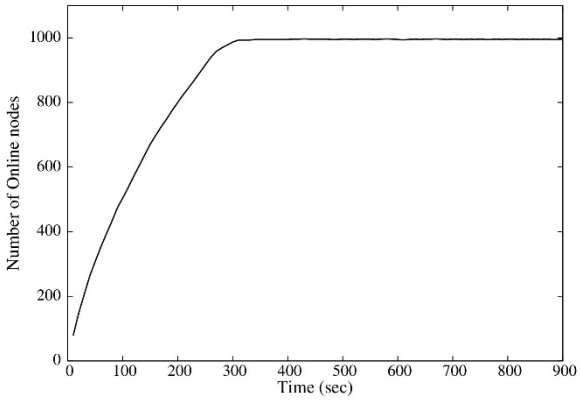

Fig. 5. Number of online peers throughout the simulation.

6.2 Amount of Data Fetched from the CCDN

At first, we evaluate the amount of data fetched from the CCDN. **Fig. 6** summarizes the results, where the horizontal axis is the elapsed time and the vertical axis is the fetch rate per second. The temporal variance of the fetch rate is due to the join/leave of peers and we could observe that the proposed schemes reduce the fetch rate of Baseline by about 40 %. The total amount of fetched data during simulation is shown in **Fig. 7**. Since at least 5.21 GB of chunks must be fetched from the CCDN, the proposed techniques certainly mitigate additional fetch due to churn. For example, although it is 7.28 (= 12.49 – 5.21) GB under Baseline, it significantly reduces to 1.33 (= 6.54 – 5.21) GB under Proactive.

The badness of Frame with respect to additional fetches is because of the redundancy mainly caused by the large frame size. **Fig. 8** shows the fraction of duplicated chunks among received chunks, which indicates that under Frame, 18 chunks among received 10000 chunks are duplicated. As a reason for the redundancy is the leave of cloud peers which frequently occurs in Proactive and Frame. More specifically, when a cloud peer leaves, its child who has received most recent chunks from the cloud peer might connect to a new parent which did

not receive the latest chunks yet. In such a case, the new parent forwards duplicated chunks to the new child. The large frame size in Frame increases the redundancy as explained in section 5.3.

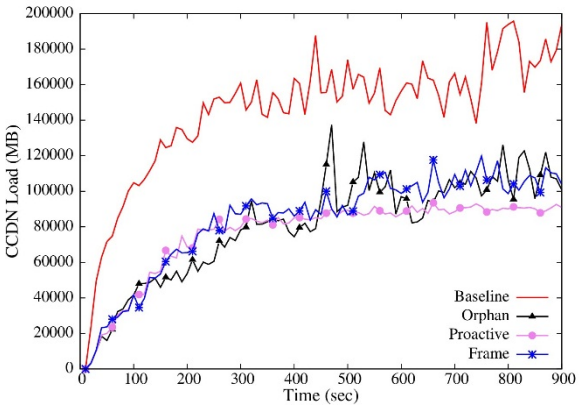

Fig. 6. Time transition of the fetch rate from the CCDN.

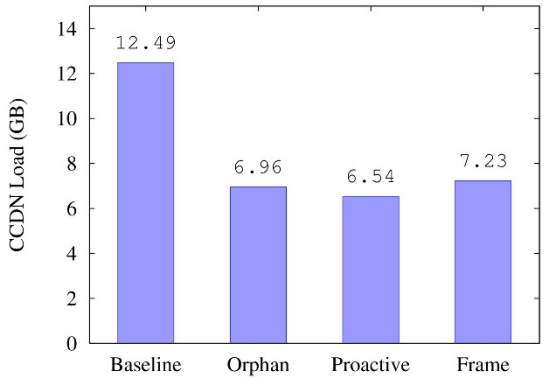

Fig. 7. Total amount of data fetched from the CCDN.

To clarify the difference of Proactive and Frame in more detail, we separately evaluate data fetches conducted by cloud peers and the other peers. **Fig. 9** summarizes the result. The result shows that the amount of fetch conducted by cloud peers is almost the same, i.e., 3.87 GB for Proactive and 3.73 GB for Frame, but the amount of fetches conducted by the other peers to mitigate churn differs, i.e., it is 2.98 GB for Proactive which is apparently less than 3.84 GB for Frame. Such a difference is due to: 1) the redundancy of Frame as was observed in **Fig. 8** and 2) the difference of average hop-count from root to the peers which is 3.07 in Proactive and 5.43 in Frame (recall that Frame invests less cloud peers than Proactive).

6.3 Number of Requests Handled by the Cloud

The number of requests handled by the cloud storage service is shown in **Fig. 10**. The proposed schemes issue more requests than Baseline since they exploit the storage service to realize a quick match-making between orphaned peers and free capacity peers. The difference of three proposed schemes is mainly due to the difference of average hop-count, since shorter hop-count implies that fewer peers are affected by churn. In fact, the result shows that a scheme with a smaller average hop-count issues fewer requests; e.g., Proactive issues the least and Frame issues the second least.

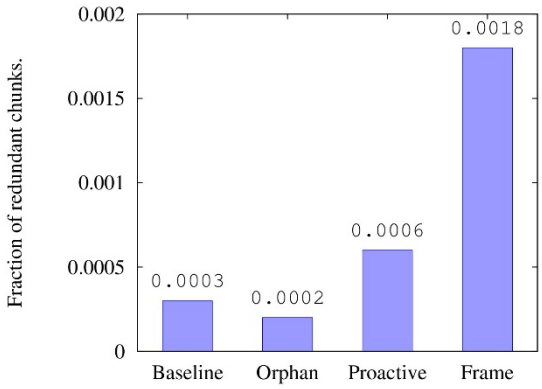

Fig. 8. Fraction of redundant chunk.

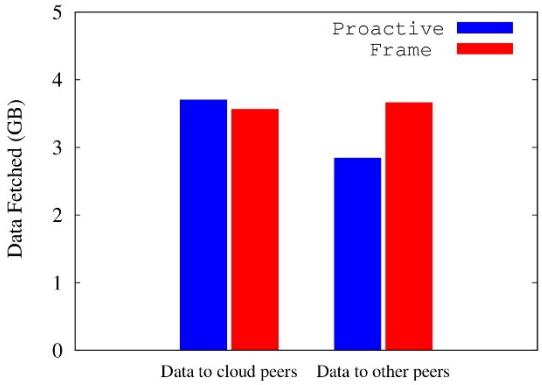

Fig. 9. The amount of fetches conducted by cloud peers and the other peers.

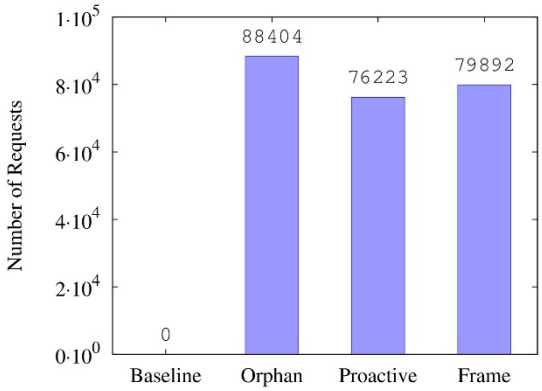

Fig. 10. The number of requests handled by the cloud storage service.

The number of requests handled by the CCDN is shown in **Fig. 11**. The difference among Baseline, Orphan and Proactive is comparable to the difference of the amount of fetched data observed in **Fig. 7**. However, the number of requests issued in Frame is much smaller than the others (e.g., it saves 42.5% of requests compared with Proactive), which is apparently because of the effect of introducing frames.

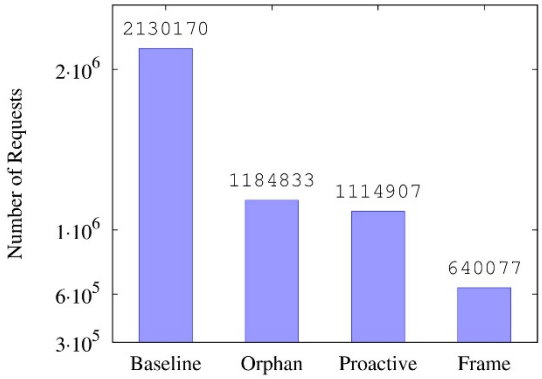

Fig. 11. The number of requests handled by the CCDN.

Table 2. AWS prices in Tokyo region

Requests	**Price**
To the storage service	$ 0.0047 per 1000 requests
To the CCDN	$ 0.0090 per 10000 requests
Data transfer out of the CCDN	**Price**
If <10 TB per month	$ 0.14 per GB

6.4 Total Monetary Cost of the Cloud-Assistance

In this subsection, we evaluate the monetary cost of the cloud-assistance for each scheme. **Table 2** shows the AWS price in Tokyo region [31-32][3], which indicates that: 1) requests to the cloud storage service take $\$ 4.7 \times 10^{-6}$ per request, 2) requests to the CCDN take $\$ 0.9 \times 10^{-6}$ per request, and 3) data fetch takes $\$ 0.14$ per GB. Let R_s be the number of requests handled by the storage service, R_c the number of requests handled by the CCDN, and D the amount of fetched data in GB. Then the total monetary cost of a scheme is calculated as

$$C(\$) = 4.7 \times 10^{-6} \times R_s + 0.9 \times 10^{-6} \times R_c + 0.14 \times D \quad (2)$$

By substituting values obtained in previous subsections, we have the monetary cost of each scheme as is shown in **Fig. 12**. From the figure, we can make the following observations:

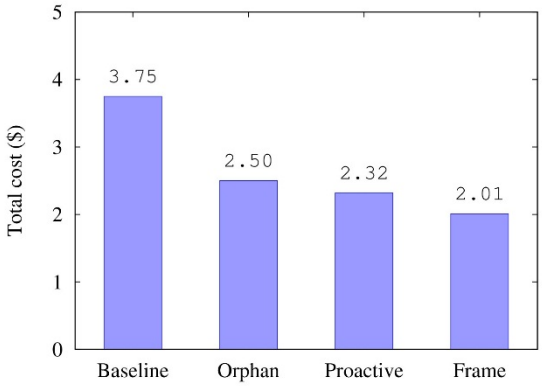

Fig. 12. Total cost ($).

[3] Billing model and prices may change over time.

1) Orphan saves the cost of Baseline by 33.3%. This implies that the benefit of quick recovery is larger than the cost of requests handled by the cloud storage service.
2) Proactive saves the cost of Baseline by 39.5%. The improvement of Proactive is because of the proactive fetch conducted by the cloud peers and the fewer orphaned peers due to the short average hop-count.
3) Frame saves the cost of Baseline by 46%. This is due to the reduction of the number of requests handled by the CCDN.

6.5 Playback Delay & Delivery Ratio

Finally, we evaluate the quality of live stream delivered to the peers in terms of the average delivery ratio and the average playback delay. The delivery ratio is the ratio of chunks received by the playback deadline among chunks transmitted by the server. The playback delay is the delay between the time when a chunk is sent out from the server and the time when it is played by the peer. The maximum playback delay is the time by which the chunk is played by all peers. Values of these metrics are sampled every 10 seconds and averaged at the end of each simulation run.

Table 3 summarizes the results. The results indicate that all schemes attain a sufficiently high delivery ratio 0.999 and a short maximum playback delay of 13.5 seconds. This is due to the behavior of peers such that they fetch chunks from the cloud when the time before playback becomes 2 seconds. Note that 13.5 seconds is shorter than the buffering time which is set to 15 seconds in the simulation. As for the average playback delay, we can make the following observations. At first, Proactive attains an average delay of 5.45 seconds thanks to the short depth of the delivery tree. However, Frame is worse than Proactive by 1.37 seconds and Orphan is worse than Frame by 1.47 seconds. The badness of Orphan is due to the behavior of peers so that they fetch chunks "after" detecting that the time before playback is 2 seconds. Frame's badness is due to the longer depth of the delivery tree and the longer time taken by cloud peers to download a frame before starting the forward of the chunks to the children. **Fig. 13** shows the time transition of the average delay of each scheme.

Table 3. Quality metrics of the live stream in each scheme.

	Baseline	Orphan	Proactive	Frame
Avg. delivery ratio	0.999	0.999	0.999	0.999
Max. playback delay [sec.]	13.561	13.548	13.377	13.447
Avg. playback delay [sec.]	9.944	8.297	5.457	6.826

7. Conclusion

This paper proposes three techniques to reduce the cost of cloud-assistance in hybrid P2P live streaming. At first, we exploit the cloud storage service to speed up the match-making between orphaned peers and peers with a free upload capacity. Then, we invest a set of cloud peers to proactively fetch chunks from the cloud, where the number of cloud peers is dynamically adjusted based on the estimated shortage of the P2P. Finally, to reduce the number of requests handled by the cloud, we introduce the notion of frame to allow cloud peers to fetch a collection of chunks using a single request. The simulation results indicate that the resulting scheme reduces the monetary cost of conventional cloud-assisted P2P by about 46% while guaranteeing a high streaming quality.

As a future work, we need to refine the estimation of P2P shortage providing a method to dynamically set its parameters.

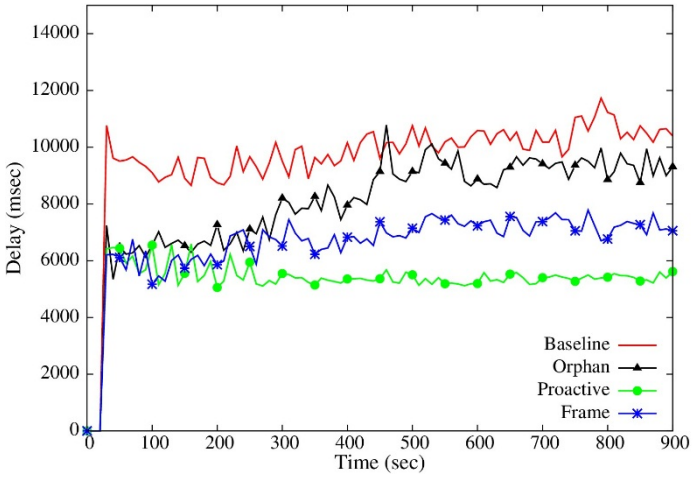

Fig. 13. Time transition of the average playback delay.

References

[1] Xinyan Zhang, Jiangchuan Liu, Bo Li and Tak-Shin Peter Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. of IEEE INFOCOM*, pp. 2102-2111, March 13-17, 2005. Article (CrossRef Link)

[2] Meng Zhang, Qian Zhang, Lifeng Sun and Shiqiang Yang, "Understanding the power of pull-based streaming protocol: can we do better?," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1678-1694, December, 2007. Article (CrossRef Link).

[3] Yang-hua Chu, Sanjay G. Rao and Hui Zhang, "A case for end system multicast," in *Proc. of ACM Int. Conf. on Measurement and Modeling of Computer Systems*, pp. 1-12, June 17-21, 2000. Article (CrossRef Link)

[4] Chuan Wu, Baochun Li and Shuqiao Zhao, "Diagnosing network-wide P2P live streaming inefficiencies," in *Proc. of IEEE INFOCOM*, pp. 2731-2735, April 19-25, 2009. Article (CrossRef Link)

[5] Feng Wang, Jiangchuan Liu and Minghua Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. of IEEE INFOCOM*, pp. 199-207, March 25-30, 2012. Article (CrossRef Link)

[6] Raymond Sweha, Vatche Ishakian and Azer Bestavros, "AngelCast: Cloud-based peer-assisted live streaming using optimized multi-tree construction," in *Proc. of the 3rd Multimedia Systems Conf.*, pp. 191-202, February 22-24, 2012. Article (CrossRef Link)

[7] Amir H. Payberah, Hanna Kavalionak, Vimalkumar Kumaresan, Alberto Montresor and Seif Haridi, "CLive: Cloud-assisted P2P live streaming," in *Proc. of the IEEE Int. Conf. on Peer-to-Peer Computing*, pp. 79-90, September 3-5, 2012. Article (CrossRef Link)

[8] Weidong Gu, Xinchang Zhang, Bin Gong, Wei Zhang and Lu Wang, "VMCAST: A VM-assisted stability enhancing solution for tree-based overlay multicast," *PLoS ONE*, vol. 10, no. 11, online, November, 2015. Article (CrossRef Link)

[9] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron and Atul Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. of the 19th ACM Symposium on Operating Systems Principles*, pp. 298-313, October 19-22, 2003. Article (CrossRef Link)

[10] Meng Zhang, Peer to Peer streaming simulator, [online] (accessed May 2015). URL: http://media.cs.tsinghua.edu.cn/~zhangm/

[11] Alberto Montresor and Luca Abeni, "Cloudy weather for P2P, with a chance of gossip," in *Proc. of the IEEE Int. Conf. on Peer-to-Peer Computing*, pp. 250-259, August 31-September 2, 2011. Article (CrossRef Link)
[12] Pietro Michiardi, Damiano Carra, Francesco Albanese and Azer Bestavros, "Peer-assisted content distribution on a budget," *Computer Networks*, vol. 56, no. 7, pp. 2038-2048, May, 2012. Article (CrossRef Link)
[13] Xin Jin and Yu-Kwong Kwok, "Cloud assisted P2P media streaming for bandwidth constrained mobile subscribers," in *Proc. of 16th IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, pp. 800-805, December 8-10, 2010. Article (CrossRef Link)
[14] Xiaofei Wang, T.T. Kwon, Yanghee Choi, Haiyang Wang and Jiangchuan Liu, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 72-79, June, 2013. Article (CrossRef Link)
[15] Haitao Li, Lili Zhong, Jiangchuan Liu, Bo Li and Ke Xu, "Cost-effective partial migration of VoD services to content clouds," in *Proc. of IEEE Int. Conf. on Cloud Computing (CLOUD)*, pp. 203-210, July 4-9, 2011. Article (CrossRef Link)
[16] Vladimir Rocha, Fabio Kon, Raphael Cobe and Renata Wassermann, "A hybrid cloud-P2P architecture for multimedia information retrieval on VoD services," *Computing*, online, September, 2014. Article (CrossRef Link)
[17] Zixia Huang, Chao Mei, Li Erran Li and Thomas Woo, "CloudStream: Delivering high quality streaming videos through a cloud-based SVC proxy," in *Proc. of IEEE INFOCOM*, pp.201-205, April 10-15, 2011. Article (CrossRef Link)
[18] Mohammad M. Hassan, Biao Song, Ahmad Almogren, M. Shamim Hossain, Atif Alamri, Mohammed Alnuem, Mohammad M. Monowar and M. Anwar Hossain, "Efficient virtual machine resource management for media cloud computing," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 5, pp. 1567-1587, May, 2014. Article (CrossRef Link)
[19] Liang Zhou and Haohong Wang, "Toward blind scheduling in mobile media cloud: fairness, simplicity, and asymptotic optimality," *IEEE Transaction on Multimedia*, vol. 15, no. 4, pp. 735-746, June, 2013. Article (CrossRef Link)
[20] Liang Zhou, Zhen Yang, Joel J. P. C. Rodrigues and Mohsen Guizani, "Exploring blind online scheduling for mobile cloud multimedia services," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 54-61, June, 2013. Article (CrossRef Link)
[21] Yu Wu, Chuan Wu, Bo Li, Xuanjia Qiu and Francis C. M. Lau, "Cloudmedia: When cloud on demand meets video on demand," in *Proc. of the Int. Conf. on Distributed Computing Systems (ICDCS)*, pp. 268–277, June 20-24, 2011. Article (CrossRef Link)
[22] Di Niu, Hong Xu, Baochun Li and Shuqiao Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. of IEEE INFOCOM*, pp. 460-468, March 25-30, 2012. Article (CrossRef Link)
[23] Jian He, Yonggang Wen, Jianwei Huang and Di Wu, "On the cost-QoE tradeoff for cloud-based video streaming under amazon EC2's pricing models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 669–680, April, 2014. Article (CrossRef Link)
[24] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang and Bo Li, "Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky," in *Proc. of the 17th ACM Int. Conf. on Multimedia*, pp. 25-34, October 19-24. 2009. Article (CrossRef Link)
[25] Zhijia Chen, Hao Yin, Chuang Lin, Xuening Liu and Yang Chen, "Towards a trustworthy and controllable peer-server-peer media streaming: an analytical study and an industrial perspective," in *Proc. of IEEE Global Telecommunications Conf.*, pp. 2086-2090, November 26-30, 2007. Article (CrossRef Link)
[26] Xuening Liu, Hao Yin, Chuang Lin, Yu Liu, Zhijia Chen and Xin Xiao, "Performance analysis and industrial practice of peer-assisted content distribution network for large scale live video streaming," in *Proc. of the 22nd Int. Conf. on Advanced Information Networking and Applications*, pp.568-574, March 25-28, 2008. Article (CrossRef Link)

[27] Zhi-Hui Lv, Li-Jiang Chen, Jie Wu, Da Deng, Si-Jia Huang and Yi Huang, "PROSE: Proactive, selective CDN participation for P2P streaming," *Journal of Computer Science and Technology*, vol. 28, no. 3, pp. 540-552, May, 2013. Article (CrossRef Link)
[28] Satoshi Fujita, "Optimal serial broadcast of successive chunks", *Theoretical Computer Science*, vol. 574, pp. 3-9, March, 2015. Article (CrossRef Link)
[29] Giuseppe Bianchi, N. Blefari Melazzi, Lorenzo Bracciale, Francesca Lo Piccolo and Stefano Salsano, "Streamline: An optimal distribution algorithm for peer-to-peer real-time streaming," *IEEE Transactions on Parallel and distributed systems*, vol. 21, no. 6, pp. 857-871, June, 2010. Article (CrossRef Link)
[30] Shao Liu, Rui Zhang-Shen, Wenjie Jiang, Jennifer Rexford and Mung Chiang, "Performance bounds for peer-assisted live streaming," in *Proc. of SIGMETRICS*, pp. 313-324, June 2-6, 2008. Article (CrossRef Link)
[31] Amazon CloudFront pricing [online] (accessed May 2015). URL: http://aws.amazon.com/cloudfront/pricing
[32] Amazon S3 Pricing [online] (accessed May 2015). URL: http://aws.amazon.com/s3/pricing/
[33] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu and Keith W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672-1687, December, 2007. Article (CrossRef Link)
[34] Meridian node to node latency matrix (2500x2500), meridian project [online] (accessed May 2015). URL: http://www.cs.cornell.edu/People/egs/meridian/data.php
[35] Zhengye Liu, Yanming Shen, Keith W. Ross, Shivendra S. Panwar and Yao Wang, "Substream trading: Towards an open P2P live streaming system," in *Proc. of the IEEE Int. Conf. on Network Protocols*, pp. 94-103, October 19-22, 2008. Article (CrossRef Link)
[36] Bo Li, Susu Xie, Gabriel Y. Keung, Jiangchuan Liu, Ion Stoica, Hui Zhang, and Xinyan Zhang, "An empirical study of the Coolstreaming+ system," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1627-1639, December, 2007. Article (CrossRef Link)

Bahaa Aldeen ALGHAZAWY received the B.E. degree in telecommunication engineering from the University of Aleppo in 2009, and the M.E. degree in information engineering form Hiroshima University in 2013. He is currently a Ph.D. candidate at the Department of Information Engineering, Hiroshima University. His research interests are in the area of internet, peer-to-peer networks, multimedia networking and cloud computing.

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, the Institute of Electronics, Information and Communication Engineers, SIAM Japan, IEEE Computer Society, and SIAM.